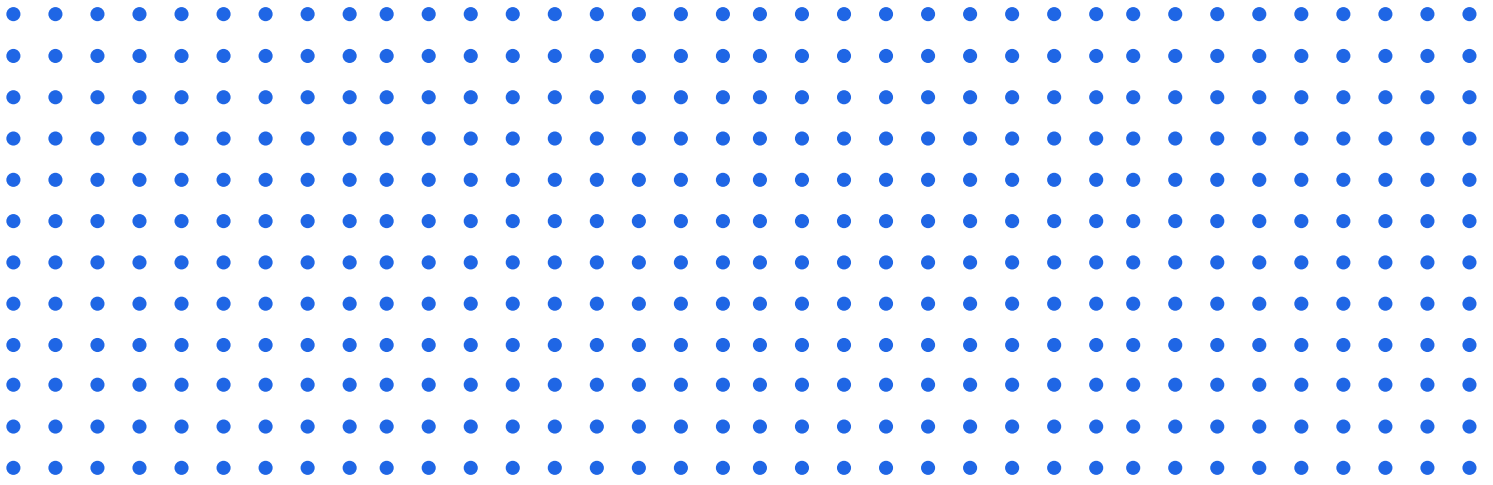




App Studio

Version 4.0

Table of Contents

- *About Legacy Docs*
- *App Studio Reference Guides*
 - *Services*
 - *\$filterService*
 - *\$lightningLog*
 - *\$lightningUrl*
 - *\$twigkit*
 - *\$util*
 - *BookmarkService*
 - *CollaborationService*
 - *DateUtils*
 - *FieldService*
 - *ModalService*
 - *PlatformService*
 - *QueryIdService*
 - *QueryService*
 - *ResponseService*
 - *ScrollToService*
 - *TogglePanelsService*
 - *TranslationService*
 - *UserService*
 - *httpInterceptor*
 - *scrollAndResizeListener*
 - *Accessing Data*
 - *Configuring the Search Platform*
 - *Filter Syntax*
 - *Formulating a Query*
 - *Search Responses*
 - *Modules*
 - *Activity Tracking*
 - *Visualizations*
 - *Adding Credits*
 - *Chart Legend*
 - *Chart Tooltips*
 - *Displaying a Chart*
 - *Exporting a Chart*
 - *Faceting a Data Series*
 - *Histogram*
 - *Search Result Data Series*
 - *Tag Cloud*
 - *Security*
 - *Authentication*
 - *OAuth*
 - *SAML Configuration*
 - *Troubleshooting SAML*
 - *Role-based Access*
 - *User Web Service*
 - *Geospatial Module*
 - *Esri Web Maps*
 - *Maps*
 - *Media Module*
 - *Export Services*
 - *Image Formatting and Transcoding*
 - *Workflow and Data Processing Module*
 - *Response Processors*
 - *Facet Processors*
 - *Field Processors*
 - *Filter Processors*
 - *General Response Processors*
 - *Query Processors*
 - *Social and Collaboration Tools*
 - *App Studio Gallery*
 - *Button*

- *Checkbox*
- *Color*
- *Comment*
- *Field*
- *Form*
- *Grid*
- *Icons*
- *Input*
- *Loader*
- *Message*
- *Pagination*
- *Panel*
- *Progress Bars*
- *Result List*
- *Table*
- *Tabs and Toggle Panes*
- *Tag*
- *Timeline*
- *Tooltip*
- *Typography*
- *FAQ*
 - *App Studio Versions*
 - *Devices and Browser Support*
 - *Empty Facets in Solr*
 - *Performance*
 - *Responsive Design*
 - *Space for Log Files*
- *Deployment*
 - *Building from Source*
- *Search Platforms*
 - *Solr*
 - *Configuring the Hierarchical Facet Service*
 - *Indexing Hierarchical Taxonomies*
 - *Query Suggestions*
 - *SolrCloud*
 - *Elasticsearch*
 - *Google Custom Search*
 - *Lucidworks Fusion*
 - *Offline*
 - *Pipl*
 - *Shared Platform Attributes*
 - *Twitter*
 - *Wikipedia*
 - *WolframAlpha*
 - *YouTube*
- *Filters*
 - *Filter Syntax*
 - *actual*
 - *actualValue*
 - *dateFormat*
 - *deepFilter*
 - *display*
 - *displayValue*
 - *encodeURIComponent*
 - *facet*
 - *fact*
 - *field*
 - *filterCount*
 - *filterValue*
 - *formatAsCssName*
 - *getUrlForQuery*
 - *hasFilter*
 - *htmlToPlainText*
 - *maxCharacters*
 - *numberFormat*
 - *orderByValue*
 - *placeholder*

- *property*
- *pyramidSort*
- *rangeDateFormat*
- *result*
- *roundedNumber*
- *translate*
- *trusted*
- *Installation*
 - *Open Source Libraries*
 - *Software Requirements*
 - *Supported Devices and Browsers*
- *Licensing*
- *Styles*
- *Tags*
 - *advanced:button*
 - *advanced:date-range*
 - *advanced:text*
 - *block:interruptor*
 - *chart:credit*
 - *chart:display*
 - *chart:export*
 - *chart:gauge*
 - *chart:histogram*
 - *chart:legend*
 - *chart:tagcloud*
 - *chart:timeline*
 - *chart:tooltip*
 - *class:toggle*
 - *click:outside*
 - *collaborate:bookmark*
 - *collaborate:bookmark-list*
 - *collaborate:like*
 - *collaborate:saved-query-list*
 - *collaborate:topic-form*
 - *collaborate:topic-list*
 - *content:boolean*
 - *content:form*
 - *content:input*
 - *content:radio*
 - *content:submit*
 - *content:textarea*
 - *controller*
 - *default:src*
 - *dev:track-digests*
 - *drawer*
 - *dynamic:styling*
 - *esri:heatmap*
 - *esri:webmap*
 - *facet:dropdown*
 - *facet:hierarchical*
 - *geo:heatmap*
 - *geo:highmaps*
 - *geo:layer*
 - *geo:map*
 - *graph:flow*
 - *helper:constant*
 - *helper:set*
 - *helper:title*
 - *hide:when*
 - *instant:result-list*
 - *layout:animate*
 - *layout:block*
 - *layout:box*
 - *layout:center-content*
 - *layout:content*
 - *layout:controls*
 - *layout:grid*

- *layout:include*
- *layout:logo*
- *layout:sidebar*
- *lifecycle:delay*
- *lifecycle:display*
- *lifecycle:event*
- *lifecycle:on*
- *lifecycle:required*
- *media:csv*
- *media:excel*
- *media:image*
- *ng:compile*
- *ng:inject*
- *padding:bottom*
- *padding:left*
- *padding:right*
- *padding:top*
- *pivot:table*
- *progress:bar*
- *progress:radial*
- *query:custom*
- *query:filter*
- *query:suggestions*
- *search:advanced*
- *search:alert*
- *search:box*
- *search:breadcrumbs*
- *search:conditional*
- *search:facet*
- *search:facet-list*
- *search:field*
- *search:fields*
- *search:filter*
- *search:filter-list*
- *search:keyword*
- *search:load*
- *search:message-list*
- *search:no-results*
- *search:offcanvas-toggle*
- *search:pagination*
- *search:platform*
- *search:query*
- *search:response*
- *search:response-statistics*
- *search:result*
- *search:result-list*
- *search:results-per-page*
- *search:sort*
- *search:tabs*
- *search:typeahead*
- *security:user-actions*
- *series:facet*
- *series:pivot*
- *series:result-list*
- *series:time*
- *show:when*
- *slide:toggle*
- *social:comment-form*
- *social:comment-list*
- *styling*
- *table:body*
- *table:cell*
- *table:header*
- *table:header-cell*
- *table:result-list*
- *table:row*
- *task:form*

- *task:list*
- *toggle:controls*
- *toggle:pane*
- *toggle:panes*
- *tooltip*
- *track:clicks*
- *track:query*
- *trigger:event*
- *url:tabs*
- *widget:carousel*
- *widget:date-range*
- *widget:datetimepicker*
- *widget:file-hierarchy*
- *widget:icon*
- *widget:in-view*
- *widget:license*
- *widget:login-form*
- *widget:login-page*
- *widget:logout*
- *widget:modal*
- *widget:open-modal*
- *widget:popover*
- *widget:scroll-to*
- *widget:slide-down*
- *widget:slider*
- *widget:spelling-suggestions*
- *widget:spinner*
- *widget:tutorial*
- *App Studio Concepts*
 - *App Development*
 - *App Studio Enterprise*
 - *App Studio Enterprise Downloads*
 - *Get Started with ASE*
 - *Appkit*
 - *Appkit Concepts*
 - *Appkit tags*
 - *Architecture and Technology Stack*
 - *Data Sources*
 - *Routing and URL Rules*
 - *Signals in Appkit*
 - *Social and Collaboration Tools*
 - *User Interface Components*
 - *Configuration*
 - *Basic Configuration*
 - *Configuration Files*
 - *Externalising Configuration*
 - *Overlaying Configuration Values*
 - *Structure of a Web Application*
 - *Navigation*
 - *Breadcrumbs tag*
 - *Faceted Navigation*
 - *Tags tag*
 - *Rendering Search Results*
 - *Results Table*
 - *Cell tag*
 - *Header Cell tag*
 - *Header tag*
 - *Row tag*
 - *Table Body tag*
 - *Table Result List tag*
 - *Result List tag*
 - *Result tags*
 - *Extending Appkit*
 - *Collaboration Events*
 - *Developing Platforms*
 - *Developing Processors*
 - *Developing Services and Servlets*

- *Layout and Styling*
 - *Layout of the Default View*
 - *Themes and Styling*
 - *Topic Pages*
 - *Fusion App Studio Release Notes*
 - *App Studio 4.4.0 Release Notes*
 - *App Studio 4.2.1 Release Notes*
 - *App Studio 4.2.0 Release Notes*
 - *App Studio 4.1.2 Release Notes*
 - *App Studio 4.1.1 Release Notes*
 - *App Studio 4.1.0 Release Notes*
 - *App Studio 4.0.0 Release Notes*
 - *Appkit SDK Release Notes*
 - *Appkit SDK 4.14.0 Release Notes*
 - *Appkit SDK 4.13.0 Release Notes*
 - *Appkit SDK 4.11.0 Release Notes*
 - *Appkit SDK 4.10.0 Release Notes*
 - *Appkit SDK 4.9.0 Release Notes*
 - *Appkit SDK 4.8.0 Release Notes*
 - *Appkit SDK 4.7.0 Release Notes*
 - *Appkit SDK 4.6.0 Release Notes*
 - *Appkit SDK 4.5.0 Release Notes*
 - *Appkit SDK 4.4.0 Release Notes*
 - *Appkit SDK 4.3.0 Release Notes*
 - *Appkit SDK 4.2.5 Release Notes*
 - *Appkit SDK 4.2.4 Release Notes*
 - *Appkit SDK 4.2.3 Release Notes*
 - *Appkit SDK 4.2.2 Release Notes*
 - *Appkit SDK 4.2.1 Release Notes*
 - *Appkit SDK 4.2.0 Release Notes*
 - *Fusion App Studio*
-

About Legacy Docs

This site includes legacy documentation for product versions that are no longer supported, according to our Lucidworks Fusion Product Lifecycle (<https://doc.lucidworks.com/policies/7shln5/lucidworks-version-support-lifecycle>).

Legacy documentation is not updated or maintained. As a result, you may discover mistakes or inaccuracies within the documentation.

Recent migrations

Product	End of support	Migrated to legacy
Fusion 5.6.x	February 10, 2023	October 21, 2024
Fusion 5.7.x	June 13, 2023	October 21, 2024
Fusion 5.8.x	September 22, 2023	October 21, 2024
Managed Fusion 5.6.x	February 10, 2023	Managed Fusion documentation is not migrated.
Managed Fusion 5.7.x	June 13, 2023	Managed Fusion documentation is not migrated.
Managed Fusion 5.8.x	September 22, 2023	Managed Fusion documentation is not migrated.

Frequently asked questions

Why did you migrate legacy docs to a new site?

A new site is the best way to provide a high quality user experience. The legacy documentation site is based on this site and will continue to receive feature updates, as appropriate.

What about my saved pages?

We understand that you may have saved pages in bookmarks, shared documents, and other places. To help minimize the impact of this migration, we added page redirects from the current documentation site. However, we recommend that you update your saved pages as soon as possible.

Will more products be affected in the future?

Yes, we plan to migrate documentation for product versions that reach their end of life support to the legacy documentation site in the future. This will take place on or after the date stated in the Lucidworks Fusion Product Lifecycle (<https://doc.lucidworks.com/policies/7shln5/lucidworks-version-support-lifecycle>) policy.

App Studio Reference Guides

The reference section contains information that is presented without explanation of the general concepts. It includes content such as definitions and attributes for services, tags, modules, etc.

For conceptual information, see the *concepts section*.

- *Accessing Data*
The articles in this section describe key aspects of accessing data.
- *Deployment Overview*
This article describes general app deployment with App Studio.
- *FAQ*
The articles in this section answer frequently asked questions about various aspects and functionality of App Studio.
- *Filters*
Filters are AngularJS based filters that can be implemented in the markup by piping the filter name to a data object, which transforms the object. See *Filter Syntax* for more information.
- *Gallery*
The gallery provides visual examples of App Studio elements, with code snippets that you can copy, paste, and edit in your own App Studio views.
- *Installation*
This section contains various articles on the installation, system requirements, and supported environments of App Studio.
- *Licensing*
This article contains basic licensing details.
- *Modules*
Appkit's suite of search components is modular, which means you are able to get a working solution into place quickly using tried-and-tested components.
- *Services Reference*
App Studio exposes a number of frontend JavaScript services which encapsulate reusable processes. These services are based on the AngularJS framework. There are also backend Java web services that perform the same functions as many of the frontend services, including platform search endpoints, platform series endpoint, query suggestion/typeahead services, csv/excel export, and facet search.
- *Tags*
The App Studio markup language defines a set of HTML-like elements, or *tags*, for defining search queries to the backend and visualizing the search results and facets, handling all the complexities of connecting to Fusion.

Services

App Studio exposes a number of frontend JavaScript services which encapsulate reusable processes. These services are based on the AngularJS framework. There are also backend Java web services that perform the same functions as many of the frontend services, including platform search endpoints, platform series endpoint, query suggestion/typeahead services, csv/excel export, and facet search.

\$filterService

Description

Utility for parsing and creating filter objects.

Methods

- **parse(filterString)**

Method to parse URL representations of a filter.

Parameters

Param	Type	Details
filterString	string	filter to parse.

Example

```
$filterService.parse('foo["bar"]');
```

\$lightningLog

Description

Twigkit factory for logging.

Methods

- **debug(file, message)**

Method to debug

Parameters

Param	Type	Details
file	string	The file the message is coming from
message	string	The message to be displayed

Example

```
$lightningLog.debug(instance,message);
```

- **error(file, message)**

Method to return error message

Parameters

Param	Type	Details
file	string	The file the message is coming from
message	string	The message to be displayed

Example

```
$lightningLog.error(instance,message);
```

- **warn(file, message)**

Method to return warn message

Parameters

Param	Type	Details
file	string	The file the message is coming from
message	string	The message to be displayed

Example

```
$lightningLog.warn(instance,message);
```

\$lightningUrl

Description

Twigkit factory for getting and setting the location (url).

Methods

- **addFilterObjectToUrl(filter, reloadSearch)**

Method to add a filter to the url

Parameters

Param	Type	Details
filter	Object	The filter object to add to the url.
reloadSearch (<i>optional</i>)	boolean	Whether to reload search results.

Example

```
$lightningUrl.addFilterObjectToUrl(filter);
```

- **addFilterToLocationCopy(fieldName, actual, display, orFilter, location, exclude, match)**

- **addFilterToUrl(fieldName, actual, display, reloadSearch, orFilter, exclude, match)**

Method to add a filter to the url

Parameters

Param	Type	Details
fieldName	string	The name of the filter to apply.
actual	string	The actual value of the filter.
display	string	The display value of the filter.
reloadSearch (<i>optional</i>)	boolean	Whether to reload search results.
orFilter (<i>optional</i>)	string	If the filter is to be ORed
exclude (<i>optional</i>)	boolean	If the filter is to be exclude values
match (<i>optional</i>)	string	'left_partial' or 'partial' or 'right_partial' or 'exact'

Example

```
$lightningUrl.addFilterToUrl(fieldName, actual, display, reloadSearch);
```

- **addRangeFilterDisplayToUrl(fieldName, from, to, displayFrom, displayTo, reloadSearch, orFilter, fromExcluded, toExcluded, exclude)**

Method to add a filter to the url

Parameters

Param	Type	Details
fieldName	string	The name of the filter to apply.
from	string	The from value of the filter.
to	string	The to value of the filter.
displayFrom	string	The from display value of the filter.
displayTo	string	The to display value of the filter.
reloadSearch (<i>optional</i>)	boolean	Whether to reload search results.
orFilter (<i>optional</i>)	boolean	If the filter is to be ORed
fromExcluded (<i>optional</i>)	string	whether the from is to be excluded.
toExcluded (<i>optional</i>)	string	Whether the to is to be excluded.
exclude (<i>optional</i>)	boolean	Whether the values should be excluded.

Example

\$lightningUrl.addRangeFilterDisplayToUrl(fieldName, from, to, display, reloadSearch, orFilter, fromExcluded, toExcluded) ;

- **addRangeFilterToUrl(fieldName, from, to, displayFrom, displayTo, reloadSearch, orFilter, toLabel, fromExcluded, toExcluded, exclude)**

Method to add a filter to the url

Parameters

Param	Type	Details
fieldName	string	The name of the filter to apply.
from	string	The from value of the filter.
to	string	The to value of the filter.
displayFrom	string	The from display value of the filter.
displayTo	string	The to display value of the filter.
reloadSearch (<i>optional</i>)	boolean	Whether to reload search results.
orFilter (<i>optional</i>)	boolean	If the filter is to be ORed
toLabel (<i>optional</i>)	string	The label which appears between the from and to values.
fromExcluded (<i>optional</i>)	string	whether the from is to be excluded.
toExcluded (<i>optional</i>)	string	Whether the to is to be excluded.
exclude (<i>optional</i>)	boolean	Whether the values should be excluded.

Example

`$lightningUrl.addRangeFilterToUrl(fieldName, from, to, displayFrom, displayTo, reloadSearch, orFilter, toLabel, fromExcluded, toExcluded);`

- **createFilter(fieldName, actual, display, orFilter, exclude, match, type)**

Creates a filter for the URL

Parameters

Param	Type	Details
fieldName	string	The name of the filter to apply.
actual	string	The actual value of the filter.
display	string	The display value of the filter.
orFilter (<i>optional</i>)	string	If the filter is to be ORed (field or value)
exclude (<i>optional</i>)	boolean	If the filter is to be exclude values
match (<i>optional</i>)	string	'left_partial' or 'partial' or 'right_partial' or 'exact'
type (<i>optional</i>)	string	Explicitly set the type of the filter, 'string' or 'number'. A string filter value will be wrapped in double quotes.

Returns

string	Returns a filter that can be added to the URL

- **createFilterFromObject(filter)**

Creates a filter for the URL

Parameters

Param	Type	Details
filter	Object	The filter object or string to create a filter from.

Returns

string	Returns a filter that can be added to the URL

- **createRangeFilter(fieldName, from, to, displayFrom, displayTo, orFilter, toLabel, fromExcluded, toExcluded, exclude)**

Creates a range filter for the URL

Parameters

Param	Type	Details
fieldName	string	The name of the filter to apply.
from	string	The from value of the filter.
to	string	The to value of the filter.
displayFrom	string	The from display value of the filter.
displayTo	string	The to display value of the filter.
orFilter (optional)	string	If the filter is to be ORed (value field)
toLabel (optional)	string	The label which appears between the from and to values.
fromExcluded (optional)	string	whether the from is to be excluded.
toExcluded (optional)	string	Whether the to is to be excluded.
exclude (optional)	boolean	Whether the values should be excluded.

Returns

string	Returns a filter that can be added to the URL

- **createRangeFilterDisplay(filterName, from, to, display, orFilter, fromExcluded, toExcluded, exclude)**

Creates a range filter for the URL

Parameters

Param	Type	Details
filterName	string	The name of the filter to apply.
from	string	The from value of the filter.
to	string	The to value of the filter.
display	string	The display value of the filter.
orFilter (optional)	string	If the filter is to be ORed (value field)
fromExcluded (optional)	string	whether the from is to be excluded.
toExcluded (optional)	string	Whether the to is to be excluded.
exclude (optional)	boolean	Whether the values should be excluded.

Returns

string	Returns a filter that can be added to the URL

- **getAllUrlParameters()**

Method to get a copy of all URL parameters

Returns

object	Object literal of all URL parameters

Example

```
$lightningUrl.getAllUrlParameters();
```

- **getFiltersFromUrl(fieldName)**

Method to gets all of a set of filters from the url

Parameters

Param	Type	Details
fieldName	string	Only get filters on the specified field.

Returns

Array	Returns values of the filterName from URL.

Example

```
$lightningUrl.getFiltersFromUrl(filterName);
```

- **getUrlParameter(parameter, defaultValue)**

Method to get a url parameter

Parameters

Param	Type	Details
parameter	string	The parameter to get from the URL.
defaultValue (optional)	string	The value to return if parameter returns null.

Returns

var	Returns value of parameter from URL or defaultValue.

Example

```
$lightningUrl.getUrlParameter(parameter, defaultValue);
```

- **getUrlParametersAsArray(parameter, defaultValue)**

Method to gets an array of url parameters

Parameters

Param	Type	Details
parameter	string	The parameter to get from the URL.
defaultValue (optional)	string	The value to return if parameter returns null.

Returns

Array	Returns values of the parameter from URL or defaultValue in an array.

Example

```
$lightningUrl.getUrlParametersAsArray(parameter, defaultValue);
```

- **removeFiltersFromUrl(filterName, reloadSearch)**

Method to remove all of a set of filters from the url

Parameters

Param	Type	Details
filterName	string	The name of the filter to remove.
reloadSearch (optional)	boolean	Whether to reload search results.

Example

```
$lightningUrl.removeFiltersFromUrl(filterName, reloadSearch);
```

- **removeFiltersFromUrl(filterName, reloadSearch)**

Method to remove all of a set of filters from the url

Parameters

Param	Type	Details
filterName	string	The name of the filter to remove.
reloadSearch (optional)	boolean	Whether to reload search results.

Example

```
$lightningUrl.removeFiltersFromUrl(filterName, reloadSearch);
```

- **resetAllParameters(reloadSearch)**

Resets all URL Parameters

Parameters

Param	Type	Details
reloadSearch	boolean	Whether to reload the search

Returns

string	Returns a filter that can be added to the URL

- **setUrlParameter(parameter, value, reloadSearch)**

Method to set a url parameter

Parameters

Param	Type	Details
parameter	string	The parameter to add to the URL.
value	Arraystring	The value to assign to the parameter.
reloadSearch (optional)	boolean	Whether to reload the results.

Example

```
$lightningUrl.setUrlParameter(parameter, value, reloadSearch);
```

- **setUrlParameters(query, reloadSearch)**

Method to set a group of url parameters from a query.

Parameters

Param	Type	Details
query	Object	Query object from which URL Parameters are to be used.
reloadSearch (optional)	boolean	Whether to reload the search once all the parameters have been set.

Example

```
$lightningUrl.setUrlParameters(query);
```

- **urlContainsFilter(filterName, actual, display, orFilter, exclude, match)**

Method to set if filter is in the url

Parameters

Param	Type	Details
filterName	string	The name of the filter to apply.
actual	string	The actual value of the filter.
display	string	The display value of the filter.
orFilter (<i>optional</i>)	string	If the filter is to be ORed on <code>value</code> or <code>field</code>
exclude (<i>optional</i>)	boolean	If the filter is to be exclude values
match (<i>optional</i>)	string	'left_partial' or 'partial' or 'right_partial' or 'exact'

Returns

boolean	Whether the URL contains the filter.

- **urlContainsFilterFromObject(filter)**

Method to see if filter is in the url

Parameters

Param	Type	Details
filter	object	The filter object

Returns

boolean	Whether the URL contains the filter.

\$twigkit

Description

Twigkit factory with methods for accessing the Twigkit API.

Methods

- **cancelAllResponses()**

Fires an event to cancel the current response being requested from the backend.

Example

```
$twigkit.cancelAllResponses();
```

- **deleteBookmark(id)**

Sends a request to delete a bookmark to the API.

Parameters

Param	Type	Details
id	string	The id of the bookmark to delete.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.deleteBookmark(id);
```

- **deleteComment(platform, id)**

Sends a request to delete a comment to the API.

Parameters

Param	Type	Details
platform	Object	The platform object (retrievable from <i>Platform Service</i>).
id	string	The id of the comment to delete.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.deleteComment(platform, id);
```

- **deleteLike(platform, id)**

Sends a request to delete a like to the API.

Parameters

Param	Type	Details
platform	Object	The platform object (retrievable from <i>Platform Service</i>).
id	string	The id of the bookmark to delete.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.deleteLike(platform,id);
```

- **deleteSavedQuery(id)**

Sends a request to delete a saved query to the API.

Parameters

Param	Type	Details
id	string	The id of the saved query to delete.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.deleteSavedQuery(id);
```

- **deleteTopic(id)**

Sends a request to delete a topic to the API.

Parameters

Param	Type	Details
id	string	The id of the topic to delete.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.deleteTopic(id);
```

- **getComments(platform, target)**

Sends a request to get comments from the API.

Parameters

Param	Type	Details
platform	Object	The platform object (retrievable from <i>Platform Service</i>).
target	string	Returns all comments that apply to the specified target document.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
$twigkit.getComments(platform, target);
```

- **getConstant(defaultValue)**

Gets a constant declared to the lightning module.

A constant can be added to your application like so

```
angular .module('lightning').constant('contextPath', 'My Constant Value');
```

You can then get this value in your controller and apply it to the scope or rootScope (if applied to the rootScope you can use `{{ $root.myConstant }}` in your views.

```
angular.module('twigkitLightApp')  
controller('MainCtrl', ['$scope', '$stateParams','$twigkit', function ($scope,  
$stateParams,$twigkit) {
```

```
    $scope.params = $stateParams;
```

```

$scope.test123 = $twigkit.getConstant(&#39;test&#39;,&#39;No 123 Found Here&#39;);
  $rootScope.test = $twigkit.getConstant(&#39;test&#39;,&#39;&#39;);
  });
</code></pre></div><h5 id="methods_getconstant_parameters">Parameters</h5>
<table class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>defaultValue</td><td><a href="" class="label type-hint type-hint-string">string</a></td><td><div class="lightning-services-twigkit-getconstant-page"><p>Value to be returned if context path does not exist.</p>
</div></td></tr></tbody></table><h5
id="methods_getconstant_returns">Returns</h5><table class="variables-matrix">
<tr><td><a href="" class="label type-hint type-hint-string">string</a></td><td>
<div class="lightning-services-twigkit-getconstant-page"><p>Returns the
constant value.</p>
</div></td></tr></table><h4 id="methods_getconstant_example">Example</h4>
<div class="example"><div class="lightning-services-twigkit-getconstant-page">
<p>$twigkit.getConstant(defaultValue);</p>
</div></div>
</div>
</li>
<li><h3 id="methods_getcontextpath">getContextPath(defaultValue)</h3>
<div class="getContextpath"><div class="lightning-services-twigkit-
getContextpath-page"><p>Gets the context path if it is available.</p>
</div><h5 id="methods_getcontextpath_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>defaultValue</td><td><div><em>(optional)</em></div></td><td><a href="" class="label
type-hint type-hint-string">string</a></td><td><div class="lightning-services-
twigkit-getcontextpath-page"><p>Value to be returned if context path does not
exist. Defaults to &#39;/&#39;;</p>
</div></td></tr></tbody></table><h5
id="methods_getcontextpath_returns">Returns</h5><table class="variables-
matrix"><tr><td><a href="" class="label type-hint type-hint-string">string</a>
</td><td><div class="lightning-services-twigkit-getcontextpath-page"><p>Returns
a string of the context path or the default value.</p>
</div></td></tr></table><h4 id="methods_getcontextpath_example">Example</h4>
<div class="example"><div class="lightning-services-twigkit-getcontextpath-
page"><p>$twigkit.getContextPath(defaultValue);</p>
</div></div>
</div>
</li>
<li><h3 id="methods_getgraphfacet">getGraphFacet(query, platform, source-field,
target-field, measure, measure-field)</h3>
<div class="getgraphfacet"><div class="lightning-services-twigkit-
getgraphfacet-page"><p>Gets graph data for a facet</p>
</div><h5 id="methods_getgraphfacet_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>query</td><td><a href="" class="label type-hint type-hint-
string">string</a></td><td><div class="lightning-services-twigkit-
getgraphfacet-page"><p>The query to use against the platform.</p>
</div></td></tr><tr><td>platform</td><td><a href="" class="label type-hint
type-hint-string">string</a></td><td><div class="lightning-services-twigkit-
getgraphfacet-page"><p>The name of the platform to use.</p>
</div></td></tr><tr><td>source-field</td><td><a href="" class="label type-hint
type-hint-string">string</a></td><td><div class="lightning-services-twigkit-
getgraphfacet-page"><p>Name of a primary field to facet on (for example
&quot;Country&quot;). The values of this facet will define the set of graph
nodes with outgoing edges.</p>
</div></td></tr><tr><td>target-field</td><td><a href="" class="label type-hint
type-hint-string">string</a></td><td><div class="lightning-services-twigkit-
getgraphfacet-page"><p>Name of a secondary field to facet on (for example
&quot;City&quot;), for every facet value of the primary field.
The values of this facet will define the set of graph nodes with incoming
edges. For example, say the primary facet

```

```

(source-field) has values for Country=Germany and Country=France and the
secondary facet (target-field) has values for City=Berlin,
City=Munich and City=Paris. Then we add graph edges Germany--&gt;Berlin,
Germany--&gt;Munich and France--&gt;Paris.</p>
</div></td></tr><tr><td>measure <div><em>(optional)</em></div></td><td><a
href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-services-twigkit-getgraphfacet-page"><p>Name of a numerical
measure that defines the weight of each graph edge. When omitted, this defaults
to the counts of the facets defined
by the target-field. Some possible measures include &quot;sum&quot;,
&quot;mean&quot; and &quot;min&quot;;. (Only applicable if not using
<code>data</code> attribute).</p>
</div></td></tr><tr><td>measure-field <div><em>(optional)</em></div></td><td><a
href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-services-twigkit-getgraphfacet-page"><p>Field to measure, when
the &quot;measure&quot; option is specified.</p>
</div></td></tr></tbody></table><h5
id="methods_getgraphfacet_returns">Returns</h5><table class="variables-matrix">
<tr><td><a href="" class="label type-hint type-hint-
httppromise">HttpPromise</a></td><td><div class="lightning-services-twigkit-
getgraphfacet-page"><p>Returns a promise which will resolve depending on the
response from the API.</p>
</div></td></tr></table><h4 id="methods_getgraphfacet_example">Example</h4>
<div class="example"><div class="lightning-services-twigkit-getgraphfacet-
page"><p>$twigkit.getGraphFacet(&#39;query&#39;, &#39;platform&#39;,
&#39;facet&#39;);</p>
</div></div>
</div>
</li>
<li><h3 id="methods_getlikes">getLikes(options)</h3>
<div class="getlikes"><div class="lightning-services-twigkit-getlikes-page">
<p>Sends a request to get a likes from the API.</p>
</div><h5 id="methods_getlikes_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>options</td><td><a href="" class="label type-hint type-hint-
object">Object</a></td><td><div class="lightning-services-twigkit-getlikes-
page"><p>Object containing options to be sent in the get request. Like target,
collection, topic etc.</p>
</div></td></tr></tbody></table><h5 id="methods_getlikes_returns">Returns</h5>
<table class="variables-matrix"><tr><td><a href="" class="label type-hint type-
hint-httpromise">HttpPromise</a></td><td><div class="lightning-services-
twigkit-getlikes-page"><p>Returns a promise which will resolve depending on the
response from the API.</p>
</div></td></tr></table><h4 id="methods_getlikes_example">Example</h4>
<div class="example"><div class="lightning-services-twigkit-getlikes-page">
<p>$twigkit.getLikes(options);</p>
</div></div>
</div>
</li>
<li><h3 id="methods_getuser">getUser()</h3>
<div class="getuser"><div class="lightning-services-twigkit-getuser-page">
<p>Calls the Twigkit API to return the current user</p>
<p>To enable this you need to make sure that your security.conf file contains
the line <code>webservice-enabled: true</code></p>
<p>Your security.conf file can be found in
&#39;resources/conf/security/security.conf&#39;.</p>
<p>For spring security you will need the following rule to allow access to the
user api, if parts of the site are public.</p>
<p><code>&lt;intercept-url pattern=&quot;/twigkit/api/user&quot;
access=&quot;isAnonymous() or isAuthenticated()&quot;;&gt;</code></p>
</div><h5 id="methods_getuser_returns">Returns</h5><table class="variables-
matrix"><tr><td><a href="" class="label type-hint type-hint-
httpromise">HttpPromise</a></td><td><div class="lightning-services-twigkit-
getuser-page"><p>Returns a promise which will resolve depending on the response
from the API.</p>
</div></td></tr></table><h4 id="methods_getuser_example">Example</h4>

```

```

<div class="example"><div class="lightning-services-twigkit-getuser-page">
<p>$twigkit.getUser();</p>
</div></div>
</li>
<li><h3 id="methods_onscancelallresponses">onCancelAllResponses(callback, scope)
</h3>
<div class="onscancelallresponses"><div class="lightning-services-twigkit-
onscancelallresponses-page"><p>When the cancel response event is fired the
callback function will be called.</p>
</div><h5 id="methods_onscancelallresponses_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>callback</td><td><a href="" class="label type-hint type-hint-
function">Function</a></td><td><div class="lightning-services-twigkit-
onscancelallresponses-page"><p>The function to be called when the cancel event
is fired.</p>
</div></td></tr><tr><td>scope</td><td><a href="" class="label type-hint type-
hint-object">Object</a></td><td><div class="lightning-services-twigkit-
onscancelallresponses-page"><p>The <code>$scope</code> object for unsubscribing
to the event when the scope is destroyed.</p>
</div></td></tr></tbody></table><h4
id="methods_onscancelallresponses_example">Example</h4>
<div class="example"><div class="lightning-services-twigkit-
onscancelallresponses-page"><pre><code>$twigkit.onCancelAllResponses(function ()
{
  console.log(&#39;Responses have been cancelled&#39;);
}, $scope);


```

```

$twigkit.cancelAllResponses(); </code></pre></div></div> </li> <li><h3
id="methods_postlike">postLike(platform, id, target, topic, collection)</h3> <div
class="postlike"><div class="lightning-services-twigkit-postlike-page"><p>Sends a
request to create a like to the API.</p> </div><h5
id="methods_postlike_parameters">Parameters</h5><table class="variables-matrix
table table-bordered table-striped"><thead><tr><th>Param</th><th>Type</th>
<th>Details</th></tr></thead><tbody><tr><td>platform</td><td><a href=""
class="label type-hint type-hint-object">Object</a></td><td><div class="lightning-
services-twigkit-postlike-page"><p>The platform object (retrievable from <a
href="/api/lightning.PlatformService">Platform Service</a>).</p> </div></td></tr>
<tr><td>id</td><td><a href="" class="label type-hint type-hint-string">string</a>
</td><td><div class="lightning-services-twigkit-postlike-page"><p>The id of the
bookmark to delete.</p> </div></td></tr><tr><td>target</td><td><a href=""
class="label type-hint type-hint-string">string</a></td><td><div class="lightning-
services-twigkit-postlike-page"><p>A unique identifier for a document that is
being bookmarked. Example values: document123 Maximum length: 2083 characters</p>
</div></td></tr><tr><td>topic <div><em>(optional)</em></div></td><td><a href=""
class="label type-hint type-hint-number">number</a></td><td><div class="lightning-
services-twigkit-postlike-page"><p>A topic to which the new bookmark should
belong.</p> </div></td></tr><tr><td>collection <div><em>(optional)</em></div></td>
<td><a href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-services-twigkit-postlike-page"><p>The collection to which the
target document belongs.</p> </div></td></tr></tbody></table><h5
id="methods_postlike_returns">Returns</h5><table class="variables-matrix"><tr><td>
<a href="" class="label type-hint type-hint-httpromise">HttpPromise</a></td><td>
<div class="lightning-services-twigkit-postlike-page"><p>Returns a promise which
will resolve depending on the response from the API.</p> </div></td></tr></table>
<h4 id="methods_postlike_example">Example</h4> <div class="example"><div
class="lightning-services-twigkit-postlike-page"><p>$twigkit.postLike(platform,
id, target, topic, collection);</p> </div></div> </div> </li> <li><h3
id="methods_search">search(searchQuery, platform, params, ignore-loading-bar,
canceller)</h3> <div class="search"><div class="lightning-services-twigkit-search-
page"><p>Search method sends a query to a Twigkit Platform endpoint and gets a
result.</p> </div><h5 id="methods_search_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>

```

```

<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>searchQuery</td><td><a href="" class="label type-hint type-hint-
string">string</a></td><td><div class="lightning-services-twigkit-search-page">
<p>The search term to search against the results.</p> </div></td></tr><tr>
<td>platform</td><td><a href="" class="label type-hint type-hint-
object">Object</a></td><td><div class="lightning-services-twigkit-search-page">
<p>The platform object (retrievable from <a
href="/api/lightning.PlatformService">Platform Service</a>).</p> </div></td></tr>
<tr><td>params <div><em>(optional)</em></div></td><td><a href="" class="label
type-hint type-hint-object">Object</a></td><td><div class="lightning-services-
twigkit-search-page"><p>Other parameters to pass to the API.</p> </div></td></tr>
<tr><td>ignore-loading-bar <div><em>(optional)</em></div></td><td><a href=""
class="label type-hint type-hint-boolean">boolean</a></td><td><div
class="lightning-services-twigkit-search-page"><p>Whether the loading bar across
the top of the page should be included for this http request. Default:true</p>
</div></td></tr><tr><td>canceller <div><em>(optional)</em></div></td><td><a
href="" class="label type-hint type-hint-promise">Promise</a></td><td><div
class="lightning-services-twigkit-search-page"><p>A promise object
(<code>$q.defer()</code>) used to cancel the request by using
<code>canceller.promise()</code></p> </div></td></tr></tbody></table><h5
id="methods_search_returns">Returns</h5><table class="variables-matrix"><tr><td><a
href="" class="label type-hint type-hint-httpromise">HttpPromise</a></td><td><div
class="lightning-services-twigkit-search-page"><p>Returns a promise which will
resolve depending on the response from the API.</p> </div></td></tr></table><h4
id="methods_search_example">Example</h4> <div class="example"><div
class="lightning-services-twigkit-search-page"><p>$twigkit.search(query, platform,
params);</p> </div></div> </div> </li> <li><h3
id="methods_searchfacet">searchFacet(facet, platform, query, rpp, facetQuery,
removeFilters, facetField)</h3> <div class="searchfacet"><div class="lightning-
services-twigkit-searchfacet-page"><p>Calls the Twigkit API to return filters from
a specific facet based on the searchQuery.</p> </div><h5
id="methods_searchfacet_parameters">Parameters</h5><table class="variables-matrix
table table-bordered table-striped"><thead><tr><th>Param</th><th>Type</th>
<th>Details</th></tr></thead><tbody><tr><td>facet</td><td><a href="" class="label
type-hint type-hint-string">string</a></td><td><div class="lightning-services-
twigkit-searchfacet-page"><p>The name of the facet to search.</p> </div></td></tr>
<tr><td>platform</td><td><a href="" class="label type-hint type-hint-
object">Object</a><a href="" class="label type-hint type-hint-string">string</a>
</td><td><div class="lightning-services-twigkit-searchfacet-page"><p>The platform
object (retrievable from <a href="/api/lightning.PlatformService">Platform
Service</a>) or the name of the platform.</p> </div></td></tr><tr><td>query</td>
<td><a href="" class="label type-hint type-hint-object">Object</a></td><td><div
class="lightning-services-twigkit-searchfacet-page"><p>The query object.</p>
</div></td></tr><tr><td>rpp</td><td><a href="" class="label type-hint type-hint-
number">Number</a></td><td><div class="lightning-services-twigkit-searchfacet-
page"><p>The number of filters to be returned by the API.</p> </div></td></tr><tr>
<td>facetQuery</td><td><a href="" class="label type-hint type-hint-
string">string</a></td><td><div class="lightning-services-twigkit-searchfacet-
page"><p>The facet query to use against the API end point.</p> </div></td></tr>
<tr><td>removeFilters <div><em>(optional)</em></div></td><td><a href=""
class="label type-hint type-hint-boolean">boolean</a></td><td><div
class="lightning-services-twigkit-searchfacet-page"><p>Whether to remove the
current filters for the facet when performing the search.</p> </div></td></tr><tr>
<td>facetField <div><em>(optional)</em></div></td><td><a href="" class="label
type-hint type-hint-string">string</a></td><td><div class="lightning-services-
twigkit-searchfacet-page"><p>If provided this field may be used to create a filter
on the facet which may not be the same as the facet name itself.</p> </div></td>
</tr></tbody></table><h5 id="methods_searchfacet_returns">Returns</h5><table
class="variables-matrix"><tr><td><a href="" class="label type-hint type-hint-
httpromise">HttpPromise</a></td><td><div class="lightning-services-twigkit-
searchfacet-page"><p>Returns a promise which will resolve depending on the
response from the API.</p> </div></td></tr></table><h4

```

```

id="methods_searchfacet_example">Example</h4> <div class="example"><div
class="lightning-services-twigkit-searchfacet-page">
<p>$twigkit.searchFacet(facet,platform,query,facetQuery);</p> </div></div> </div>
</li> <li><h3 id="methods_searchseriesfacet">searchSeriesFacet(query, platform,
facet, measure, rpp)</h3> <div class="searchseriesfacet"><div class="lightning-
services-twigkit-searchseriesfacet-page"><p>Gets series data for a facet.</p>
</div><h5 id="methods_searchseriesfacet_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr><td>query</td>
<td><a href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-services-twigkit-searchseriesfacet-page"><p>The query to use
against the platform.</p> </div></td></tr><tr><td>platform</td><td><a href=""
class="label type-hint type-hint-object">Object</a></td><td><div class="lightning-
services-twigkit-searchseriesfacet-page"><p>The platform object (retrievable from
<a href="/api/lightning.PlatformService">Platform Service</a>).</p> </div></td>
</tr><tr><td>facet</td><td><a href="" class="label type-hint type-hint-
object">Object</a></td><td><div class="lightning-services-twigkit-
searchseriesfacet-page"><p>The facet to search upon.</p> </div></td></tr><tr>
<td>measure</td><td><a href="" class="label type-hint type-hint-string">string</a>
</td><td><div class="lightning-services-twigkit-searchseriesfacet-page"><p>What
fact to use as an alternative to count as quantitative value for a filter.</p>
</div></td></tr><tr><td>rpp</td><td><a href="" class="label type-hint type-hint-
number">Number</a></td><td><div class="lightning-services-twigkit-
searchseriesfacet-page"><p>The number of results to fetch.</p> </div></td></tr>
</tbody></table><h5 id="methods_searchseriesfacet_returns">Returns</h5><table
class="variables-matrix"><tr><td><a href="" class="label type-hint type-hint-
httppromise">HttpPromise</a></td><td><div class="lightning-services-twigkit-
searchseriesfacet-page"><p>Returns a promise which will resolve depending on the
response from the API.</p> </div></td></tr></table><h4
id="methods_searchseriesfacet_example">Example</h4> <div class="example"><div
class="lightning-services-twigkit-searchseriesfacet-page">
<p>$twigkit.searchSeriesFacet(query, platform, facet);</p> </div></div> </div>
</li> <li><h3 id="methods_searchseriesfields">searchSeriesFields(query, platform,
fields, rpp)</h3> <div class="searchseriesfields"><div class="lightning-services-
twigkit-searchseriesfields-page"><p>Gets series data for a result list.</p> </div>
<h5 id="methods_searchseriesfields_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr><td>query</td>
<td><a href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-services-twigkit-searchseriesfields-page"><p>The query to use
against the platform.</p> </div></td></tr><tr><td>platform</td><td><a href=""
class="label type-hint type-hint-object">Object</a></td><td><div class="lightning-
services-twigkit-searchseriesfields-page"><p>The platform object (retrievable from
<a href="/api/lightning.PlatformService">Platform Service</a>).</p> </div></td>
</tr><tr><td>fields</td><td><a href="" class="label type-hint type-hint-
object">Object</a></td><td><div class="lightning-services-twigkit-
searchseriesfields-page"><p>The facet to fields upon.</p> </div></td></tr><tr>
<td>rpp</td><td><a href="" class="label type-hint type-hint-number">Number</a>
</td><td><div class="lightning-services-twigkit-searchseriesfields-page"><p>The
number of results to fetch.</p> </div></td></tr></tbody></table><h5
id="methods_searchseriesfields_returns">Returns</h5><table class="variables-
matrix"><tr><td><a href="" class="label type-hint type-hint-
httppromise">HttpPromise</a></td><td><div class="lightning-services-twigkit-
searchseriesfields-page"><p>Returns a promise which will resolve depending on the
response from the API.</p> </div></td></tr></table><h4
id="methods_searchseriesfields_example">Example</h4> <div class="example"><div
class="lightning-services-twigkit-searchseriesfields-page">
<p>$twigkit.searchSeriesFacet(query, platform, fields);</p> </div></div> </div>
</li> </ul> </div> </div>

```


\$util

Description

Twigkit factory with utility functions for Development.

Methods

- **hasChildren(transclude)**

Method to check if the element has children.

Parameters

Param	Type	Details
transclude	Object	The transclude function from the elements post or pre linking function.

Returns

boolean	True if element has children.

Example

```
$util.hasChildren(transclude);
```

BookmarkService

Description

Twigkit service storing bookmarks over a session

CollaborationService

The Appkit Collaboration API access service is only available in Appkit 4.1+.

Description

Appkit service for accessing Appkit Collaboration API's

Methods

- **delete(annotation, platform, id)**

Deletes an annotation

Parameters

Param	Type	Details
annotation	string	The type of collaboration annotation to delete. Valid values are:- comment – For deleting a comment. bookmark – For deleting a bookmark. topic – For deleting a topic. link – For deleting a like. saved-query – For deleting a saved-query.
platform	string	The name of the platform.
id	string	The ID of the annotation to be deleted.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
CollaborationService.delete('comment', 'platform', 123456);
```

- **getCommentAddedEventName(target, topic, collection)**

Method for returning the event emitted when a comment is added

Parameters

Param	Type	Details
target	string	The comment target.
topic (optional)	string	The topic associated to the comment.
collection (optional)	string	The collection associated to the comment.

Returns

string	The event name that will be emitted.

Example

```
CollaborationService.getCommentAddedEventName(target, collection, topic);
```

- **post(annotation, platform, data)**

Creates an annotation

Parameters

Param	Type	Details
annotation	string	The type of collaboration annotation to delete. Valid values are:- comment – For creating a comment. bookmark – For creating a bookmark. topic – For creating a topic. link – For creating a like. saved-query – For creating a saved-query.
platform	string	The name of the platform.
data	object	The data object for creating the required annotation.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
CollaborationService.post('comment', 'platform', data);
```

- **query(annotation, platform, target, collection, topic, results)**

Gets an array of Collaboration Annotations (comments, likes, topics or bookmarks) based on the options provided.

Parameters

Param	Type	Details
annotation	string	The type of collaboration annotation you are searching for. Valid values are:- comments – Returns a list of comments bookmarks – Returns a list of bookmarks topics – Returns a list of topics links – Returns a list of likes. saved-queries – Returns a list of saved-queries.
platform	string	The name of the platform.
target (optional)	string	When provided it will return only annotations associated with this target (not applicable to topics).
collection (optional)	string	When provided it will return only annotations associated with this collection (not applicable to topics).
topic (optional)	string	When provided it will return only annotations associated with this topic (not applicable to topics).
results (optional)	number	The number of annotations to return.

Returns

HttpPromise	Returns a promise which will resolve depending on the response from the API.

Example

```
CollaborationService.query('comments', 'platform', 'foo-bar');
```

DateUtils

Description

Twigkit service for useful date functions

Methods

- **dateSortResults(results, dateField, oldestFirst)**

Sort results array by date

Parameters

Param	Type	Details
results	mixed	Array of results from response
dateField	string	Result field to use for ordering
oldestFirst	boolean	If true the results will be order from oldest to newest

Returns

mixed	Sorted array of results

- **groupByDateRanges(results, dateField, groupBy)**

Group results into date ranges, used for the timeline

Parameters

Param	Type	Details
results	mixed	Sorted array of results
dateField	string	Result field to use for ordering
groupBy	string	Optionally specify how to group the results, by default it tries to get the best fit

Returns

mixed	Array of grouped results

FieldService

Description

Twigkit service for getting the values from a field

Methods

- **getDisplayValue(field)**

Gets the correct property from the value object

Parameters

Param	Type	Details
field	mixed	The field to find a display value in.

- **getValues(field)**

Gets an array of all values for the field

Parameters

Param	Type	Details
field	mixed	The field to find all the values in

ModalService

Description

Twigkit service for registering and un-registering modals

Methods

- **close(name)**

Hides the modal in the view.

Parameters

Param	Type	Details
name	string	The ID of the modal to remove.

Example

```
ModalService.close(name);
```

- **onClose(name, callback, scope)**

When the modal's close event is fired the callback function will be called.

Parameters

Param	Type	Details
name	string	The name of the modal, if null the callback will fire whenever a modal is closed
callback	Function	The function to be called when the cancel event is fired.
scope	Object	The <code>\$scope</code> object for unsubscribing to the event when the scope is destroyed.

Example

```
ModalService.onClose('MyModal', function (modalId) { console.log('Modal: 'modalId
+ ' has been closed'); }, $scope);
ModalService.close('my_modal'); </code></pre></div></div> </li> <li><h3
id="methods_register">register(modal)</h3> <div class="register"><div
class="lightning-modal-service-register-page"><p>Registers the modal to the
factory</p> </div><h5 id="methods_register_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr><td>modal</td>
<td><a href="" class="label type-hint type-hint-string">string</a></td><td><div
class="lightning-modal-service-register-page"><p>The modal object.</p> </div></td>
</tr></tbody></table><h4 id="methods_register_example">Example</h4> <div
class="example"><div class="lightning-modal-service-register-page">
<p>ModalService.register(modal);</p> </div></div> </div> </li> <li><h3
id="methods_show">show(name)</h3> <div class="show"><div class="lightning-
modal-service-show-page"><p>Shows the modal in the view.</p> </div><h5
id="methods_show_parameters">Parameters</h5><table class="variables-matrix table
table-bordered table-striped"><thead><tr><th>Param</th><th>Type</th>
<th>Details</th></tr></thead><tbody><tr><td>name</td><td><a href="" class="label
type-hint type-hint-string">string</a></td><td><div class="lightning-modal-service-
show-page"><p>The ID of the modal to remove.</p> </div></td></tr></tbody></table>
<h4 id="methods_show_example">Example</h4> <div class="example"><div
class="lightning-modal-service-show-page"><p>ModalService.show(name);</p> </div>
</div> </div> </li> <li><h3 id="methods_unregister">unregister(name)</h3> <div
class="unregister"><div class="lightning-modal-service-unregister-page"><p>Un-
registers the modal removing it from the view and the factory.</p> </div><h5
id="methods_unregister_parameters">Parameters</h5><table class="variables-matrix
table table-bordered table-striped"><thead><tr><th>Param</th><th>Type</th>
<th>Details</th></tr></thead><tbody><tr><td>name</td><td><a href="" class="label
type-hint type-hint-string">string</a></td><td><div class="lightning-modal-service-
unregister-page"><p>The ID of the modal to remove.</p> </div></td></tr></tbody>
</table><h4 id="methods_unregister_example">Example</h4> <div class="example"><div
class="lightning-modal-service-unregister-page"><p>ModalService.unregister(name);
</p> </div></div> </div> </li> </ul> </div> </div>
```

PlatformService

Description

Twigkit service for setting and getting Platforms.

QueryIdService

Description

Twigkit service for setting and getting query references for tracking.

QueryService

Description

Twigkit service for queries

Methods

- **clearQueries()**

Will remove all the queries from the factory.

Example

```
QueryService.clearQueries();
```

- **convertHttpParamsToQuery(httpParams)**

Converts http params to query value..

Parameters

Param	Type	Details
httpParams	object	The httpParams of the query.

Example

```
QueryService.convertHttpParamsToQuery(value);
```

- **convertHttpParamsToUrl(httpParams)**

Converts http params into a URL.

Parameters

Param	Type	Details
httpParams	object	A query object as http params

Example

```
QueryService.convertHttpParamsToUrl(query, true);
```

- **convertQueryToHttpParams(query, excludeHiddenFilters)**

Converts an object based query to HTTP Params.

Parameters

Param	Type	Details
query	object	Query to convert
excludeHiddenFilters (optional)	boolean	Whether to include hidden filters in the params.

Returns

object	Returns the query object as http params.

Example

```
QueryService.convertQueryToHttpParams(query);
```

- **convertQueryToUrl(query, includeHiddenFilters)**

Converts query object into a URL, encoding necessary values correctly and optionally removing hidden filters

Parameters

Param	Type	Details
query	object	The query object.
includeHiddenFilters	boolean	Whether to include hidden filters in the URL

Example

```
QueryService.convertQueryToUrl(query, true);
```

- **convertSortObjectToParam()**

Will convert sort object into an parameter.

Example

```
QueryService.convertSortObjectToParam(obj);
```

- **convertSortParamToObject()**

Will convert sort parameter into an object.

Example

```
QueryService.convertSortParamToObject(param);
```

- **getQuery(name)**

Gets the value of the query from the factory. If no query is found, the factory will return null.

Parameters

Param	Type	Details
name	string	Name of the query to retrieve.

Returns

object	Returns the query object, if no query object exists it will return null.

Example

```
QueryService.getQuery(name);
```

- **getQueryAsHttpParams(name, excludeHiddenFilters)**

Gets the value of the query from the factory. If no query is found, the factory will return null.

Parameters

Param	Type	Details
name	string	Name of the query to retrieve.
excludeHiddenFilters (<i>optional</i>)	boolean	Whether to include hidden filters in the params.

Returns

object	Returns the query object, if no query object exists it will return null.

Example

```
QueryService.getQueryAsHttpParams(name);
```

- **queryContainsFilterForField(query, fieldName, filter)**

Will identify if the query contains the filter.

Parameters

Param	Type	Details
query	object	The query object.
fieldName	string	The name of the field
filter	object	The filter object

Example

```
QueryService.queryContainsFilterForField(query, fieldName, filter);
```

- **removeFiltersForField(query, fieldName)**

Will remove any filters relating to the field name

Parameters

Param	Type	Details
query	object	The query object.
fieldName	string	The name of the field

Example

```
QueryService.removeFiltersForField(query, fieldName);
```

- **setQuery(name, value)**

Adds the query value to the factory.

Parameters

Param	Type	Details
name	string	Name to store the query under.
value	object	The value of the query.

Example

```
QueryService.setQuery(name,value);
```

Events

- **query_name_updated**

When the query is set or changed Lightning will broadcast this message where name is the name of your query `query_name_updated`

Type:

broadcast

Target:

root scope

ResponseService

Description

Twigkit service for setting and getting responses.

Methods

- **clearResponses()**

Removes all responses from the factory.

Example

```
ResponseService.clearResponses();
```

- **createResponseFromResults(results)**

Creates a response object from the set of results you provide

Parameters

Param	Type	Details
results	array	The list of results to add to this response.

Returns

Object	Returns a newly created response object with the results provided.

Example

```
ResponseService.createResponseFromResults(results);
```

- **getResponse(name, keys)**

Gets response from the factory.

Parameters

Param	Type	Details
name	string	The name of the response to retrieve
keys (<i>optional</i>)	string	A comma separated list of keys you would like returned. If null whole response will be returned.

Returns

Object	Returns the response object from the factory, if it doesn't exist it will return <code>null</code>

Example

```
ResponseService.getResponse(name);
```

- **setResponse(name, value)**

Adds response to response factory for retrieval using the `getResponse` method. Broadcasts an event of 'responsename_updated' with the new response value.

Parameters

Param	Type	Details
name	string	The name of the response
value	Object	The response object to add to the factory.

Example

```
ResponseService.setResponse(name, value);
```

ScrollToService

Description

Appkit service providing functions for scrolling to the top or a particular element ID on the page.

Methods

- **scrollToElement(id)**

Scrolls to a specific element based on ID (if no element id is provided it will scroll to the top of the page).

Parameters

Param	Type	Details
id (<i>optional</i>)	string	The ID of the element to scroll to (if the ID is not defined then this function will scroll the user to the top of the page).

Example

```
ScrollToService.scrollToElement('foo');
```

TogglePanesservice

Description

Twigkit service for the toggle pane(s) components

Methods

- **onTogglePane(panesId, callback, scope)**

When the togglePanes event is fired the callback function will be called.

Parameters

Param	Type	Details
panesId	string	The id of the toggle panes component where the callback is being fired.
callback	Function	The function to be called when the cancel event is fired.
scope	Object	The <code>\$scope</code> object for unsubscribing to the event when the scope is destroyed.

Example

```
TogglePanesservice.onTogglePane('MyPaneId', function (paneNumber) {
  console.log('Pane has been toggled to ' + paneNumber); }, $scope);
TogglePanesservice.togglePane('MyPaneId', 1); </code></pre></div> </div>
</li> <li><h3 id="methods_togglepane">togglePane(panesId, paneNumber)</h3> <div
class="togglepane"><div class="lightning-togglepanesservice-togglepane-page">
<p>Fires an event to change the currently selected pane for a specific set of
panes.</p> </div><h5 id="methods_togglepane_parameters">Parameters</h5><table
class="variables-matrix table table-bordered table-striped"><thead><tr>
<th>Param</th><th>Type</th><th>Details</th></tr></thead><tbody><tr>
<td>panesId</td><td><a href="" class="label type-hint type-hint-string">string</a>
</td><td><div class="lightning-togglepanesservice-togglepane-page"><p>The id of
the toggle panes component to fire the toggle against.</p> </div></td></tr><tr>
<td>paneNumber</td><td><a href="" class="label type-hint type-hint-
number">number</a></td><td><div class="lightning-togglepanesservice-togglepane-
page"><p>The number of the pane to switch to (0 based so the first pane is 0, the
second pane is 1 etc.).</p> </div></td></tr></tbody></table><h4
id="methods_togglepane_example">Example</h4> <div class="example"><div
class="lightning-togglepanesservice-togglepane-page">
<p>TogglePanesservice.togglePane('MyPaneId', 1);</p> </div></div> </div> </li>
</ul> </div> </div>
```

TranslationService

Description

Appkit Translation Service

Methods

- **addBundle(name, bundle)**

Add a translation bundle with a specified name

Parameters

Param	Type	Details
name	string	
bundle	object	

- **get(term, bundle)**

Get a translation for a term if it exists, otherwise return the original term

Parameters

Param	Type	Details
term	string	The term to translate
bundle (optional)	string	The bundle names to use. Comma delimited (default: "default")

Returns

string	The translated term

- **put(key, translation, bundleName, override)**

Add a new translation

Parameters

Param	Type	Details
key	string	
translation	string	
bundleName (optional)	string	
override (optional)	string	

UserService

Description

Twigkit service for setting and getting User.

httpInterceptor

Description

Intercepts all `$http` requests that return with an error. It will then broadcast the event `httpInterceptorError` which will provide the error object containing the status code, status text and other error information.

Usage

```
angular module('myApp', ['lightning']) .*.run(['$rootScope',
'$templateCache','$location','$timeout', function ($rootScope, $templateCache,
$location,$timeout) { $rootScope.$on('httpInterceptorError', function(event,error) {
//File not found is HTML if(error.config.headers.Accept == 'text/html'){
if(error.status == '404'){ alert('Page Not Found!'); //Timeout is required to
redirect the page after all other events have fired $timeout(function(){
$location.path('search'); },0) } } } });
```

Events

- **httpInterceptorError**

Broadcast the event `httpInterceptorError` when a `$http` request receives an error, this will provide the error object containing the status code, status text and other error information.

Usage

```
` angular module('myApp', ['lightning']) .*.run(['$rootScope',
'$templateCache','$location','$timeout', function ($rootScope, $templateCache,
$location,$timeout) { $rootScope.$on('httpInterceptorError', function(event,error) {
//File not found is HTML if(error.config.headers.Accept == 'text/html'){
if(error.status == '404'){ alert('Page Not Found!'); //Timeout is required to redirect
the page after all other events have fired $timeout(function(){
$location.path('search'); },0) } } } });
```

Type:

broadcast

Target:

root scope

scrollAndResizeListener

Description

A Twigkit service that will fire an event when the window is scrolled or resized.

Methods

- **bindListener(listener)**

A method for binding functions to a listener.

Parameters

Param	Type	Details
listener	function	The function to run when the event occurs

Returns

function	Runs the function when the event occurs.

Example

```
scrollAndResizeListener.bindListener(function(){ console.log('Window Scrolled or Resized'); });
```

Accessing Data

The articles in this section describe key aspects of accessing data.

- *Configuring the Search Platform*

The Appkit search platform abstraction presents a unified way of carrying out common data operations such as searching, storing, and deleting information from the underlying repository. You can use multiple platforms on the same page to send a query to different repositories and aggregate the response. Each search platform is encapsulated as a `twigkit.platform.Platform` Java object.

- *Filter Syntax*

This article describes the syntax of a single Appkit search filter to include in an HTTP request.

- *Formulating a Query*

The Appkit Query tag (`<search:query>`) instantiates a search query that can be submitted to the underlying search platform.

- *Search Response*

The Response tag (`<search:response>`) sends a Query to a given Platform and stores the Response in a JSP variable.

Configuring the Search Platform

The Generic Platform tag can instantiate all platforms by name or configuration.

What is an Appkit platform?

The Appkit search platform abstraction presents a unified way of carrying out common data operations such as searching, storing, and deleting information from the underlying repository. You can use multiple platforms on the same page to send a query to different repositories and aggregate the response. Each search platform is encapsulated as a `twigkit.platform.Platform` Java object.

Searching

When a platform is used for searching, it will map an Appkit query to something the underlying search engine will understand and send it off. When the search engine responds the platform will again map the response from the engine specific structure to a generic Appkit search response that can be processed by other Appkit components such as workflow processors and UI widgets.

Storing data

If a particular platform supports storing data, then it will take care of adapting an Appkit representation of a textual content to a piece of data that the underlying engine can store, and send it off for indexing or saving.

Usage

This example shows how to use any of the platforms using the generic Platform tag in an html page. This would retrieve a platform configuration from a file (under the `conf` directory).

```
<search:platform var="platform" conf="platforms.fusion.data"></search:platform>
```

Configuration parameters

var (java.lang.String)

Variable name to reference this `twigkit.model.Platform` instance.

Default: `platform`

conf (java.lang.String)

Retrieve a `twigkit.model.Platform` instance by either configuration name, or a platform annotation class name such as `twigkit.search.solr.Solr`. This is not the same as the platform class name used by the `className` attribute. Use either this or the `name` attribute.

host (java.lang.String)

The URI of the Solr Server. Must be specified using the full URI (for example, `http://localhost:8983/solr`; `localhost:8983` will not work).

For configuration settings for a specific provider, see *Search Platforms*.

Filter Syntax

This article describes the syntax of a single Appkit search filter to include in an HTTP request.

```
<filter-expression> ::= <field-expression> <group-expression> <value-expression>
<display-expression> <field-expression> ::= <exclusion-expression> <field-
optional-expression> <field-name> <exclusion-expression> ::= "-" | "" <field-
optional-expression> ::= "~" | "" <group-expression> ::= "(" <group-number> ")"
<group-number> ::= <digit> | <digit> <group-number> <field-name> ::= <field-name-
prefix> <field-name-suffix> <field-name-prefix> ::= <letter> | "_" <field-name-
suffix> ::= <field-symbol> | <field-symbol> <field-name-suffix> <field-symbol>
::= <letter> | <digit> | "_" | "-" | "." | ":" | " " | "" <value-expression> ::=
<value-bracket-left> <filter-value> <value-bracket-right> <value-bracket-left>
::= "[" | "(" <value-bracket-right> ::= "]" | ")" <filter-value> ::= <value> |
<range-value> <value> ::= <string-expression> | <numeric-value> | <date-value>
<string-expression> ::= <partial-match> <string-value> <partial-match> <string-
value> ::= "'" <text> "'" | "\"" <text> "\"" <partial-match> ::= '|' | '*' <numeric-
value> ::= <floating-point> | <integer> <number-sequence> ::= <digit> | <number-
sequence> <digit> <integer> ::= "0" | <positive-digit> <number-sequence>
<floating-point> ::= <integer> | <integer> "." <number-sequence> <date-value> ::=
<four-digits> "-" <two-digits> "-" <two-digits> "T" <two-digits> ":" <two-
digits> ":" <two-digits> <two-digits> ::= <digit> <digit> <four-digits> ::= <two-
digits> <two-digits> <range-value> ::= <value> "<value> | "<value> |
<value> "<value> <display-expression-opt> ::= "" | "[" <display-expression> "]"
<display-expression> ::= "[" "" <display-value> "" "]" | "[" "" <display-
value> "" "]"
```

Where:

```
<letter> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y"
| "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" |
"m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
; <positive-digit> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<digit> ::= "0" | <positive-digit>
```

Here:

- `<group-number>` is an integer value denoting a grouping of filter constraints
- `<field-name>` is a the name of the field to apply the constraint to. A valid field name is one that matches the regular expression `[A-Za-z_][A-Za-z_0-9\-\.\:]`
- `<display-value>` is an optional label to attach to the filter
- `<filter-value>` is the actual constraint to be applied to the field named `<field-name>`

Formulating a Query

The Appkit Query tag (`<search:query>`) instantiates a search query that can be submitted to the underlying search platform.

Usage

This example shows how to create a Query using the Query tag. The Query tag can use all applicable parameters from the query string (HTTP request) to create a Query object.

As an example, if the URL of the page is `http://intranet/search?q=guidelines&rpp=10`, then the word 'guidelines' would be set as the query term, and the query would instruct the search engine to limit the results per page (rpp) to 10.

This example shows how you can override the `rpp` (short form of the `results-per-page` attribute) to fix how many results should be displayed per page, or override the value specified in the URL query string. To do this, add the long-form name of the attribute to override to the Query tag:

```
<search:query var="myQuery" parameters="*" results-per-page="5" />
```

The resulting query object can be accessed using an angular expression in the HTML page on which it is located. For example, `{{myQuery}}`.

You can also create a Query instance from another one using the `from` attribute for cloning the former:

```
<search:query var="query" from="otherQuery" results-per-page="5" />
```

Query Parameters

var (`java.lang.String`)

This is the name of the variable in which the query object will be stored.

Default: `query`

parameters (`java.lang.String`)

Parameters to use from the HTTP request to create the Query object. Set `parameters` to `*` to use all parameters. A comma-separated list of parameters to use is also valid.

from (`twigkit.model.Query`)

Create a new Query from existing Query object.

Adding any of these query parameters means that the value is fixed for all queries, that is, the value overrides any parameters inherited from the URL query string.

query (`&q=`) (`java.lang.String`)

Use the query (or `q`) parameter to specify the query term to use. This will override default-query parameter of the Platform.

modified-query (`&mq=`) (`java.lang.String`)

Specify an alternative query that will be submitted to the search engine in place of the query parameter supplied by the user (but can include user submitted parameter). To the user it will appear as the query parameter had been used, but the modified-query will be submitted to the search engine.

type (`&t=`) (`java.lang.String`)

Specifies how to interpret multiple query terms. Takes one of three values: `any` (boolean OR the terms), `all` (boolean OR/AND the terms), or `adv` (interpret using search platform-specific advanced query language). For example, take the query `apple ipod`. With

type="all" documents containing both the terms (for example, apple and ipod) will be returned, whereas with type="any" documents containing either term (for example, apple or ipod) will be returned. See special cases section below for type="adv" usage examples.
Default: any

results-per-page (&rpp=) (java.lang.Integer)

The number of results to display per page.
Default: 10

max-results (&max=) (java.lang.Integer)

The maximum number of results to evaluate.
Default: -1, which means no upper limit

page (&p=) (java.lang.Integer)

Set the page of results to display. Default values is 1 (1-based index) which would display the first page of results (1 to getResultsPerPage).
Default: 1

facets (&fa=) (java.lang.String)

Specifies which facets to evaluate and return. Specified as comma-separated list without spaces (for example, facets="cat,inStock"). Will override defaultFacets parameter of the Platform tag.

fields (&fi=) (java.lang.String)

A comma-separated list of fields that should be returned in the Response.

sorts (&s=) (java.lang.String)

Specifies fields and order for sorting. Sort order is specified by one of two characters: + for ascending and - for descending. To sort ascending by the field price, the sort would be specified as sorts="+price".

Multilevel sorting is supported. To sort by multiple levels using the QueryTag use & to separate the values, for example, to sort by price ascending and then sku descending, use sorts="+price&-sku". To do the same using the URL query string use multiple instances of the s parameter, for example, ?&s=+price&s=-sku.

view (&v=) (java.lang.String)

Set which search engine view to use. Behavior will vary by installation.

filters (&f=) (java.lang.String)

Specifies filters to apply, specified as an ampersand (&) separated String. In the query string however there will be multiple &f= arguments. Filters are used to restrict field values in that appear in results. Also see *Alternative way to add Filters* below.

facetsEnabled (java.lang.Boolean)

Whether to calculate and return facets for the query. Behavior is platform dependent.

custom (&c=) (java.lang.String)

The custom attribute lets you send custom parameters to the underlying search platform. Parameters are specified using the format parameter[value] for a single parameter and corresponding value. Multiple parameters can be submitted with the QueryTag by separating them with & characters. For example:

custom="parameter1[value1]&parameter2[value2]"

To submit multiple custom parameters using the URL query string, use multiple instances of the c parameter. For example:

?c=parameter1[value1]&c=parameter2[value2]

Short format

Filters are specified using the format fieldName[value]['display name']. The ['display name'] part is optional in all cases.

For example, adding filters="locations['san francisco']" to the Query tag restricts the results to those with san francisco appearing in the locations field.

Multiple filters can be specified on the Query tag by using the separator characters &. For example, filter="locations['san francisco']&people['obama']" would restrict the results to those with the value san francisco appearing in the locations field and the value obama appearing the people field.

Multiple filters can be specified with the URL query string by adding multiple `f` parameters:

```
?f=locations['san francisco']&f=people['obama']
```

If the value is a String it should be surrounded by quotes, all other primitives and dates should have the quotes omitted see below (URL query parameter examples).

For example:

- Filtering on String: `&f=stringField['My String']["String example"]`
- Filtering on Integer: `&f=integerField[123] ["Integer example"]``
- Filtering on Double: `&f=doubleOrFloatField[123.45] ["Double example"]`

Long format

When the query value of the filter differs from the display value the format is slightly different to accommodate this extra information: `filters="fieldName[value] ["display name"]`.

Range format

Appkit supports range queries for numbers, dates and strings. For example `filters=salary[1000,100000]` will search the salary field for values between 1000 and 100000.

Date format

To filter by date (on date fields allowing range queries) use the `DateFilter` format (in all cases the display name is optional):

- To Filter on dates before the provided Date use: `filters=dateFieldName[,yyyy-MM-dd'T'HH:mm:ss] ["Before the given date"]`
- To Filter on dates after the provided Date use: `filters=dateFieldName[yyyy-MM-dd'T'HH:mm:ss,] ["After the given date"]`
- To Filter on dates within a given range use: `filters=dateFieldName[yyyy-MM-dd'T'HH:mm:ss,yyyy-MM-dd'T'HH:mm:ss] ["Date range"]`
- From 2.10 onwards, to exclude the upper or lower limit of a date range it is possible to replace the `[` and `]` with a `(` and `)`. For example: `filters=dateFieldName[yyyy-MM-dd'T'HH:mm:ss,yyyy-MM-dd'T'HH:mm:ss) ["Date range excluding upper limit"]`

Invariant filters

To configure a query with *invariant constraints*, you can add filter tags to the body of the query tag like so:

```
<search:query var="query" ... > <query:filter field="myField" value="myValue">
</query:filter> </search:query>
```

This constrains the query to all documents with the value "myValue" in "myField". It is not possible to use ranges with the `<query:filter>` tag. The value will always be interpreted as a string.

See the `query:filter` tag doc for available parameters.

Optional Operators

Appkit supports combining filters using different logical operators (connectives).

Enabling Logical Disjunction for Filters (OR connectives)

There are two ways of combining a set of filters using an inclusive disjunction where each filter represents an logically optional condition whilst keeping filters individually distinct. Each filter's optional state is represented as being either 'value', 'field' or 'none', with the corresponding Java Enumerations below.

'value' – Values within a given field

Filters for the same field that are 'value' optional will be combined using an OR operator. This group of filters will then be combined with the rest of the filters using a logical AND.

This would be the recommended approach when using checkboxes to select values within a facet. In that case the default behavior would be to allow the user to select multiple options which would be combined using an OR. This is encoded in the query URL using an asterisk after the filter value:

```
&f=foo['bar']&f=marx['groucho']*&f=marx['harpo']*
```

This would result in a query like: `(foo:bar AND (marx:groucho OR marx:harpo))`

'field' – Fully optional condition

To make a given filter's condition fully optional you would use the 'field' flag. This would mean that these filters are combined with the rest using an OR.

This is encoded in the query URL using a tilde before the field name:

```
&f=foo['bar']&f=~marx['groucho']&f=~marx['harpo']*
```

This would result in a query like: `foo:bar) OR (marx:groucho) OR (marx:harpo)`

'none' – Logical Conjunction, default (AND)

By default, individual filters (&f=) are applied to the query using logical conjunction (AND). This means that the documents returned must satisfy both filter conditions. This is the default connective.

Alternative way to add Filters

Another way to add Filters to a Query is using dynamic attributes on the Query tag. Any parameters other than the ones detailed above will be treated as field names, and their corresponding value as a filter expression.

For example, this Query tag would create a query from the URL query string and additionally add two hidden filters 1) the source field should contain the value FT and 2) the `bestbets` field should be true:


```
<search:query var="query" parameters="*" results-per-page="5" source="FT"
bestbets="true" />
```

To add exclude filters prefix the filter name with not-. This would add a filter that excludes all documents where the source field contains the value FT:

```
<search:query var="query" parameters="*" not-source="FT" />
```

Tag body

Queries may include the `query:filter` tag and the `query:custom` tag in the body of a `search:query` tag to add invariant query filters or custom engine parameters, respectively.

```
<search:query var="query" parameters="*" maxResults="5"> <query:filter
field="myField" value="myValue"></query:filter> <query:custom name="facet.range"
value="year"></query:custom> <query:custom name="facet.mincount" value="1">
</query:custom> </search:query>
```

Search Responses

The Response tag (`<search:response>`) sends a Query to a given Platform and stores the Response in a JSP variable.

Usage

This example shows how to submit a query using a given Platform and retrieve the response:

```
<search:response var="myResponse" platform="platform" query="query">
</search:response>
```

The resulting response object can be accessed using an angular expression in the HTML page on which it is located. For example, `{{myResponse}}`.

Configuration parameters

var (`java.lang.String`)

Variable name to reference the Response from the Platform.

Default: `response`

platform (`twigkit.platform.Platform`)

The Platform instance to use.

query (`twigkit.model.Query`)

An instance of an Appkit Query to send to the Platform.

errorPage (`java.lang.String`)

Error page to redirect to when Platform throws an exception.

Default: `/WEB-INF/tags/widgets/error/error.jsp`

on (`java.lang.String`)

Event that is broadcasted for when the response should reload. Default: `reloadSearch`. (optional)

Tag body

The body of the Response tag can be used to trigger a `widget:spinner` as the app waits for the response to return from the platform.

```
<search:response var="response" platform="platform" query="query" error-
page="/error"> <widget:spinner></widget:spinner> </search:response>
```

Modules

Appkit's suite of search components is modular, which means you are able to get a working solution into place quickly using tried-and-tested components.

Activity Tracking

Appkit can capture any and all interaction users have with an Appkit powered application. This includes what queries were typed, filters applied, sort order used, and links opened.

The information is stored in a highly scalable embedded search index which can be used to:

- Data mine or analyze information through a regular Appkit user interface
- Empower and aid users in content discovery by grouping together similar queries using statistical methods and suggesting alternatives to users seeking similar content
- Perform auditing and compliance functions in secure applications

Read more

Collaboration

The social and collaboration tools provided by the Social module extend the search application with the ability to store user-generated information on individual documents and document collections, and to share this information among groups of authenticated users. For example, documents can be tagged or bookmarked, and users can save search queries and add comments to individual documents.

Features include:

- **Saved searches.** Users can save important keyword search queries, apply filters, and retrieve the information later.
- **Saved results.** Users can bookmark results, add them to a basket, and search those saved items.
- **Rating.** Users can choose to rate or "like" results, giving a social signal to their peers. Rating analytics can be used to influence ranking.
- **Tagging.** Users can tag content with keywords, which content owners can use to improve data quality.
- **Annotations.** Users can annotate/comment on results and edit or delete their own comments. Users cannot edit or delete the comments of others.
- **Topics.** Users can create topics and any captured metadata. For example, saved searches can be associated with a topic.

Read more

Geospatial

Esri Web Maps

The Esri Web Map tag is used within the Geo Map tag to present content from an ArcGIS web map hosted with Esri or on premises.

Maps

The Map tag displays results on a geographical map, abstracting the underlying map provider.

Media

The Media module supports on-the-fly *image transcoding* to mobile formats and resizing including cropping and square cropping. It further supports capturing and transcoding particular pages from multi-page TIFF files and features built in caching. The Media module also supports efficient *exporting search results* to CSV and Excel.

Read more

Export Services

The Excel export service is constrained by the fact that the Excel document needs to be created in memory and therefore recommended for relatively small result sets. For larger exports, we recommend using the CSV export service.

Image Formatting and Transcoding

The Image tag displays images from a given field in a result. It will automatically resize, and cache the image for performance.

Security

Read more

Authentication

OAuth

Authentication against OAuth 2.0 services is possible using the individual authentication provider modules which simplify the configuration of OAuth parameters and deal with the differences that exist between common OAuth endpoints.

A general OAuth provider can be used by including the `twigkit.security.provider.oauth` module. This requires a custom `spring-security.xml` file, but it is recommended to use one of the prepackaged modules we provide for these services.

Read more

SAML

We have a method of configuring SAML authentication with a simplified process. SAML is still a complex standard and so a lot of the relevant information from the previous SAML instructions is still included here.

Appkit provides an extension for SAML based authentication. SAML is a recognized standard for authentication with many providers and implementations available.

Read more

SAML Troubleshooting

The SAML authentication standard is a broad XML specification covering a range of functionality and 'binding' protocols. As such, it is not uncommon to encounter problems integrating applications (Service Provider's) with SAML Identity Providers. Numerous technologies can form the SSO back-end (Identity Provider) and there is no 'one-size-fits-all' approach.

Authorization

Role-based Access

In many cases, you might want to restrict access to your application, or parts of your application, to specific groups of users. To enable this option, you must add the file `conf/security/access.conf` to your application's configuration tree.

Configuration

User Web Service

Although it is disabled by default, there is a web service that returns all of the details for the currently logged in user.

Every operation and capability of Appkit can be invoked via REST-ful web service endpoints supporting both XML and JSON responses for asynchronous client-side operations, and integrating with external systems. For example, users can perform secure searches to any platform supported by Appkit.

Visualizations

The *chart tag library* is a collection of JSP tags for creating charts to visualize both search results (document result sets) and facets.

Read more

Adding Credits

To display credits on the chart in the bottom right hand corner, use the Chart Credit tag with the `text` and `href` attributes.

Chart Legend

A legend is displayed when the chart has multiple series. With the Chart Legend tag, you can customize the legend with a title, border and location on the chart along with other things. This tag must be nested in the *Chart Display tag*.

Chart Tooltips

Tooltips display information when you hover over a point or area of the chart. Tooltips can be set across the whole chart or specifically to a series using the Chart Tooltip tag.

Displaying a Chart

The Chart Display tag displays the facet provided in a chart.

Exporting a Chart

To give users the option to export a chart, include the Chart Export tag within the Chart Display tag. Adding the tag will enable exporting without adding any attributes.

Faceting a Data Series

To render information in facets, use the Series Facet tag, which must be nested within a *Chart Display tag*.

Histogram

The histogram visualization displays the filters of a facet as a left-to-right series of bars.

Search Result Data Series

To render information from fields of a result list into a data series, use the Series Result List tag and use the `label-field` and `value-field` attributes.

Tag Cloud

Visualize the top filters of a facet as a tag cloud.

Workflow and Data Processing

A workflow processor is an atomic unit of business logic that can modify a query before it is submitted to a search platform, or transform a search response before rendering. This capability is used extensively for doing runtime data cleaning when, for example, it is not feasible to re-index a whole collection to fix issues in the source data.

Read more

Query Processors

Add a Filter to a Query. By creating or passing in a Filter instance, you can override or set all attributes of the filter that will then be added to the Query.

Read more

Response Processors

Facet Processors

Facet response processors include ones to perform these operations:

- Duplicate a facet.
- Create facets.
- Fill in date filters for a facet.
- Parse and formatting filter values from one Date format to another.
- Create a pseudo date facet.
- Set or changing Parameters for a facet.
- Sort facets based on various properties and parameters.
- Process facet filter values formatted as paths to represent them as hierarchical trees.
- Remove facets from a Response.
- Create pseudo facets based on values in a given field or fields.
- Process hierarchical facets so that only one level is rendered at a time.
- Define a pseudo facet from a list of facets in the response.

Field Processors

Field response processors include ones to perform these operations:

- Capitalize the display value of the given field names.
- Add Highlighting to Fields.
- Tag a document with classifications based on field values.
- Format a date object, replacing the original date value with another.
- Parse dates out of field values.
- Set the value of a field that is missing a value, based on the value of a different field.
- Parse the String value of the specified fields into an Object representation.
- Create a multivalued field from a single field value by using a separator.
- Extract the hostname from URLs and place it in a field named 'site'.
- Process fully qualified URLs in field values and markup, and add anchor tags for active links in the display values.
- Statically add metadata to documents that match a given regular expression.
- Replace field values (actual, display, or both) that are HTML or URL encoded with decoded values.
- Replace field values (actual, display, or both) that match a given regular expression with a different value.
- Make Twitter users and hashtags clickable in the display value.
- Duplicate a field, creating two separate instances.
- Create a new field by joining multiple existing fields using a pattern expression.
- Localize the values of a field using a specified bundle.

Filter Processors

Filter response processors include ones to perform these operations:

- Capitalize the display value of the given filter.
- Parse dates out of facet filter values.
- Sort facet filters by count.
- Sort facet filters by actual value.
- Replace display values in FacetFilters for specified or all Facets.
- Retrieve other filter values for the same facet, despite there being one or more filters for that facet already applied.
- Remove facet filters that match a regular expression pattern from a facet.
- Replace filter values (actual, display, or both) that match a given regular expression with a different value.
- (Since 4.2.0) Dynamically format the display value of a date range facet according to the size of the range.

General Response Processors

General response processors include ones to perform these operations:

- Perform a left join on fields.
- Log response metadata and attributes to log4j.
- Group results by a given Field value.

Activity Tracking

Appkit can capture any and all interaction users have with an Appkit powered application. This includes what queries were typed, filters applied, sort order used, and links opened.

The information is stored in a highly scalable embedded search index which can be used to:

- Data mine or analyze information through a regular Appkit user interface
- Empower and aid users in content discovery by grouping together similar queries using statistical methods and suggesting alternatives to users seeking similar content
- Perform auditing and compliance functions in secure applications

Activity processing

The activity tracking framework supports these endpoints for processing activity events:

- *Lucidworks Fusion Signals*
- *Log file writer*

Query Tracking

To track user-generated queries on your page, add a `<track:query>` tag with a reference to the query object that captures the user's input:

```
<track:query query="query" path="/search" page-title="Search Page"></track:query>
```

To see all related attributes, see the *track:query* tag documentation.

Click Tracking

To track click events, wrap a `<track:clicks>` tag around any result list to be tracked:

```
<track:clicks query="query" type="People Results"> <search:result-list  
response="response"> <search:result> ... </search:result> </search:result-list>  
</track:clicks>
```

To see all related attributes, see the *track:clicks* tag documentation.

Alternatively, to track a static URL, wrap the address in your href as shown. Optionally, you can use the `_type` parameter to provide a click type:

```
/twigkit/secure/services/url?url=http://www.google.com&_type=Other_Sites"
```

Visualizations

The *chart tag library* is a collection of JSP tags for creating charts to visualize both search results (document result sets) and facets.

Usage

You can visualize your data using the charts markup language. As an example, let us say you want to plot distinct segments of a search response facet as a chart, with each segment shown in proportion to size of that segment. For this you would add the *Chart Display* tag to your page, passing in a reference to the `twigkit.model.Response` containing the facet you want to plot and specify the facet name and type of chart you want to render. For example:

```
<chart:display response="response" facet-name="current_products" title="Current Products" type="pie"></chart:display>
```

In some cases, you might want to specify more options for your chart, or provide multiple data series to plot, for example. You do that by opening the `chart:display` tag and providing one or more chart options or tags in the markup body:

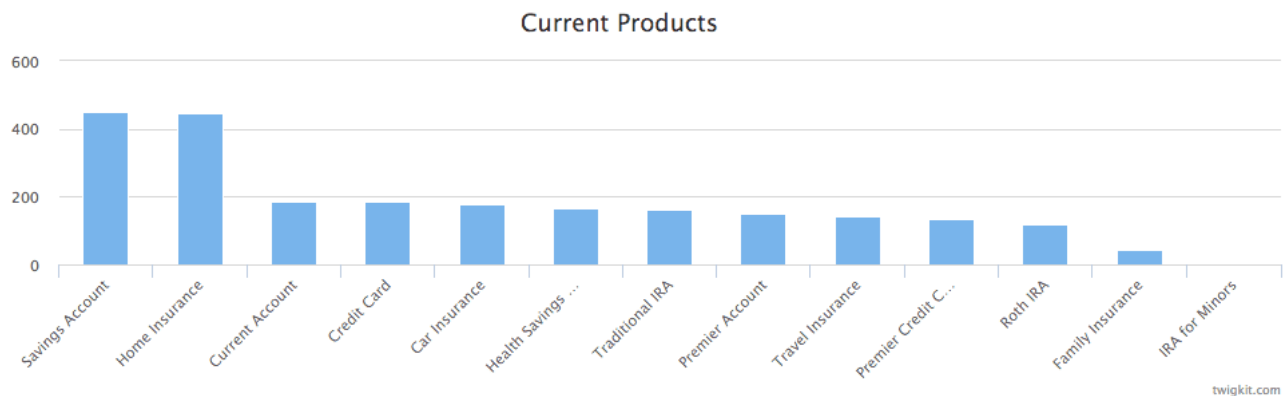
```
<chart:display response="response" title="Current Products"> <series:facet name="current_products" type="column" title="Current Products"></series:facet> <chart:tooltip header-format="<b>{point.x}</b><br/>" point-format="{series.name}: <b>{point.y}</b><br/>"></chart:tooltip> </chart:display>
```


Adding Credits

To display credits on the chart in the bottom right hand corner, use the `Chart Credit` tag with the `text` and `href` attributes.

This tag must be nested in the `Chart Display` tag.

```
<chart:display response="response" title="Current Products" type="column" facet-  
name="current_products"> <chart:credit href="http://twigkit.com"  
text="twigkit.com"></chart:credit> </chart:display>
```



To see all related attributes, see the `chart:credit` tag documentation.

Chart Legend

A legend is displayed when the chart has multiple series. With the `Chart Legend` tag, you can customize the legend with a title, border and location on the chart along with other things. This tag must be nested in the `Chart Display` tag.

```
<chart:display response="response" title="Current Products vs Recommended  
Products" stacking="normal"> <series:facet name="current_products" type="column"  
title="Current Products"></series:facet> <series:facet  
name="recommended_products" type="column" title="Recommended Products">  
</series:facet> <chart:legend title="Legend Title" align="right" vertical-  
align="top" layout="vertical" y="100" border-radius="5" border-width="1">  
</chart:legend> </chart:display>
```

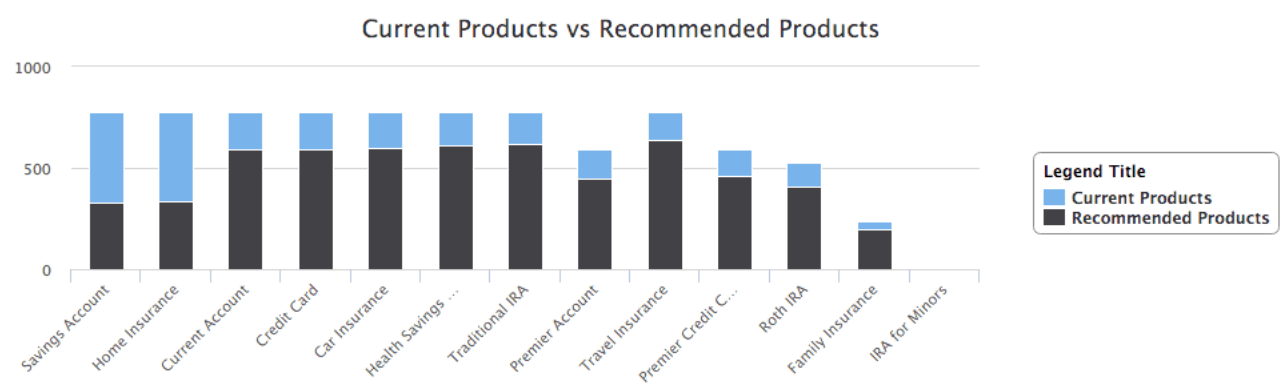


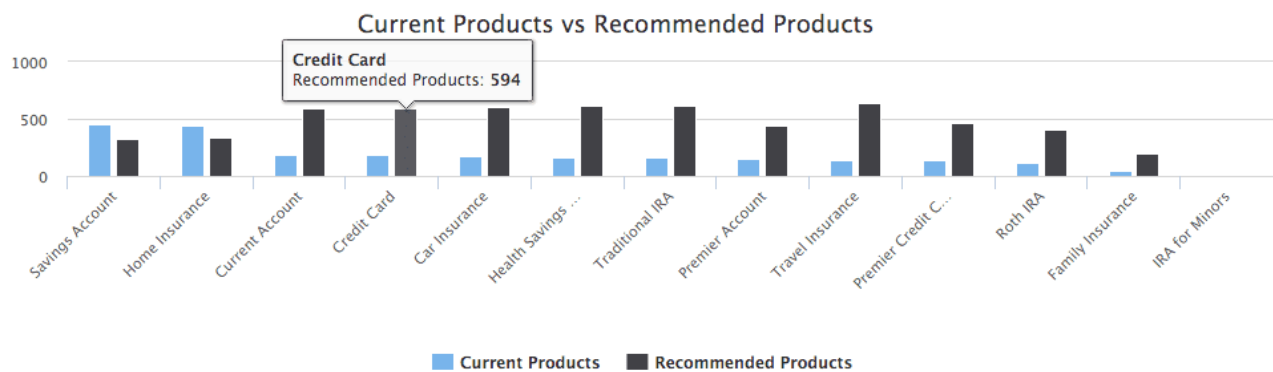
Chart Tooltips

Tooltips

Tooltips display information when you hover over a point or area of the chart. Tooltips can be set across the whole chart or specifically to a series using the Chart Tooltip tag.

This tag must be nested in the *Chart Display* tag, or in either of the series tags (*Series Facet* tag and *Series Result List* tag) if the tooltip is specific to a series.

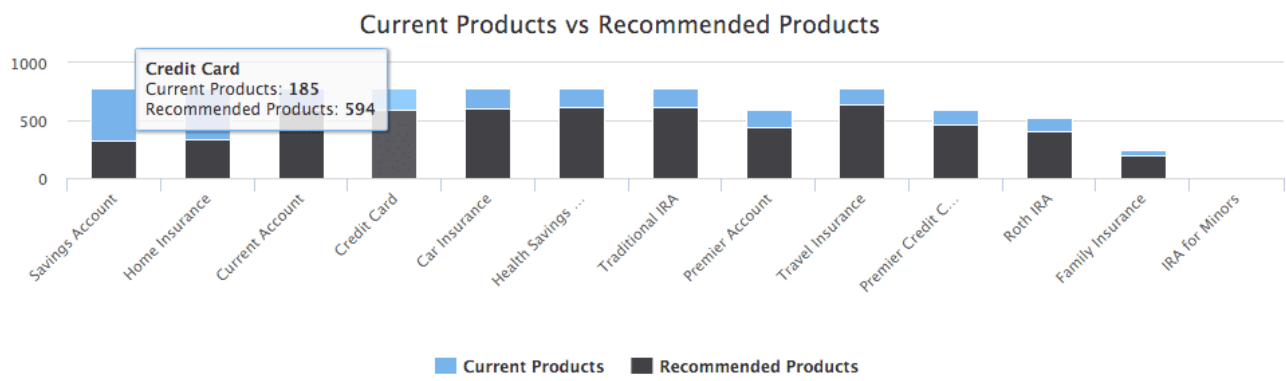
```
<chart:display response="response" title="Current Products vs Recommended Products"> <series:facet name="current_products" type="column" title="Current Products"></series:facet> <series:facet name="recommended_products" type="column" title="Recommended Products"></series:facet> <chart:tooltip header-format="<b>{point.x}</b><br/>" point-format="{series.name}: <b>{point.y}</b><br/>"></chart:tooltip> </chart:display>
```



Shared Tooltip

If you are rendering more than one series you can set the tooltip to be shared, this means it will display information about all the series for the point you have hovered over.

```
<chart:display response="response" title="Current Products vs Recommended Products" stacking="normal"> <series:facet name="current_products" type="column" title="Current Products"></series:facet> <series:facet name="recommended_products" type="column" title="Recommended Products"></series:facet> <chart:tooltip headerFormat="<b>{point.x}</b><br/>" point-format="{series.name}: <b>{point.y}</b><br/>" shared="true"></chart:tooltip> </chart:display>
```



To see all related attributes, see the `chart:tooltip` tag documentation.

Displaying a Chart

The Chart Display tag displays the facet provided in a chart.

The body of the Chart Display tag can contain a *Series Facet* tag for rendering multiple series using facets and also a *Series Result List* tag for rendering multiple series using results.

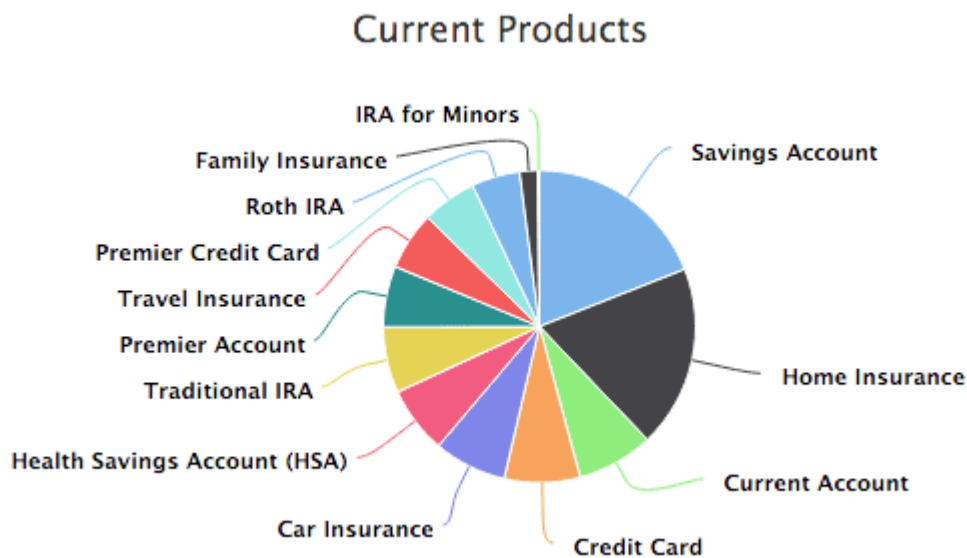
Along with this, there are other tags to customize the chart. These include the *Chart Tooltip* tag for modifying the tooltip, *Chart Legend* tag for modifying the legend, *Chart Credit* tag for adding a credit to the chart, and the *Chart Export* tag for exporting the chart.

Usage

Displaying a facet

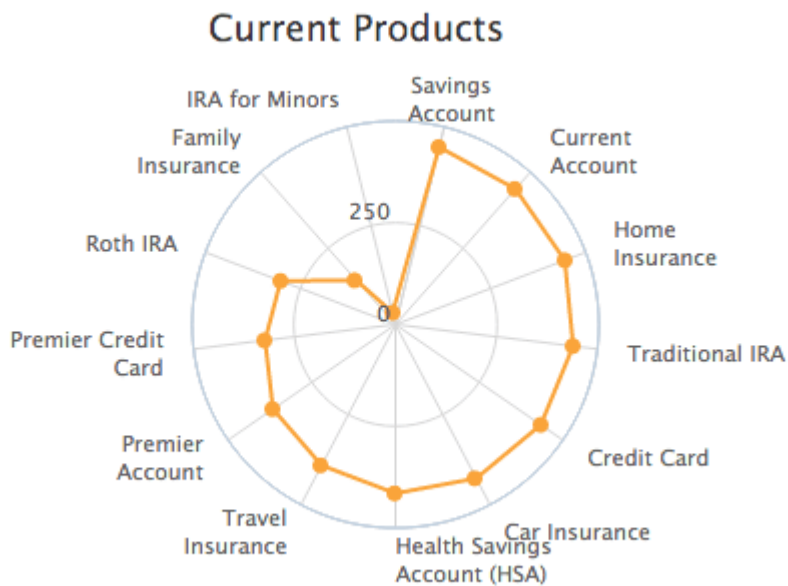
To visualize information in facets using frequency to build the graph, use the facetName attribute:

```
<chart:display response="response" facet-name="current_products" title="Current Products" type="pie"></chart:display>
```



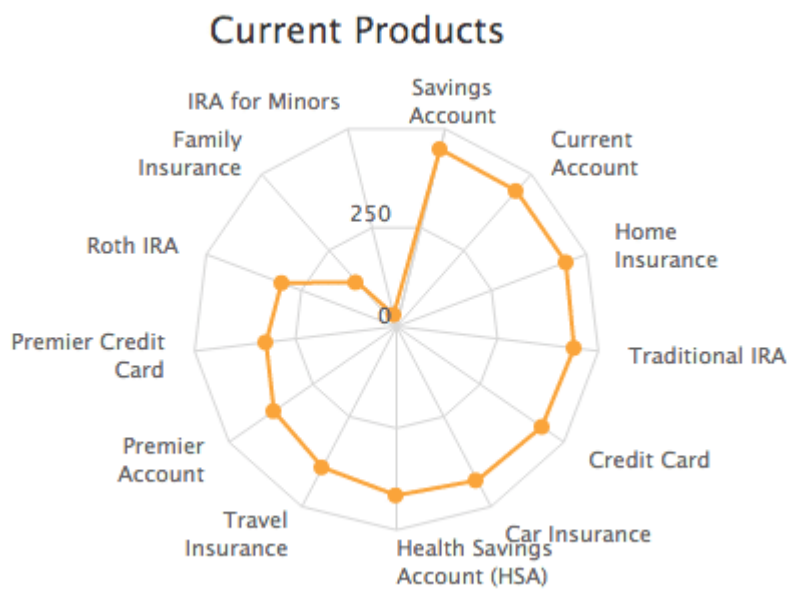
Displaying a polar chart

```
<chart:display response="response" facet-name="current_products" title="Current Products" type="line" radar-or-polar="polar"></chart:display>
```



Displaying a radar chart

```
<chart:display response="response" facet-name="current_products" title="Current Products" type="line" radar-or-polar="radar"></chart:display>
```



To see all related attributes, see the *chart:credit* tag documentation.

Exporting a Chart

To give users the option to export a chart, include the `Chart Export` tag within the `Chart Display` tag. Adding the tag will enable exporting without adding any attributes.

This tag must be nested in the `Chart Display` tag.

```
<chart:display response="response" title="Current Products vs Recommended Products"> <series:facet name="current_products" type="column" title="Current Products" color="blue"></series:facet> <series:facet name="recommended_products" type="line" title="Recommended Products" color="#006600"></series:facet> <chart:export></chart:export> </chart:display>
```



To see all related attributes, see the `chart:export` tag documentation.

Faceting a Data Series

To render information in facets, use the `Series Facet` tag, which must be nested within a `Chart Display` tag.

Usage

```
<chart:display response="response" title="Current Products vs Recommended Products"> <series:facet name="current_products" type="column" title="Current Products" color="blue"></series:facet> <series:facet name="recommended_products" type="line" title="Recommended Products" color="#006600"></series:facet> </chart:display>
```



To see all related attributes, see the `series:facet` tag documentation.

Histogram

The histogram visualization displays the filters of a facet as a left-to-right series of bars.

Usage

Simply pass a facet name into the histogram visualization to see that facet presented as a histogram chart.

```
<chart:histogram facet-name="facet" show="20"></chart:histogram>
```

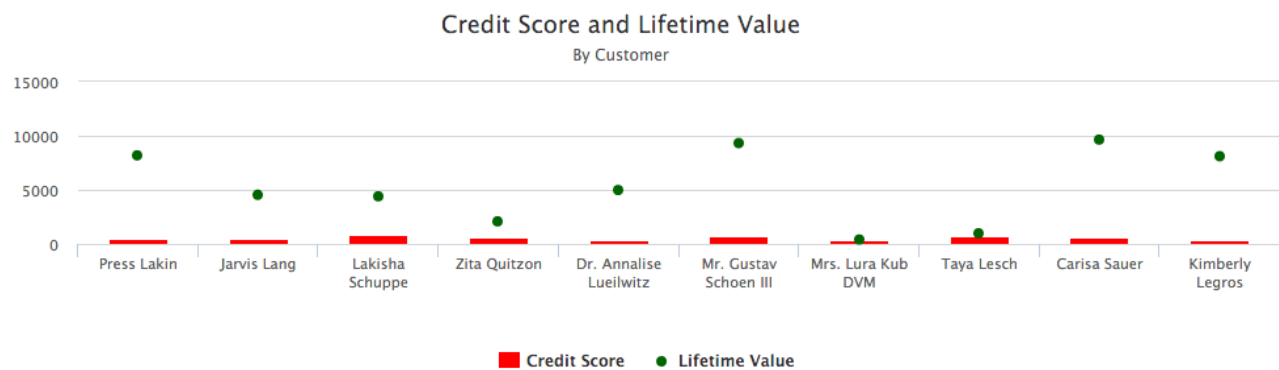
To see all related attributes, see the `chart:histogram` tag documentation.

Search Result Data Series

To render information from fields of a result list into a data series, use the `Series Result List` tag and use the `label-field` and `value-field` attributes.

This tag must be nested in the *Chart Display* tag.

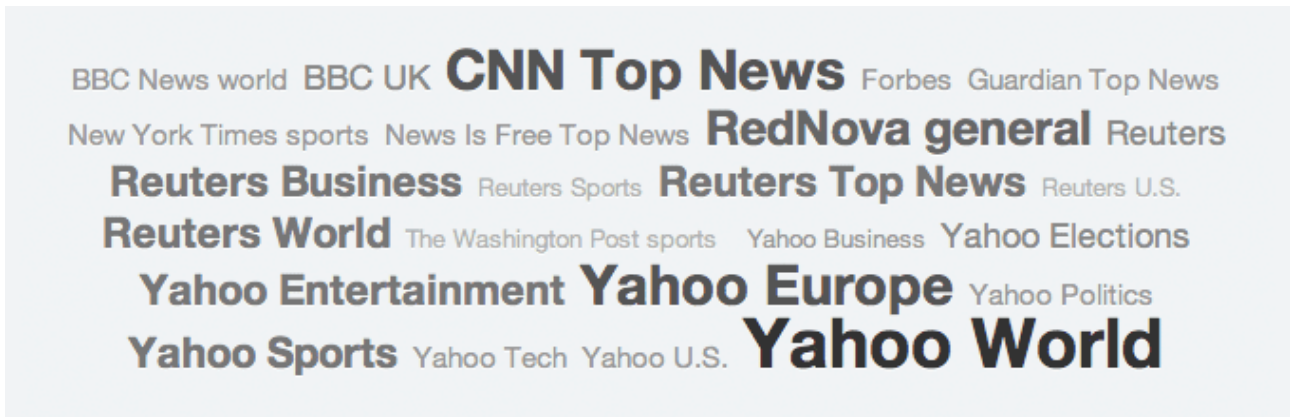
```
<chart:display response="response" title="Credit Score and Lifetime Value"
  subtitle="By Customer"> <series:result-list label-field="name" value-
  field="credit_score" type="column" title="Credit Score" color="red">
</series:result-list> <series:result-list label-field="name" value-
  field="lifetime_value" type="scatter" title="Lifetime Value" color="#006600">
</series:result-list> </chart:display>
```



To see all related attributes, see the *series:result-list* tag documentation.

Tag Cloud

Visualize the top filters of a facet as a tag cloud.



Usage

To add a tag cloud, specify the response, which facet to display, and (optionally) how many tags to display.

```
<chart:tagcloud response="response" facet-name="facet"></chart:tagcloud>
```

To see all related attributes, see the *chart:tagcloud* tag documentation.

Security

Authentication

The Security Module provides authentication and authorization capabilities to the search application. It supports multiple authentication providers and will pass authentication credentials on to search engines capable of handling authorization (for example, the FAST Security Access Module), providing personalization and role based filtering. The Security Module can be extended and customized via a pluggable API.

The Security module supports a number of authentication providers, including:

- **Single sign-on** via Windows Integrated Authentication
- **Container-managed** authentication
- **Form-based** authentication
- **Cookie-based** authentication
- **Pass-through** authentication
- **Custom authentication providers** via API

Role-based Access

In many cases, you might want to restrict access to your application, or parts of your application, to specific groups of users. To enable this option, you must add the file `conf/security/access.conf` to your application's configuration tree.

User Web Service

Although it is disabled by default, there is a web service that returns all of the details for the currently logged in user.

Every operation and capability of Appkit can be invoked via REST-ful web service endpoints supporting both XML and JSON responses for asynchronous client-side operations, and integrating with external systems. For example, users can perform secure searches to any platform supported by Appkit.

Authentication

The Security Module provides authentication and authorization capabilities to the search application. It supports multiple authentication providers and will pass authentication credentials on to search engines capable of handling authorization (for example, the FAST Security Access Module), providing personalization and role based filtering. The Security Module can be extended and customized via a pluggable API.

The Security module supports a number of authentication providers, including:

- **Single sign-on** via Windows Integrated Authentication
- **Container-managed** authentication
- **Form-based** authentication
- **Cookie-based** authentication
- **Pass-through** authentication
- **Custom authentication providers** via API == Authentication methods

OAuth

Authentication against OAuth 2.0 services is possible using the individual authentication provider modules which simplify the configuration of OAuth parameters and deal with the differences that exist between common OAuth endpoints.

A general OAuth provider can be used by including the `twigkit.security.provider.oauth` module. This requires a custom `spring-security.xml` file, but it is recommended to use one of the prepackaged modules we provide for these services.

Read more about OAuth authentication.

SAML

We have a method of configuring SAML authentication with a simplified process. SAML is still a complex standard and so a lot of the relevant information from the previous SAML instructions is still included here.

Appkit provides an extension for SAML based authentication. SAML is a recognized standard for authentication with many providers and implementations available.

Read more about SAML authentication.

Available security providers

Appkit supports a range of security providers for authentication and authorization. Typically one of these providers is used in an application:

Active Directory

While integration with Active Directory can be achieved using the LDAP setup, Spring Security provides a tailored security provider which makes configuring security using Active Directory easier.

Set Up Active Directory Authentication

Fusion Sessions API

The Fusion security provider for Appkit allows authentication against Fusion using the Fusion Sessions API. With this provider configured, an application deployed within a Fusion defined 'realm' such as `native` can securely authenticate a user against Fusion.

Set Up Lucidworks Fusion Authentication

HTTP Headers

Appkit supports authentication via generic HTTP headers passed through from a component behind a secure firewall sitting in front of the Appkit application. If there is such an authentication component in the environment where you are using Appkit, you can configure the HTTP headers that the application will use to pick up the user's details.

Set Up HTTP Headers Authentication

Kerberos

The Kerberos security provider for Appkit allows authentication using Kerberos. With this provider configured, an application deployed within a Kerberos 'realm' can securely authenticate a user against a Kerberos 'Key Distribution Center'. This module is primarily intended to simplify the process of authenticating a user silently in a Windows environment by leveraging the built in support for integrated Windows authentication in Internet Explorer. However it should be possible to use this module to authenticate the user in any Kerberos compliant environment.

Set Up Kerberos Authentication

LDAP

Appkit allows for authentication against a server using LDAP (Lightweight Directory Access Protocol). It assumes that your Appkit application is managed as a Maven (<https://maven.apache.org/>) project with all project dependencies described in a Maven POM file (<https://maven.apache.org/pom.html>).

Set Up LDAP Authentication

SiteMinder

Appkit can integrate with a SiteMinder single-sign-on authentication provider. SiteMinder is typically installed in a web server placed in front of the application server that runs Appkit.

Set Up SiteMinder Authentication

Spring Security

The Spring Security authentication provider for Appkit interfaces with both Spring Security (<http://projects.spring.io/spring-security/>) 3 and 4. Projects that adopt Appkit version 3 must switch to using Spring Security 4.

Authorized Spring principals become fully-qualified Appkit user profiles, along with any roles or groups that the Spring module provides. Spring Security integrates with a number of authentication protocols, including LDAP, Active Directory, Basic Authentication, Kerberos, JAAS, and others. For a full list, see here. (<http://docs.spring.io/spring-security/site/docs/3.1.1.RELEASE/reference/introduction.html>)

Set Up Spring Authentication

Tomcat Container

Apache Tomcat (https://hub.docker.com/_/tomcat) is an open source Java servlet container. Tomcat implements Java enterprise specs, including the Java Servlet, JavaServer Pages JSP, and WebSockets APIs.

Set Up Tomcat Container Authentication

Windows

Appkit can be used with Integrated Windows Authentication (http://en.wikipedia.org/wiki/Integrated_Windows_Authentication) for single sign on (SSO) (http://en.wikipedia.org/wiki/Single_sign-on) in your search application using Microsoft Internet Information Services (<http://www.iis.net/>).

Set Up Windows Authentication

Configuration

Primary configuration is done via the `security.conf` configuration file and other files in the `/conf/security/` folder. The Security Module however also supports mechanisms like Spring Security that might require its own configuration files (for example, `spring-security.xml`) and custom elements in `web.xml`.

Define the security provider in `security.conf` using one of the following options:

```
type: spring_security type: saml type: kerberos type: tomcat7 type: generic type:
siteminder type: passthrough type: oauth
```

Spring security

Spring Security is an industry standard security framework for Java. It is a popular option allowing the security to be configured initially via a simple XML file, and offers a lot of flexibility due to its integration with many different technologies, supporting a range of security standards. The spring documentation (<http://projects.spring.io/spring-security/>) contains thorough guides to the use of Spring Security.

By default Appkit applications are configured to use Spring Security with a simple hard-coded set of credentials in the provided `spring-security.xml` file, but you can see some other common configurations *defined here*. Further documentation on configuring Spring Security can be found here (<http://docs.spring.io/spring-security/site/docs/3.0.x/reference/ns-config.html>).

Appkit can be configured to use Spring Security for both authentication and role extraction.

Container managed security

In this case the application server is responsible for authentication such as:

- Internet Information Server and Integrated Windows Authentication
- Tomcat 7 Container Managed

The application server can be configured to pick up the user's credentials when the request is forwarded from a 'front-end' like IIS or Apache. In cases where authentication is done by the container it is still possible to do role extraction using Active Directory or LDAP for the user provider (see below).

Generic security

The generic security option uses `getRemoteUser()` or the `REMOTE_USER` variable provided through the servlet API. This is for authentication only, but it can be combined with other methods for picking up the roles assigned to a user and submitting them to the search platform (see Active Directory and LDAP role extraction below).

Custom Providers

You can add your own `SecurityProvider` which would be responsible for providing the user identity to Appkit. To do this, implement the `twigkit.security.SecurityProvider` interface and bind the class using Guice in your custom module.

Authorization, Role Extraction, and other features

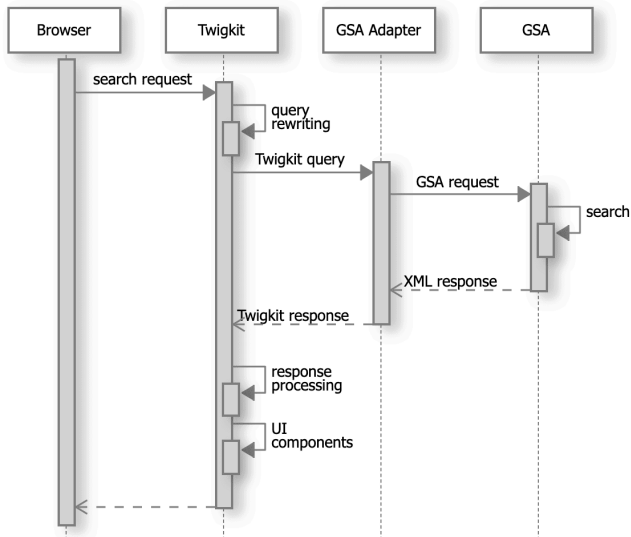
Active Directory and LDAP Role Extraction

After the user's identity has been established during the authentication stage, it is possible to carry out further authorization tasks such as role extraction or adding other metadata about the current user.

User and Domain Mapping

Appkit Security Module comes with pre-built filters to handle mapping between user names and domains. It is also possible to provide your own logic during this phase to deal with specific scenarios for determining a user's credentials and details before rendering the page or passing them to the search platform.

Security lifecycle



1. Authentication

When the Security Module is installed, every request made to Appkit is filtered through the configured authentication provider. Credentials are added to the request via a ThreadLocal variable. Authentication providers can be container managed, single sign on or configured for form based authentication in Appkit. You can create your own authentication provider by implementing and binding the twigkit.security.SecurityProvider class.

2. Role Extraction

Next step in the filter chain can handle role/group extraction for the authenticated user. This is frequently done via an Active Directory/LDAP lookup. The role extraction mechanisms will add the roles for the user to the User object.

3. Query

When the Security Module is used queries should be created using the secure version of the Query tag. The secure query tag will provide a reference to the user that is logged in and pass the identity on to the search platform if necessary. The Security Module also provides a number of processors and utilities for adding filters to the query based on business rules or identity/role information.

4. Personalization

The User and Role information is available in the user interface. Using the Security Conditional tag you can customize the view to the user that is logged in.

OAuth

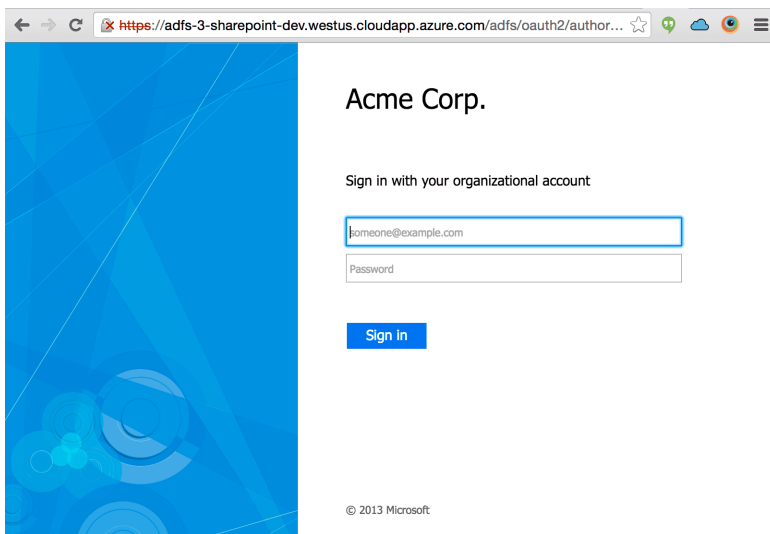
Authentication against OAuth 2.0 services is possible using the individual authentication provider modules which simplify the configuration of OAuth parameters and deal with the differences that exist between common OAuth endpoints.

A general OAuth provider can be used by including the `twigkit.security.provider.oauth` module. This requires a custom `spring-security.xml` file, but it is recommended to use one of the prepackaged modules we provide for these services.

Using OAuth can create a seamless experience where, after the token handshake and approval is complete, the user will always be logged into the application as long as their session is active.

ADFS

The Active Directory Federation Services (ADFS) security provider is available in Appkit.



OAuth based authentication against ADFS requires **ADFS 3** (or newer) which is available from Windows Server 2012 R2 onwards.

Learn how to set up ADFS authentication.

Facebook

Learn how to set up Facebook authentication.

Google

Learn how to set up Google authentication.

Office 365

Learn how to set up Office 365 authentication.

SAML Configuration

We have a method of configuring SAML authentication with a simplified process. SAML is still a complex standard and so a lot of the relevant information from the previous SAML instructions is still included here.

Appkit provides an extension for SAML based authentication. SAML is a recognized standard for authentication with many providers and implementations available.

To learn how to configure SAML based authentication, see the *how-to article*.

Typically SAML provides Single-Sign-On (SSO) between services that can be accessed using the same identity. This might, for example, be services provided by different government departments, or different sites in a company's intranet.

An entity that provides the back-end system determining a user's access and credentials is known as the **identity provider (IdP)**. The individual web resources that a user might want to access are referred to as **service providers (SP)**.

Generally one IdP provides the SSO login form and numerous SPs that can all be used after the user has logged into the IdP's login form. However, federated configurations are also common where either single or multiple login forms can enable authentication against multiple IdPs. In this case, there will often be some 'middleware' that mediates the login requests.

Additional configuration

If additional customizations to the configuration are needed, *download the default spring-security.xml file for SAML*. This adds all the required filters for the SAML extension to function, but it will need customising for the security configuration the application is to authenticate against.

Additional parameters

Some additional parameters can be used in `saml.conf` to change the behavior of the SAML module:

`metadata-display`: set to 'metadataDisplayFilter' and you will be able to download an SP metadata XML file at `http://your-twigkit-host/your-twigkit-app/saml/metadata`. This is useful for debugging, but should be removed for production deployments because it exposes information about the authentication protocol.

`saml-keystore-file`: location of the keystore file to use for checking trust and identity of the IdP and SP security certificates

`key-alias`: the alias of a key in the keystore to use as the default SP security key

`key-password`: password for the keystore and the key defined under using 'key-alias' (must match)

`max-authentication-age`: a TTL for the authentication in seconds, useful if the authentication expires at the IdP end prior to the default which is 7200 seconds.

`response-skew`: a time window in seconds to accept initial SAML responses from the IdP. This is useful if the IdP server and application server are on different timezones. Default: 60 seconds

Configure for a load balancer

When using a load balancer or reverse-proxy in front of an App Studio web app, you must *download a modified spring-security.xml file*. You can then add these additional attributes to `saml.conf` with the respective connection values for your load balancer:

`server-scheme`: The load balancer's communication protocol (for example, http)

`server-name`: The load balancer's host name (for example, search.twigkit.com)

`server-port`: The load balancer's port number

`entity-base-url`: A full URL for the load balancer (for example,
`http://search.twigkit.com` (*`http://search.twigkit.com`*))

Troubleshooting SAML

The SAML authentication standard is a broad XML specification covering a range of functionality and 'binding' protocols. As such, it is not uncommon to encounter problems integrating applications (Service Provider's) with SAML Identity Providers. Numerous technologies can form the SSO back-end (Identity Provider) and there is no 'one-size-fits-all' approach.

If you have reviewed the main SAML documentation for integration in Appkit and are encountering problems, here we document tips and tricks picked up from previous installations.

Because Appkit uses the Spring Security SAML (<http://docs.spring.io/spring-security-saml/docs/1.0.x/reference/htmlsingle/>) extension for underlying SAML compliant client behavior, a useful reference is the Spring documentation (<http://docs.spring.io/spring-security-saml/docs/1.0.x/reference/htmlsingle/>).

Debugging

To enable detailed debugging, add this to your `logback.xml` file:

```
<logger name="org.opensaml" level="TRACE"/> <logger  
name="org.springframework.security" level="TRACE"/>
```

You will now see the content of any SAML request and response messages, as well as a lot of extra debugging information. Also, if you suspect a certificate or SSL related error, you can start the JVM/Tomcat with this argument:

```
-Djava.net.debug=ssl
```

IdP rejects an authentication request (AuthnRequest) from app

A range of issues could cause this. Most commonly, it is because the request from the application failed validation which is often due to the IdP lacking the correct configuration. You must ensure that the endpoints and application identifiers match in the IdP's configuration for the application. For example if using ADFS, this is the 'Relying Party Trust' configuration. If manually configuring this is proves problematic, you can ensure that the IdP has the correct configuration for the app by generating an SP 'metadata' XML file that can be imported into the IdP.

To enable this you must:

1. Add this to `spring-security.xml` in the filter chain:

```
<filter-chain pattern="/saml/metadata/**" filters="metadataDisplayFilter"/>
```

For example:

```
...
<filter-chain pattern="/saml/logout/**" filters="samlLogoutFilter"/>
<filter-chain pattern="/saml/metadata/**" filters="metadataDisplayFilter"/>
<filter-chain pattern="/saml/SSO/**" filters="samlWebSSOProcessingFilter"/>
...
```

2. Add the bean definition for this filter:

```
<!-- The filter is waiting for connections on URL suffixed with filterSuffix
and presents SP metadata there -->
<beans:bean id="metadataDisplayFilter"
class="org.springframework.security.saml.metadata.MetadataDisplayFilter"/>
```

3. Deploy the application and browse to `http://your-app-server/saml/metadata/`.

This should generate the SP metadata XML file describing the applications configuration which can then be imported to almost any compliant IdP in the administration console/configuration.

After the issue is resolved in production, we recommend that you remove this filter because it exposes information about the authentication mechanism.

IdP rejects authentication or sends back a status code in response message indicating an error

It is worth inspecting the status code in the response message and the logs from the IdP as a next step. This issue is often found if the configuration matches at both the IdP and SP, but there is a 'trust' issue, which can be related to the signing certificates used. We have found some IdPs (ADFS for example) that reject self-signed certificates from the application. It is recommended to use a genuine trusted certificate to avoid this problem.

Illegal Key Size error

You might find you get an error at some stage that is a `java.security.InvalidKeyException` or similar stating 'Illegal Key Size'. This is because a key has been used with a size larger than that which the JVM accepts by default. This is a known Java issue to do with cryptography restrictions shipped with a standard JVM. The solution is to install the 'unrestricted JCE policy files' in the JVM on which the application is running. You can obtain this from Oracle and it is usually installed in the JRE directory in `lib/security`. Make sure you install the files for the correct version of Java.

Role-based Access

In many cases, you might want to restrict access to your application, or parts of your application, to specific groups of users. To enable this option, you must add the file `conf/security/access.conf` to your application's configuration tree.

Use the following configuration parameters:

```
allow: role-with-access-1,role-with-access-2 deny: role-without-access-1,role-
without-access-2 pattern: regex-of-uris-to-intercept
```

These configuration parameters are:

- `allow`: A comma-separated list of roles that should be granted access. When not specified or set to a wildcard ('*'), Appkit defaults to allowing access to all roles.
- `deny`: A comma-separated list of roles that should be denied access.
- `pattern`: A regular expression defining the pattern of URIs that should be intercepted for checking access. This defaults to `.*` (all paths are checked).

Example

Let us assume you have a user directory containing two user, `user` and `admin`, where only the latter has an `ADMIN` role. For example, you can define this using a simple `spring-security.xml` configuration with a static list of users like so:

```
<authentication-manager> <authentication-provider> <user-service> <user
name="user" password="user" authorities="USER"/> <user name="admin"
password="admin" authorities="USER,ADMIN"/> </user-service> </authentication-
provider> </authentication-manager>
```

To configure role-based access, you add `conf/security/access.conf` to the application with this configuration:

```
allow: ADMIN pattern: (/)|(/twigkit/api/.*)
```

This says that for all requests to `/` (root page) and the API service we apply role-based authorization rules. The former is strictly not necessary, just leads to better UX for users denied access.

To validate this setup, first log in as `user` and get denied access, as shown in the application logs:

```
TRACE t.s.SecurityFilterExecutionModule - Filtering with  
[twigkit.security.filter.RoleBasedAuthorisationFilter] TRACE  
t.s.f.InterceptAuthorisationFilter - Filtering request to / - comparing against  
access pattern (/)|(/twigkit/api/.* ) TRACE t.s.f.RoleBasedAuthorisationFilter -  
User 'user' is DENIED access to protected resource ERROR  
t.s.SecurityFilterExecutionModule - Authorization chain failed - returning 403
```

Subsequently, log in as `admin` and get access to the app, as shown in the logs:

```
TRACE t.s.SecurityFilterExecutionModule - Filtering with  
[twigkit.security.filter.RoleBasedAuthorisationFilter] TRACE  
t.s.f.InterceptAuthorisationFilter - Filtering request to / - comparing against  
access pattern (/)|(/twigkit/api/.* ) TRACE t.s.f.RoleBasedAuthorisationFilter -  
User 'admin' is GRANTED access to protected resource
```

User Web Service

Although it is disabled by default, there is a web service that returns all of the details for the currently logged in user.

Every operation and capability of Appkit can be invoked via REST-ful web service endpoints supporting both XML and JSON responses for asynchronous client-side operations, and integrating with external systems. For example, users can perform secure searches to any platform supported by Appkit.

You can invoke the User web service by adding this parameter to `conf/security.conf`:

```
webservice-enabled: true
```

The User web service can then be accessed via GET request at `/twigkit/api/user` for example:

```
http://my-twigkit-host/my-twigkit/twigkit/api/user
```

See also *How to Import Security Certificates*.

Geospatial Module

The articles in this section describe the Geospatial module:

- *ESRI Web Maps*
The Esri Web Map tag is used within the Geo Map tag to present content from an ArcGIS web map hosted with Esri or on premises.
- *Maps*
The Map tag displays results on a geographical map, abstracting the underlying map provider.

Esri Web Maps

The Esri Web Map tag is used within the Geo Map tag to present content from an ArcGIS web map hosted with Esri or on premises.

Usage

If the web map is hosted with Esri, you can add the web map without a portal URL by specifying the web map ID. (For example, <http://www.arcgis.com/home/webmap/viewer.html?webmap=de87749b005346d59e1cd34275154c96> (<http://www.arcgis.com/home/webmap/viewer.html?webmap=de87749b005346d59e1cd34275154c96>))

```
<geo:map provider="esri" response="response" address-field="customer_address"
label-field="name" height="700px"> <esri:webmap
id="de87749b005346d59e1cd34275154c96"></esri:webmap> </geo:map>
```

If the web map is hosted on your own servers, simply specify the web map ID and the URL of the server; for example:

```
<geo:map provider="esri" response="response" address-field="customer_address"
label-field="name" height="700px"> <esri:webmap
id="de87749b005346d59e1cd34275154c96"
url="http://yourportalurl.com/sharing/rest/content/items"></esri:webmap>
</geo:map>
```

To see all related attributes, see the *esri:webmap* tag documentation.

Maps

The Map tag displays results on a geographical map, abstracting the underlying map provider.

Usage

If the results contain latitude and longitude values as fields you can render the results using the Google Map API with this configuration:

```
<geo:map provider="google" response="response" latitude-field="MT.lat" longitude-
field="MT.lng"></geo:map>
```

If you do not have coordinates in the results, then the Map tag can do a geocoding of addresses and render these on a map:

```
<geo:map provider="bing" response="response" address-field="fulladdress">
</geo:map>
```

To see all related attributes, see the *geo:map* tag documentation.

Media Module

The Media module supports on-the-fly *image transcoding* to mobile formats and resizing including cropping and square cropping. It further supports capturing and transcoding particular pages from multi-page TIFF files and features built in caching. The Media module also supports efficient *exporting search results* to CSV and Excel.

Adding the media module

Add the media module dependency to your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.media</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

Export Services

Exporting results to Excel

The Excel export service is constrained by the fact that the Excel document needs to be created in memory and therefore recommended for relatively small result sets. For larger exports, we recommend using the CSV export service.

Use the `media:excel` tag to generate a link to export results to Excel.

```
<media:excel conf="platforms.gsa.foo" query="query" fetch="10000"
fields="title,foo" header="myTitle,myFoo">Export</media:excel>
```

Exporting results to CSV

The CSV export service is used for writing search results to a file in comma-delimited format (or other delimited formats). This service requires the platform configuration to use and any combination of query parameters you would normally pass to an Appkit search page.

The CSV export service is highly scalable and will retrieve entire indices (if deep offset is supported by the search engine) and spool to files in a scalable manner.

You can generate an export link using the `media:csv` tag:

```
<media:csv conf="platforms.gsa" query="query" delimiter="|" fetch="100000" batch-
size="100" fields="foo,bar,zip">Export to CSV</media:csv>
```

Offline export

When exporting a large number of documents, it can be preferable to run the export process in the background, to avoid browser time-out. We cater for this possibility by running *offline* exports in a separate thread and storing the CSV output in a file in the local file system of the application server (which might be a shared mount when running multiple nodes). To control how export files are generated and stored, you have these configuration options in `services/media/export/offline` in the configuration hierarchy:

- `compress`: Whether or not to compress the final CSV file into a ZIP archive.
- `temp-folder`: Folder in which to store exported files on the application server, whilst streaming results from the platform (and before they are moved to their final destination). This can either be an absolute file path or relative to the application server root. If omitted, OS-specific temporary file folders are used.
- `redirect-path`: Path to redirect user after export has been initiated.
- `folder`: Folder in which to store exported files on the application server. This can either be an absolute file path or relative to the application server root. The path can be written as an expression, with variables that are resolved for the user that is performing the export:
 - `{{id}}`: The ID of the user.
 - `{{name}}`: The user name of the user (if different from the ID).
 - `{{domain}}`: The domain of the user. For example,
`/var/data/export/{{domain}}/{{id}}/`.

When an export has been completed, the system sends out a `twigkit.media.export.ExportCompleted` event. You can write your own event handler like so:

```
import com.google.common.eventbus.Subscribe; public class ExportListener {
@Override public void export(ExportCompleted event) { System.out.println("Done
exporting " + event.getFile().getAbsolutePath() + " for user " +
event.getUser()); // do something specific with this information } }
```

To register an event listener, add this entry to your application's `AppkitModule`:

```
@Override protected void configure() { ... eventBus.register(new
ExportListener()); ... }
```

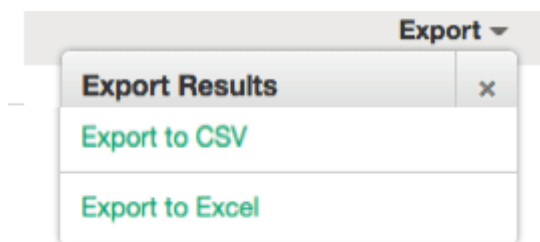
Depending on your business requirements, you might want to email the user a link to the exported CSV, list available CSVs on a home screen, and so forth.

Configuring export links within your search controls

Often it is useful to provide export options alongside your search results. You can achieve this using the above export tags along with some widget popover code:

```
<block:controls> <widget:popover id="export-controls" title="Export results"
link-text="Export" align="right" close-on-click="li"> <ul class="tk-stl-toggle-
menu"> <li> <media:csv conf="platforms.gsa" query="query" delimiter="|"
fetch="100000" batch-size="100" fields="title,foo,bar">Export to CSV</media:csv>
</li> <li> <media:excel conf="platforms.gsa" query="query" fetch="100000"
fields="title,foo" header="myTitle,myFoo">Export to Excel</media:excel></li>
</ul> </widget:popover> </block:controls>
```

This displays the various export tags in a list, which is shown on toggle of the Export drop-down link.



Edit the `link-text="Export"` to change the title of the link, and add further `<li>` elements with links in as required.

Image Formatting and Transcoding

The Image tag displays images from a given field in a result. It will automatically resize, and cache the image for performance.

Usage

This example shows how to display an image from a field:

```
<media:image field-name="image" field-prefix="http://foo.com" validate="true"
width="150"></media:image>
```

In this case, the image URL is retrieved from a field named `image`. Before rendering, the image URL is prefixed with `http://foo.com` (*http://foo.com*) and its existence is validated. It is then resized to 150 pixels wide.

To see all related attributes, see the *media:image* tag documentation.

Workflow and Data Processing Module

A workflow processor is an atomic unit of business logic that can modify a query before it is submitted to a search platform, or transform a search response before rendering. This capability is used extensively for doing runtime data cleaning when, for example, it is not feasible to re-index a whole collection to fix issues in the source data.

The workflow module ships with a number of query and response processors for common tasks out of the box and it is easy to hook these or your own ones into the query/response lifecycle. These include, but are not restricted to:

- **Query pattern matching.** Intercept queries for common terms that have no corresponding documents in the index (for example, postal codes, phone numbers, or email addresses).
- **Natural language processing (NLP).** Analyse free-text input typed in by the end-user to provide a structured, more specific search command to the backend engine. For example, given the query "restaurants in new york" the NLP pre-processor might produce a structured query along the lines of `category:restaurant and city:nyc`. Appkit has integration with a number of third-party NLP parsers, including Expert Systems, WolframAlpha and Smartlogic. More commonly, it is also easy to hook in custom natural language parsers, customized for a particular domain.
- **Data augmentation.** Transform and extend existing data on the fly with re-indexing, for example by fetching external linkages (look up latest stock ticker price, perform relational joins, etc.) or perform translation or lookups (replace ontology terms with common name).
- **Remove facet filters by pattern.** The workflow module includes a comprehensive set of runtime data cleansing tools. When re-indexing is not feasible Appkit can remove irrelevant filters from dynamic navigation options based on for example, regular expression patterns.

The figure below illustrates the query-response lifecycle including a pipeline of workflow processing.

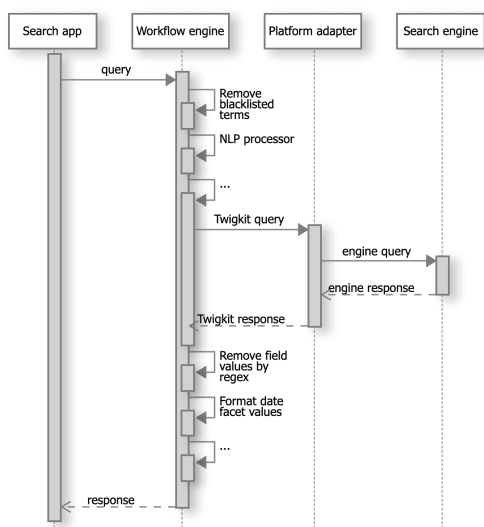


Figure 1. Sample query-response lifecycle involving the workflow engine.

In this example:

1. A search request is pre-processed before it is translated into an engine-specific command and sent to the search engine (for example, by removing specific forbidden or blacklisted words, applying natural language processing to the free-text portion of the query, etc.).
2. The data that comes back from the search engine is translated into a generic Appkit search response by the platform adapter, and then fed into a post-processing pipeline (which might, for example, apply regular expressions to remove specific terms from data fields, apply date formatting to facet aggregation values, etc.).
3. This produces finally a generic Appkit response that is returned to the search application, for rendering or further processing.

Workflow processors: building blocks of data transformation

An Appkit workflow processor is a Java class that is invoked and passed a reference to either a query or response object. A query pre-processor rewrites a query before it is submitted to the underlying search platform. Conversely, a response post-processor transforms a search response (both search results and facets) after it gets returned from a search engine (and before rendering). Workflow processors are typically declared in markup using JSP tags. Multiple processor tags can be specified for sequential processing of queries or responses.

Configuring a workflow pipeline

A workflow pipeline is configured at a platform level. This is useful for example for when you want to reuse processors in several places. This can be done using *configuration files*, chaining processors together along with a platform as a "workflow platform", which then can be referenced like any other platform.

Lucidworks has packaged up a selection of *commonly used response post-processors* for reuse.

Response Processors

The articles in this section describe response processors.

Field Processors

Field response processors include ones to perform these operations:

- Capitalize the display value of the given field names.
- Add Highlighting to Fields.
- Tag a document with classifications based on field values.
- Format a date object, replacing the original date value with another.
- Parse dates out of field values.
- Set the value of a field that is missing a value, based on the value of a different field.
- Parse the String value of the specified fields into an Object representation.
- Create a multivalued field from a single field value by using a separator.
- Extract the hostname from URLs and place it in a field named 'site'.
- Process fully qualified URLs in field values and markup, and add anchor tags for active links in the display values.
- Statically add metadata to documents that match a given regular expression.
- Replace field values (actual, display, or both) that are HTML or URL encoded with decoded values.
- Replace field values (actual, display, or both) that match a given regular expression with a different value.
- Make Twitter users and hashtags clickable in the display value.
- Duplicate a field, creating two separate instances.
- Create a new field by joining multiple existing fields using a pattern expression.
- Localize the values of a field using a specified bundle.

Filter Processors

Filter response processors include ones to perform these operations:

- Capitalize the display value of the given filter.
- Parse dates out of facet filter values.
- Sort facet filters by count.
- Sort facet filters by actual value.
- Replace display values in FacetFilters for specified or all Facets.
- Retrieve other filter values for the same facet, despite there being one or more filters for that facet already applied.
- Remove facet filters that match a regular expression pattern from a facet.
- Replace filter values (actual, display, or both) that match a given regular expression with a different value.
- (Since 4.2.0) Dynamically format the display value of a date range facet according to the size of the range.

Facet Processors

Facet response processors include ones to perform these operations:

- Duplicate a facet.
- Create facets.
- Fill in date filters for a facet.
- Parse and formatting filter values from one Date format to another.
- Create a pseudo date facet.
- Set or changing Parameters for a facet.
- Sort facets based on various properties and parameters.
- Process facet filter values formatted as paths to represent them as hierarchical trees.
- Remove facets from a Response.
- Create pseudo facets based on values in a given field or fields.
- Process hierarchical facets so that only one level is rendered at a time.
- Define a pseudo facet from a list of facets in the response.

General Response Processors

General response processors include ones to perform these operations:

- Perform a left join on fields.
- Log response metadata and attributes to log4j.
- Group results by a given Field value.

Facet Processors

Facet response processors include ones to perform these operations:

- Duplicate a facet.
- Create facets.
- Fill in date filters for a facet.
- Parse and formatting filter values from one Date format to another.
- Create a pseudo date facet.
- Set or changing Parameters for a facet.
- Sort facets based on various properties and parameters.
- Process facet filter values formatted as paths to represent them as hierarchical trees.
- Remove facets from a Response.
- Create pseudo facets based on values in a given field or fields.
- Process hierarchical facets so that only one level is rendered at a time.
- Define a pseudo facet from a list of facets in the response.

For more information about structuring hierarchical facet data in Fusion, see *Indexing Hierarchical Taxonomies*.

twigkit.search.processors.response.CopyFacetProcessor

Duplicate a facet, creating two separate instances.

from (java.lang.String)

Name of facet to copy (clone).

to (java.lang.String)

Name to assign to the new Facet.

twigkit.search.processors.response.CreateFacetProcessor

Create facets by name if they do not exist.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

twigkit.search.processors.response.DateFacetIntervalProcessor

Processor for filling in date filters for a facet, based on a particular step unit (year, month, day). It assumes the named facet has date filters.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

inputFormat (java.lang.String)

Pattern for the original date to be parsed.

outputFormat (java.lang.String)

Format of the output.

step (java.lang.String)

year, month, or day

twigkit.search.processors.response.DateFacetValueFormatter

Parse and format filter values from one Date format to another.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

sourceFormat (java.lang.String)

Pattern for the original date to be parsed.

format (java.lang.String)

Format of the output.

twigkit.search.processors.response.DateRangeFacetProcessor

Create a pseudo date facet.

field (java.lang.String)

The name of the field containing the Date value.

displayName (java.lang.String)

The display name of the Facet generated for field.

twigkit.search.processors.response.FacetParameterProcessor

Set or change Parameters for a facet.

operation (java.lang.String)

Currently only supports 'set'.

parameter (java.lang.String) The name of the parameter to set.

value (java.lang.Object)

The value to set the parameter to.

twigkit.search.processors.response.FacetSortProcessor

Sort facets based on properties and parameters.

sort (java.lang.String)

Sort string as per Query parameter (-field1,field2,-field3).

twigkit.search.processors.response.PathHierarchicalProcessor

Process facet filter values formatted as paths to represent them as hierarchical trees, for example, "Europe/Central Europe/Germany/Berlin".

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

levelSeparator (java.lang.String)

The character sequence that represents a level, for example, levelSeparator="/".

After this processor has been applied to a facet, the facet filters are sorted alphabetically by value (rather than by count).

twigkit.search.processors.response.RemoveFacetsProcessor

Remove facets from a Response by name.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

twigkit.search.processors.response.ShallowFacetGenerator

Create pseudo facets based on values in a given field or fields.

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.ShallowHierarchicalFacetProcessor

Process hierarchical facets so that only one level is rendered at a time, with an HTTP request/response between levels.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

twigkit.search.processors.response.SequentialFacetChoiceProcessor

Define a pseudo facet from a list of facets in the response. The facet that gets copied over to the new pseudo facet will be determined by looking at a sequence of facets (specified by the developer) and choosing the first one that does not have any filter

constraints against it. When the current query contains filters against all facets in the sequence, the last facet in the sequence is copied to the pseudo facet.

For example, assume you have fields in your schema for `Region`, `Country`, and `City` that you want to facet on to guide users in their search. These facets forms a natural hierarchy of `Region > Country > City`. Instead of giving the user a choice from all of the three facets at every step (which would show for example, cities from all regions mixed together), you want to present the user with a single `Location` facet that moves from `Region` to `Country` and finally `City` as users make more specific choices.

To illustrate this, assume you have these data values across your corpus of documents:

```
- Region: Europe, America, Asia - Country: France,United Kingdom,Germany, Mexico,
Brazil,Japan,Korea - City:
Paris,Lyon,London,Manchester,Berlin,Hamburg,Mexico,Tijuana,Sao
Paulo,Brazilia,Tokyo,Kyoto,Seoul
```

Configure a `SequentialFacetChoiceProcessor` with these parameters:

```
facetNames: Region,Country,City target: Location
```

In your search application, present the user with a *single* facet – `Location`. Now consider:

- If no `Location` filter is specified, then the `Location` facet will contain `Region` values: Europe, America, Asia.
- If the user chooses `Asia` as a location, then the `Location` facet will contain the follow `Country` option: Japan, Korea.
- Finally, if the user chooses `Japan`, then the `Location` facet will contain these options: Tokyo, Kyoto.

facetSequence (java.lang.String)

Comma-separated list of facets that should be copied to the new target facet.

target (java.lang.String)

Name to assign to the new facet.

separator (java.lang.String)

Character sequence to separate between facet names when assembling a display label for the new target facet. By default this is ` ▸ ` (an arrow pointing to the right).

Field Processors

Field response processors include ones to perform these operations:

- Capitalize the display value of the given field names.
- Add Highlighting to Fields.
- Tag a document with classifications based on field values.
- Format a date object, replacing the original date value with another.
- Parse dates out of field values.
- Set the value of a field that is missing a value, based on the value of a different field.
- Parse the String value of the specified fields into an Object representation.
- Create a multivalued field from a single field value by using a separator.
- Extract the hostname from URLs and place it in a field named 'site'.
- Process fully qualified URLs in field values and markup, and add anchor tags for active links in the display values.
- Statically add metadata to documents that match a given regular expression.
- Replace field values (actual, display, or both) that are HTML or URL encoded with decoded values.
- Replace field values (actual, display, or both) that match a given regular expression with a different value.
- Make Twitter users and hashtags clickable in the display value.
- Duplicate a field, creating two separate instances.
- Create a new field by joining multiple existing fields using a pattern expression.
- Localize the values of a field using a specified bundle.

twigkit.search.processors.response.CapitaliseFieldValuesProcessor

Capitalize the display value of the given field names.

```
name: twigkit.search.processors.response.CapitaliseFieldValuesProcessor fields:
firstName,lastName
```

fields (`java.lang.String`)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.HighlightFieldValuesProcessor

Add Highlighting to Fields. For a more detailed overview, see *the highlighting page*.

twigkit.search.processors.response.FieldEntityExtractor

Tag a document with classifications based on field values. Using the specified fields, look for patterns provided in a properties file and add classifications to a given field if the value matches.

```
name: twigkit.search.processors.response.FieldEntityExtractor fields: issues
classificationField: categorizedIssues bundle: my-issues
```

With:

```
#my-issues_en.properties in /resources/conf Foo Bar Bam
```

This can be used to pull out keywords from a given field into a new field for display. If the Example above had a field named 'issues' with 'The problem is Foo something else' as value, then the FieldEntityExtractor would match on Foo, and stick 'Foo' into a new categorizedIssues field. This can be used to create new metadata fields, or even filter fields with sensitive data, to just pull out what you want to show.

bundle (java.lang.String)

A properties file containing entities to extract and an optional replacement value when found (for example, IBM = International Business Machines where the latter would be stored as the match for IBM). These are expressed as regular expressions (regex) so as well as simple matching based on whether the value was found within a field, entities can be recognized based on a specific pattern within the text of the chosen fields.

fields (java.lang.String)

Comma-separated list of fields that should be used to search for matches in order to classify the document.

classificationField (java.lang.String)

The field in which to store the classification values. The flexible pattern based approach used allows a document to be tagged with multiple classifications if several matches are found.

twigkit.search.processors.response.DatePartExtractor

Format a date object, replacing the original date value with another. *Use on fields that are already date objects.*

```
name: twigkit.search.processors.response.DatePartExtractor fields: issues  
pattern: dd MMM yyyy
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

pattern (java.lang.String)

Format a Date object according to this Date pattern. Use Java's SimpleDateFormat (<http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>) syntax. In the examples, for example, you will see that `pattern="EEE, MMM d, 'yy"` results in `Wed, Jul 4, '01`.

twigkit.search.processors.response.FieldDateParser

Parse dates out of field values. To convert String data into Date objects.

```
name: twigkit.search.processors.response.FieldDateParser fields: issues pattern:  
dd MMM yyyy
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

pattern (java.lang.String)

The pattern to use when parsing the String value to a Date.

twigkit.search.processors.response.FallbackFieldValue

Set the value of a field that is missing a value, based on the value of a different field.

```
name: twigkit.search.processors.response.FallbackFieldValue field: phoneNumber
fallback: mainPhoneNumber pattern: values: display decode: false
```

field (java.lang.String)

Field that should be affected by this processor.

fallback (java.lang.String)

Field to use for the fallback values.

pattern (java.lang.String)

Regex pattern to use to extract a value from the fallback field

values (java.lang.String)

Which forms of the value to check for emptiness – 'display', 'actual', or 'either'

decode (java.lang.Boolean)

Whether to perform URL decoding on the fallback values

twigkit.search.processors.response.FieldValueParser

Parse the String value of the specified fields into an Object representation.

```
name: twigkit.search.processors.response.FieldValueParser fields: amount
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.FieldToMultiValueFieldProcessor

Create a multivalued field from a single field value by using a separator.

```
name: twigkit.search.processors.response.FieldToMultiValueFieldProcessor field:
cities separator: ,
```

field (java.lang.String)

The name of the field containing the value to be split.

separator (java.lang.String)

The separator to use when splitting the value.

twigkit.search.processors.response.HostNameExtractor

Extract the hostname from URLs and place it in a field named 'site'.

```
name: twigkit.search.processors.response.HostNameExtractor
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.LinkMarkupProcessor

Process fully qualified URLs in field's 'actual' value, and markup with anchor tags for active links in display value.

```
name: twigkit.search.processors.response.LinkMarkupProcessor fields: url
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.RegExFieldValueTaggerProcessor

Statically add metadata to documents that match a given regular expression (in one or more fields).

```
name: twigkit.search.processors.response.RegExFieldValueTaggerProcessor fields:
path classificationField: type pattern: [\w]+
```

The example above looks at a field named path for example, "foo/bar/bam", and breaks all words into a multivalued field named type with multiple values foo, bar and bam using a regex pattern.

If you want to change data rather than just use it, use a ReplaceFieldValue processor.

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

classificationField (java.lang.String)

The field that should contain the metadata classificationValue if the pattern matches.

pattern (java.lang.String)

The pattern to match to the values in the fields defined with the fields parameter.

twigkit.search.processors.response.DecodeFieldValueProcessor

Replace field values (actual, display, or both) that are HTML or URL encoded with decoded values. A use case might be to replace the display value of a URL field. Example usage:

```
name: twigkit.search.processors.response.DecodeFieldValueProcessor fields:
url_display encoding: url values: display
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

values (java.lang.String)

Whether to replace 'actual', 'display' or 'both' values.

Default: 'both'

encoding (java.lang.String)

The encoding of the value to be decoded; 'url' or 'html'.

Default: 'url'

twigkit.search.processors.response.ReplaceFieldValue

Replace field values (actual, display, or both) that match a given regular expression with a different value. The replacement value can contain back-references to matches. A common use case for this is to use a CopyFieldProcessor first, then make changes.

Example:

```
name: twigkit.search.processors.response.ReplaceFieldValue fields: folder
replace: ^(.*/).* with: $1
```

The example above strips off a file name from the end of a folder path to leave just the path using capture expressions and back-references regular expressions

(<http://www.regexplanet.com/advanced/java/index.html>).

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

replace (java.lang.String)

The pattern to replace (can contain regular expressions).

with (java.lang.String)

The replacement (can contain backreferences to the regular expression pattern).

values (java.lang.String)

Whether to replace 'actual', 'display' or 'both' values. Default: 'both'

ignoreCase (java.lang.Boolean)

Whether to ignore case during pattern matching. Default: false

twigkit.search.processors.response.TweetMarkupProcessor

Make Twitter users and hashtags clickable in the display value.

```
name: twigkit.search.processors.response.TweetMarkupProcessor fields:
twitter_user
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.CopyFieldProcessor

Duplicate a field, creating two separate instances.

```
name: twigkit.search.processors.response.CopyFieldProcessor from: url to: my_url
```

from (java.lang.String)
Name of field to copy (clone).

to (java.lang.String)
Name to assign to the new field.

twigkit.search.processors.response.ConcatenateFields

Create a new field by joining multiple existing fields using a pattern expression.

expression (java.lang.String)
Concatenated field pattern defined using double curly braces (see below)

target (java.lang.String)
Name of the new field to create in the response

```
name: twigkit.search.processors.response.ConcatenateFields expression:
{{MemberStreet1}} {{MemberStreet2}} {{MemberCity}} {{MemberState}}
{{MemberZipCode}} {{MemberCountry}} target: compositeAddress
```

Or create a new image field that uses a field value as part of an expression:

```
name: twigkit.search.processors.response.ConcatenateFields expression:
http://your/custom/path/{{MemberCity}}.jpg target: image_url
```

Then use a `<media:image>` tag to output the image field in the result output:

```
<media:image field-name="image_url" width="156" height="156" ... >
```

twigkit.search.processors.response.LocaliseFieldValueProcessor

Localize the values of a field using a specified bundle. For example:

```
name: twigkit.search.processors.response.LocaliseFieldValueProcessor bundle:
languages locale: en fields: language
```

As an example, add a file named `languages_en.properties` to your class path (for example, to `src/main/resources`) and containing these key-value pairs (truncated):

```
aa = Afrikaans ab = Abkhaz am = Amharic ar = Arabic az = Azerbaijani ba = Bashkir
be = Belarusian bg = Bulgarian bm = Bamanankan bn = Bengali bo = Tibetan br =
Breton bs = Bosnian ca = Catalan co = Corsican cr = Cree cs = Czech cy = Welsh
etc.
```

fields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

values (java.lang.String)

Whether to replace 'actual', 'display' or 'both' values. Default: `both`

bundle (java.lang.String)

The bundle to use.

locale(java.lang.String)

The locale to use.

Filter Processors

Filter response processors include ones to perform these operations:

- Capitalize the display value of the given filter.
- Parse dates out of facet filter values.
- Sort facet filters by count.
- Sort facet filters by actual value.
- Replace display values in FacetFilters for specified or all Facets.
- Retrieve other filter values for the same facet, despite there being one or more filters for that facet already applied.
- Remove facet filters that match a regular expression pattern from a facet.
- Replace filter values (actual, display, or both) that match a given regular expression with a different value.
- (Since 4.2.0) Dynamically format the display value of a date range facet according to the size of the range.

twigkit.search.processors.response.CapitaliseFilterValuesProcessor

Capitalise the display value of the given filter.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

twigkit.search.processors.response.FacetFilterDateParser

Parse dates out of facet filter values (which converts the underlying value type from String to Date).

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

pattern (java.lang.String)

Date formatting pattern of the String to be parsed for example, dd/MM/yyyy

twigkit.search.processors.response.FacetFilterSortByCount

Sort facet filters by count.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

twigkit.search.processors.response.FacetFilterSortByValue

Sort facet filters by actual value.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

type (java.lang.String)

Sort by number, string or date.

Default: string

direction (java.lang.String)

Either 'ascending' or 'descending'.

twigkit.search.processors.response.FacetFilterValueMappingProcessor

Replace display values in FacetFilters for specified or all Facets. This is a convenient way, for example, to replace display values in facets with boolean values.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

mappings (java.lang.String)

Mappings parameter defines the mappings/replacements that should be done on the FacetFilter display values. For example, mappings="en=English,fr=French" will replace all instances of en and fr in the FacetFilter list with the full English/French variants.

twigkit.search.processors.response.ParallelFilterProcessor

Retrieve other filter values for the same facet, despite there being one or more filters for that facet already applied.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

platform (twigkit.platform.Platform)

The platform instance to use when resubmitting the query with the facet filters removed (so that parallel filter values can be retrieved).

twigkit.search.processors.response.RemoveFacetFiltersByRegex

Remove facet filters that match a regular expression pattern from a facet.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

pattern (java.lang.String)

A regular expression pattern for the filters that should be removed.

twigkit.search.processors.response.ReplaceFilterValue

Replace filter values (actual, display, or both) that match a given regular expression with a different value. The replacement value can contain back references to matches.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

replace (java.lang.String)

The pattern to replace (can contain regular expressions).

with (java.lang.String)

The replacement (can contain backreferences to the regular expression pattern).

values (java.lang.String)

Whether to replace 'actual', 'display' or 'both' values.
Default: 'both'

ignoreCase (java.lang.Boolean)

Whether to ignore case during pattern matching.
Default: false

twigkit.search.processors.response.ApplyDynamicSolrDateFormattingResponseProcessor

Since 4.2.0

Dynamically format the display value of a date range facet according to the size of the range. This assumes the facet was generated by applying the `twigkit.search.processors.query.DynamicSolrDateRangeQueryProcessor` processor.

facetNames (java.lang.String)

Comma-separated list of facets that should be affected by this processor.

day-format (java.lang.String)

The format to apply to day size range filters.
Default: `dd-MM-yyyy`

month-format (java.lang.String)

The format to apply to month size range filters.
Default: `MM-yyyy`

year-format (java.lang.String)

The format to apply to year size range filters.
Default: `yyyy`

Example configuration

```
name:  
twigkit.search.processors.response.ApplyDynamicSolrDateFormattingResponseProcesso  
r facet-names: Modified Date month-format: yyyy-MM day-format: yyyy-MM-dd
```

General Response Processors

General response processors include ones to perform these operations:

- Perform a left join on fields.
- Log response metadata and attributes to log4j.
- Group results by a given Field value.

twigkit.search.processors.response.LeftJoinProcessor

Perform a left join on the specified fields, adding results with the same values in the specified fields as 'related' results. Appkit carries out additional searches against the platform to retrieve each related result.

joinFields (java.lang.String)

Comma-separated list of fields that should be affected by this processor.

twigkit.search.processors.response.ResponseLoggingProcessor

Log Response metadata and attributes to log4j as TRACE.

twigkit.search.processors.response.ResultGroupProcessor

Group results by a given Field value, making group entries related results for the first one.

groupBy (java.lang.String)

The field containing the identifying value to group results by.

Query Processors

twigkit.search.processors.query.AddFilterProcessor

Add a Filter to a Query. By creating or passing in a Filter instance, you can override or set all attributes of the filter that will then be added to the Query.

Parameters

filter (twigkit.model.Filter)

Instance of a filter to add (optional).

field (java.lang.String)

Field name of the filter.

actualValue (java.lang.String)

The actual value of the filter to be applied.

displayValue (java.lang.String)

The display value of the filter to be applied.

hidden (java.lang.Boolean)

Whether the filter should be hidden (visible in breadcrumbs and URLs) or not.

exclude (java.lang.Boolean)

Whether the filter should exclude values in the given field.

twigkit.search.processors.query.CompositeFieldProcessor

Create a pseudo composite field for user fielded search, where a users query expression will be reformatted as an OR query on a set of fields.

Parameters

fieldNames (java.lang.String)

Names of fields to apply query terms to as an OR query.

compositeField (java.lang.String)

Name of the pseudo composite field that the user can invoke via fielded search syntax (compositeField:expression).

twigkit.search.processors.query.ReplaceFilterValue

Replace all Query filter values (actual, display or both) that match a given regular expression with a different value. The replacement value can contain back-references to matches.

Parameters

replace (java.lang.String)

The pattern to replace (can contain regular expressions).

with (java.lang.String)

The replacement (can contain backreferences to the regular expression pattern).

values (java.lang.String)

Whether to replace 'actual', 'display' or 'both' values. Default: 'both'

ignoreCase (java.lang.Boolean)

Whether to ignore case during pattern matching.

Default: false

twigkit.search.processors.query.QueryLoggingProcessor

Logs the full details of each query sent to the platform at TRACE log level.

twigkit.social.processor.AddBookmarkListToQueryProcessor

Query processor for retrieving a list of bookmarks in a query

Parameter

id (java.lang.String)

Required. The field name to use as a filter when filtering the query with the bookmark id.

Example configuration

```
name: twigkit.social.processor.AddBookmarkListToQueryProcessor id: ID
```

There is no reliable way of making the query return nothing if there are no bookmarks in the list, so this logic is best handled in the JSP, testing whether they exist using a conditional test with a `social:getBookmarkCount` function before applying the processor.

twigkit.search.processors.query.DynamicSolrDateRangeQueryProcessor

Since 4.2.0

Query processor to create Solr (and Fusion) query parameters to enable date range faceting with a dynamic bucket size based on the current filters applied to the query. This processor should be used in conjunction with the `twigkit.search.processors.response.ApplyDynamicSolrDateFormattingResponseProcessor` response processor.

Parameters

fields (java.lang.String)

Required

Comma separated list of fields to apply the processor to. If the field is aliased on the platform it must be provided as `field=alias`.

max-data-points (java.lang.int)

Required

The number of date buckets do you want to generate at a maximum. This drives the size of the buckets being requested. For example if the overall interval is 1 year, and this parameter is set to 36, the size of the buckets will be 1 month. If this parameter is set to 500 the size will be 1 day.

dynamic-buckets (java.lang.Boolean)

Optional

Whether or not to apply the `facet.range.gap` Solr query parameter or rely on it being set elsewhere.

Default: `true`

default-start-date (java.lang.Date)

Required+

When no date filters are applied on the query, this will be the lower bound of the first bucket requested to the platform endpoint.

default-end-date (java.lang.Date)

Required

When no date filters are applied on the query, this will be the higher bound of the last bucket requested to the platform endpoint.

default-other (java.lang.String)

Optional

This specified the solr `facet.range.other` parameter. If none is specified this parameter

will not be set on the query. See here

(https://lucene.apache.org/solr/guide/7_2/faceting.html#range-faceting) for the different values this parameter can take.

default-unit (java.lang.String)

Optional

Initial size of the buckets to request when no filters are applied. Default: one year

Example configuration

```
name: twigkit.search.processors.query.DynamicSolrDateRangeQueryProcessor fields:  
last_modified_dateX=Modified Date max-data-points: 36 default-start-date: 2016-  
01-01T00:00:00.000Z default-end-date: 2018-06-01T00:00:00.000Z
```

Social and Collaboration Tools

The social and collaboration tools provided by the Social module extend the search application with the ability to store user-generated information on individual documents and document collections, and to share this information among groups of authenticated users. For example, documents can be tagged or bookmarked, and users can save search queries and add comments to individual documents.

Features include:

- **Saved searches.** Users can save important keyword search queries, apply filters, and retrieve the information later.
- **Saved results.** Users can bookmark results, add them to a basket, and search those saved items.
- **Rating.** Users can choose to rate or "like" results, giving a social signal to their peers. Rating analytics can be used to influence ranking.
- **Tagging.** Users can tag content with keywords, which content owners can use to improve data quality.
- **Annotations.** Users can annotate/comment on results and edit or delete their own comments. Users cannot edit or delete the comments of others.
- **Topics.** Users can create topics and any captured metadata. For example, saved searches can be associated with a topic.

User-generated content can be stored in a relational database or any other kind of data store, using our pluggable architecture.

Authentication and authorization of users is managed via the Security module.

Prerequisite

Prior to using social and collaboration tools, you must *enable them in Fusion and in App Studio*.

Terminology

While much of the language used to describe collaborative tools is carefully chosen to be self explanatory, there are several general concepts which are used throughout and are introduced in this section.

Annotations

Key to the collaboration enabled by inclusion of this module, there are several types of *annotation* which can be applied to documents. These include bookmarks, comments, and likes, which are commonly found in many applications and provide a versatile way for users to contribute and save information during their search.

Topics

A *topic* is a general term for a subject that users are intending to collect information about. This module provides facilities for users to create a named group with which they can associate any number of annotated documents. Topics themselves can be created and selected using the *Topic Form tag*. Many of the collaborative tags included in this module have a *topic* attribute that links the saved information to a specific topic (which can be picked up from the URL). Topics can be defined by the developer as being 'namespaced' so they can be global (available for all users to collaborate on) or local (only viewable by the individual user that created them).

Collections

Some annotations can be categorized by *collection*. This is a mechanism for separating annotations of different types of document. There is a subtle difference between topics and collections, in that collections are usually determined by the categories of information in the data and generally pre-defined by the application developer offering a small number of collections with which to associate annotations. Collections are intended for a finite number of categorical groups whereas any number of topics can be created.

Social and Collaboration tags

Like Documents

The *collaborate:like* tag displays controls for the user to "like" a document. It also displays statistics on the number of times a specified target document has been liked.

Saving Queries

The Appkit social module lets users save and retrieve their search queries. Use the *collaborate:saved-query-list* tag to let users save and retrieve their saved search queries. Alternatively, you can enable social features in the *search:box* to have access to the same form.

Bookmarks

The *collaborate:bookmark* tag displays controls for adding a new bookmark, in addition to statistics on the number of times a specified target document has been bookmarked. This tag is typically used within a Result List tag, to enable the user to bookmark a particular document in the result list.

Bookmark List tag The *collaborate:bookmark-list* tag retrieves and displays all bookmarks that have been created by a particular user, are within a given topic, or are linked to a particular document (or any combination of the three constraints). Alternatively, you can enable social features in the *search:box* to have access to the same form.

Create Topics

The *collaborate:topic-form* tag lets users create named topics with which to associate saved information. Topics can be restricted to the user who creates a topic or global to all users. The *collaborate:topic-list* tag will display these topics.

App Studio Gallery

This gallery provides visual examples of App Studio elements, with code snippets that you can copy, paste, and edit in your own App Studio views.

- *Button*
- *Checkbox*
- *Color*
- *Comment*
- *Field*
- *Form*
- *Grid*
- *Icon*
- *Input*
- *Loader*
- *Message*
- *Pagination*
- *Panel*
- *Progress Bar*
- *Result List*
- *Tab*
- *Table*
- *Tag*
- *Timeline*
- *Tooltip*
- *Typography*

Button

Default

Standard button

Example

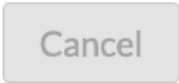


```
<button styling="button">Follow</button>
```

Secondary

Button can be primary and secondary

Example



```
<button styling="button">Okay</button> <button styling="button button-  
secondary">Cancel</button>
```

Icon

Can be only icon

Example



```
<button styling="button"> <widget:icon icon="heart"></widget:icon> </button>
```


Disabled

Example

Disabled

```
<button styling="disabled button"> <i class="icon icon-head"></i> Disabled  
</button>
```

Size

Buttons are in different sizes

Example

Extra Small

Small

Medium

Large

Extra Large

```
<button styling="xs button">Extra Small</button> <button styling="sm  
button">Small</button> <button styling="md button">Medium</button> <button  
styling="lg button">Large</button> <button styling="xl button">Extra  
Large</button>
```

Floated

Buttons can float to left or right

Example

Left Floated

Right Floated

```
<button styling="button pull-right">Right Floated</button> <button  
styling="button pull-left">Left Floated</button>
```

Checkbox

Default

Example



```
<input styling="checkbox-switch" id="switch1" type="checkbox"> <label  
for="switch1"></label>
```

Square

Example



```
<input styling="checkbox-switch" id="switch2" type="checkbox"> <label  
styling="square" for="switch2"></label>
```

Flat

Example



```
<input styling="checkbox-switch" id="switch3" type="checkbox"> <label  
styling="flat" for="switch3"></label>
```

Thin

Example



```
<input styling="checkbox-switch" id="switch4" type="checkbox"> <label  
styling="thin" for="switch4"></label>
```

Colors

Example



```
<input styling="checkbox-switch" id="switch5" type="checkbox"> <label  
styling="positive" for="switch5"></label> <input styling="checkbox-switch"  
id="switch6" type="checkbox"> <label styling="negative" for="switch6"></label>  
<input styling="checkbox-switch" id="switch8" type="checkbox"> <label  
styling="warning" for="switch8"></label> <input styling="checkbox-switch"  
id="switch10" type="checkbox"> <label styling="info" for="switch10"></label>
```

Size

Example



```
<input styling="checkbox-switch" id="switch13" type="checkbox"> <label  
styling="tiny" for="switch13"></label> <input styling="checkbox-switch"  
id="switch14" type="checkbox"> <label styling="massive" for="switch14"></label>
```

Color

All of the colors can be used in the styling attribute: `styling="bg-raspberry text-paper-light"`

Light	Light	Light
Primary	Secondary	Tertiary
Dark	Dark	Dark

<code>bg-primary-light</code>	<code>bg-secondary-light</code>	<code>bg-tertiary-light</code>
<code>bg-primary</code>	<code>bg-secondary</code>	<code>bg-tertiary</code>
<code>bg-primary-dark</code>	<code>bg-secondary-dark</code>	<code>bg-tertiary-dark</code>

Light	Light	Light
Raspberry	Strawberry	Carrot
Dark	Dark	Dark

<code>bg-raspberry-light</code>	<code>bg-strawberry-light</code>	<code>bg-carrot-light</code>
<code>bg-raspberry</code>	<code>bg-strawberry</code>	<code>bg-carrot</code>
<code>bg-raspberry-dark</code>	<code>bg-strawberry-dark</code>	<code>bg-carrot-dark</code>

Light	Light	Light
Pumpkin	Sunflower	Leaf
Dark	Dark	Dark

bg-pumpkin-light	bg-sunflower-light	bg-leaf-light
bg-pumpkin	bg-sunflower	bg-leaf
bg-pumpkin-dark	bg-sunflower-dark	bg-leaf-dark

Light	Light	Light
Grass	Emerald	Mint
Dark	Dark	Dark

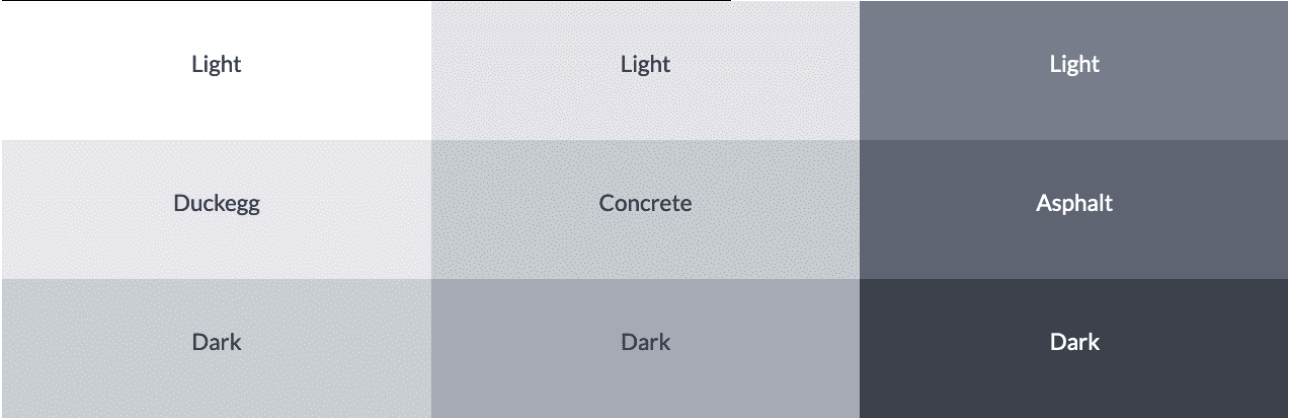
bg-grass-light	bg-emerald-light	bg-mint-light
bg-grass	bg-emerald	bg-mint
bg-grass-dark	bg-emerald-dark	bg-mint-dark

Light	Light	Light
Aqua	Sky	Royal
Dark	Dark	Dark

bg-aqua-light	bg-sky-light	bg-royal-light
bg-aqua	bg-sky	bg-royal
bg-aqua-dark	bg-sky-dark	bg-royal-dark



bg-lavender-light	bg-pink-light	bg-paper-light
bg-lavender	bg-pink	bg-paper
bg-lavender-dark	bg-pink-dark	bg-paper-dark



bg-duckegg-light	bg-concrete-light	bg-asphalt-light
bg-duckegg	bg-concrete	bg-asphalt
bg-duckegg-dark	bg-concrete-dark	bg-asphalt-dark

Comment

Default

Example



Stevie Feliciano 2 days ago 5 Faves

Hey guys, I hope this example comment is helping you read this documentation.

```
<div styling="comment-list"> <div styling="comment"> <a class="avatar">  </a> <div class="content"> <a class="author">Stevie  
Feliciano</a> <div class="metadata"> <div class="date">2 days ago</div> <div  
class="rating"> <i class="star icon"></i> 5 Faves </div> </div> <div  
class="text"> Hey guys, I hope this example comment is helping you read this  
documentation. </div> </div> </div> </div>
```

Comment List

Example



Paul 03/08/2016 at 5:42PM
Who doesn't love Police Academy?!

[Reply](#) [Delete](#)

a.



John 03/08/2016 at 5:42PM
Look at me! I'm nested!

[Reply](#)

a.



Ethan 03/08/2016 at 5:42PM
I'm nested even further.

[Reply](#)



Matt 03/08/2016 at 5:42PM
I have a default avatar and a long message.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Accusamus ad adipisci delectus
dolore, dolorem facere facilis hic impedit iste laboriosam maiores nulla perspiciatis placeat
praesentium qtka qtkas repellat rerum ullam.

[Reply](#)



Steve 03/08/2016 at 5:42PM
Great job!

[Reply](#)

Reply...

Add Comment

```
<social:comment-list target="{string}" platform="{object}" [format="{string}"
]="" [delete-label="{string}" [show="{number}" [show-more="{number}" [show-more-
label="{string}" [show-less-label="{string}" [empty-text="{string}" [default-
delete-message="{string}" [avatars="{string}"]> </social:comment-list>
```


Field

Example

Sir John Carew Eccles

year 1963
Gender Male Age 60 Facial Hair none
Country of birth Australia
Prize Category
medicine

"for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane"

```
<search:result result="response | result:5"> <!-- Title field styling-->
<search:field name="full_name" styling="title"></search:field> <!-- Default -
label left --> <search:field name="year"></search:field> <!-- Inline labels -->
<search:field name="gender" label="Gender" styling="label-inline"></search:field>
<search:field name="age" label="Age" styling="label-inline" show="1">
</search:field> <search:field name="facialhair" label="Facial Hair"
styling="label-inline"></search:field> <br> <!-- Label left that moves on top for
smaller screens --> <search:field name="Country of birth" styling="label-left">
</search:field> <!-- Label is always above the field --> <search:field
name="Prize Category" styling="label-above"></search:field> <!-- Description
styling, ideal for blocks of text that don't require a label --> <search:field
name="motivation" styling="description"></search:field> </search:result>
```

Form

Default Compact Form

Example

A compact inline form (Default)

☐ Remember me

```
<form styling="form"> <fieldset> <legend>A compact inline form (Default)</legend>
<input type="email" placeholder="Email"> <input type="password"
placeholder="Password"> <label for="remember"> <input id="remember"
type="checkbox"> Remember me </label> <button type="submit" styling="button">Sign
in</button> </fieldset> </form>
```

Stacked

Example

A Stacked Form

Email**Password****State**☐ Remember me

```
<form styling="form form-stacked"> <fieldset> <legend>A Stacked Form</legend>
<label for="email">Email</label> <input id="email" type="email"
placeholder="Email"> <label for="password">Password</label> <input id="password"
type="password" placeholder="Password"> <label for="state">State</label> <select
id="state"> <option>AL</option> <option>CA</option> <option>IL</option> </select>
<label for="remember" class="pure-checkbox"> <input id="remember"
type="checkbox"> Remember me </label> <button type="submit" styling="button">Sign
in</button> </fieldset> </form>
```

Aligned Form

Example

Aligned Form

Username	Username
Password	Password
Email Address	<input type="email" value="Email Address"/>
Supercalifragilistic Label	Enter something here...
☐ I've read the terms and conditions	
<input type="submit" value="Submit"/>	

```
<form styling="form form-aligned"> <fieldset> <legend>Aligned Form</legend> <div styling="control-group"> <label for="name">Username</label> <input id="name" type="text" placeholder="Username"> </div> <div styling="control-group"> <label for="password">Password</label> <input id="password" type="password" placeholder="Password"> </div> <div styling="control-group"> <label for="email">Email Address</label> <input id="email" type="email" placeholder="Email Address"> </div> <div styling="control-group"> <label for="foo">Supercalifragilistic Label</label> <input id="foo" type="text" placeholder="Enter something here..."> </div> <div styling="controls"> <label for="cb" styling="checkbox"> <input id="cb" type="checkbox"> I've read the terms and conditions </label> <button type="submit" styling="button pure-button-primary">Submit</button> </div> </fieldset> </form>
```

Multi Column Form (Using Grid)

Example

First Name	Last Name	E-Mail
City	State	
	AL	
☐ I've read the terms and conditions		
<input type="submit" value="Submit"/>		

```
<form styling="form form-stacked"> <fieldset> <div class="tk-rgrid-g"> <div
class="tk-rgrid-u-1 tk-rgrid-u-md-1-3"> <label for="first-name">First
Name</label> <input id="first-name" class="tk-rgrid-u-23-24" type="text"> </div>
<div class="tk-rgrid-u-1 tk-rgrid-u-md-1-3"> <label for="last-name">Last
Name</label> <input id="last-name" class="tk-rgrid-u-23-24" type="text"> </div>
<div class="tk-rgrid-u-1 tk-rgrid-u-md-1-3"> <label for="email">E-Mail</label>
<input id="email" class="tk-rgrid-u-23-24" type="email" required=""> </div> <div
class="tk-rgrid-u-1 tk-rgrid-u-md-1-3"> <label for="city">City</label> <input
id="city" class="tk-rgrid-u-23-24" type="text"> </div> <div class="tk-rgrid-u-1
tk-rgrid-u-md-1-3"> <label for="state">State</label> <select id="state"
styling="input-1-2"> <option>AL</option> <option>CA</option> <option>IL</option>
</select> </div> </div> <label for="terms" styling="checkbox"> <input id="terms"
type="checkbox"> I've read the terms and conditions </label> <button
type="submit" styling="button">Submit</button> </fieldset> </form>
```

Grouped Inputs

Example


```
<form styling="form"> <fieldset styling="group"> <input type="text"
styling="input-1-2" placeholder="Username"> <input type="text" styling="input-1-
2" placeholder="Password"> <input type="email" styling="input-1-2"
placeholder="Email"> </fieldset> <fieldset styling="group"> <input type="text"
styling="input-1-2" placeholder="A title"> <textarea styling="input-1-2"
placeholder="Textareas work too"></textarea> </fieldset> <button type="submit"
styling="button input-1-2">Sign in</button> </form>
```

Input Sizing

Example

.input-1

input-2-3

input-1-2

input-1-3

input-1-4

```
<form styling="form"> <input styling="input-1" type="text" placeholder="input-1">
<br> <input styling="input-2-3" type="text" placeholder="input-2-3"><br> <input
styling="input-1-2" type="text" placeholder="input-1-2"><br> <input
styling="input-1-3" type="text" placeholder="input-1-3"><br> <input
styling="input-1-4" type="text" placeholder="input-1-4"><br> </form>
```

Rounded Inputs

Example

```
<form styling="form"> <input type="text" styling="input-rounded"> </form>
```

Checkboxes and Radio Buttons

Example

- ☐ Here's option one.
 - ☒ Here's a radio button. You can choose this one..
 - ☐ ..Or this one!
- Checkboxes that can be styled*
- ☐ Some label text

```
<form styling="form"> <label for="option-one" styling="checkbox"> <input
id="option-one" type="checkbox" value=""> Here's option one. </label> <label
for="option-two" styling="radio"> <input id="option-two" type="radio"
name="optionsRadios" value="option1" checked=""> Here's a radio button. You can
choose this one.. </label> <label for="option-three" styling="radio"> <input
id="option-three" type="radio" name="optionsRadios" value="option2"> ..Or this
one! </label> <h4>Checkboxes that can be styled</h4> <input id="check1"
type="checkbox" styling="styled-checkbox"> <label for="check1">Some label
text</label> </form>
```

Labelled Inputs

Example

Left Label

http:// twigkit.com

Right Label

Find something

Search

Both

£ 30

.00

```
<p>Left Label</p> <div styling="labelled-input"> <div class="label grey"> <span
class="content">http://</span></div> <input type="text"
placeholder="twigkit.com"> </div> <hr> <p>Right Label</p> <div styling="labelled-
input label-right input-1-3"> <input type="text" placeholder="Find something">
<div class="label grey"> <button styling="button">Search</button> </div> </div>
<hr> <p>Both</p> <div styling="labelled-input label-right"> <div class="label
basic"> <span class="content">f</span> </div> <input type="text"
placeholder="30"> <div class="label basic"> <span class="content">.00</span>
</div> </div>
```

Grid

Grid System

API	Device width when breakpoint overrides default
default	Sets default layout direction unless overridden by another breakpoint.
@xs-width	width < 35.5 em
@sm-width	600 em $\Leftarrow$ width < 35.5 em
@md-width	35.5 em $\Leftarrow$ width < 48 em
@lg-width	48 em $\Leftarrow$ width < 64 em
@xl-width	width $\geq$ 80 em

Twigkit uses a responsive 24 column grid system. The column widths are specified using a ratio of 24ths.

For example at the attribute `sm="1-3"` to the `layout:block` tag to set the width to 1/3 of the container width on small screens.

By default `layout:blocks` will be 100% width.

Example

1 (Default)

23-24

7-8

5-6

4-5

3-4

2-3

1-2

```
<layout:grid styling="text-white"> <layout:block xs="1"><layout:box styling="bg-raspberry-dark">1 (Default)</layout:box></layout:block> <layout:block xs="23-24">
<layout:box styling="bg-raspberry">23-24</layout:box></layout:block>
<layout:block xs="7-8"><layout:box styling="bg-raspberry-light">7-8</layout:box>
</layout:block> <layout:block xs="5-6"><layout:box styling="bg-strawberry-light">5-6</layout:box></layout:block> <layout:block xs="4-5"><layout:box
styling="bg-strawberry">4-5</layout:box></layout:block> <layout:block xs="3-4">
<layout:box styling="bg-strawberry-dark">3-4</layout:box></layout:block>
<layout:block xs="2-3"><layout:box styling="bg-carrot">2-3</layout:box>
</layout:block> <layout:block xs="1-2"><layout:box styling="bg-carrot-light">1-
2</layout:box></layout:block> </layout:grid>
```

Grid Breakpoints

Widths can be specified for each of the `xs`, `sm`, `md`, `lg`, and `xl` breakpoints individually.

`lg`

Example

Always full width

xl=1-3, md=1-2

xl=1-3, md=1-2

xl=1-3, md=1

`md`

Example

Always full width

xl=1-3, md=1-2

xl=1-3, md=1-2

xl=1-3, md=1

`sm`

Example

Always full width

xl=1-3, md=1-2

xl=1-3, md=1-2

xl=1-3, md=1


```
<layout:grid styling="text-white"> <layout:block xs="1"> <layout:box styling="bg-sunflower">Always full width</layout:box> </layout:block> <layout:grid>
<layout:block xl="1-3" md="1-2"> <layout:box styling="bg-pumpkin-light">xl=1-3, md=1-2</layout:box> </layout:block> <layout:block xl="1-3" md="1-2"> <layout:box styling="bg-pumpkin">xl=1-3, md=1-2</layout:box> </layout:block> <layout:block xl="1-3" md="1"> <layout:box styling="bg-pumpkin-dark">xl=1-3, md=1</layout:box> </layout:block> </layout:grid> </layout:grid>
```

Grid Offsets

An offset attribute can be used to push the `layout:block` to the right, leaving an empty space

Example

Always full width

xl=1-3, md=1-2

xl=1-3 xl-offset=1-3, md=1-2

No offset md and above

Offset at sm and above

```
<layout:grid styling="text-white"> <layout:block xs="1"> <layout:box styling="bg-mint">Always full width</layout:box> </layout:block> <layout:grid> <layout:block xl="1-3" md="1-2"> <layout:box styling="bg-mint-light">xl=1-3, md=1-2</layout:box> </layout:block> <layout:block xl="1-3" md="1-2" xl-offset="1-3"> <layout:box styling="bg-mint">xl=1-3 xl-offset=1-3, md=1-2</layout:box> </layout:block> </layout:grid> <layout:grid> <layout:block xs="1-2" xs-offset="1-4" sm-offset="1-2" md-offset="0"> <layout:box styling="bg-aqua">No offset md and above</layout:box> </layout:block> </layout:grid> <layout:grid> <layout:block xs="1-2" sm-offset="1-4"> <layout:box styling="bg-aqua-dark">Offset at sm and above</layout:box> </layout:block> </layout:grid> </layout:grid>
```

Nested

Standard Button

Example

.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4

.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4

.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4

.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4

```
<layout:grid styling="text-white"> <layout:block xs="1-1" sm="1-2" md="1-4">
<layout:box styling="bg-aqua-light"> .tk-rgrid-u-1 <br> .tk-rgrid-u-sm-1-2 <br>
.tk-rgrid-u-md-1-4 <br> </layout:box> </layout:block> <layout:block xs="1-1"
sm="1-2" md="1-4"> <layout:box styling="bg-sky-light"> .tk-rgrid-u-1 <br> .tk-
rgrid-u-sm-1-2 <br> .tk-rgrid-u-md-1-4 <br> </layout:box> </layout:block>
<layout:block xs="1-1" sm="1-2" md="1-4"> <layout:box styling="bg-sky"> .tk-
rgrid-u-1 <br> .tk-rgrid-u-sm-1-2 <br> .tk-rgrid-u-md-1-4 <br> </layout:box>
</layout:block> <layout:block xs="1-1" sm="1-2" md="1-4"> <layout:box
styling="bg-sky-dark"> .tk-rgrid-u-1 <br> .tk-rgrid-u-sm-1-2 <br> .tk-rgrid-u-md-
1-4 <br> </layout:box> </layout:block> </layout:grid>
```

Image

Standard Button

Example

```
.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4
```

```
.tk-rgrid-u-1
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-1-4
```

```
.tk-rgrid-u-sm-1-2
.tk-rgrid-u-md-2-4
```

```
.tk-rgrid-u-1
.tk-rgrid-u-md-1-2
```

```
.tk-rgrid-u-1
.tk-rgrid-u-md-1-2
```

```
<layout:grid> <layout:block xs="1-1" sm="1-2" md="1-4"> <layout:box styling="bg-
paper"> <p> .tk-rgrid-u-1 <br> .tk-rgrid-u-sm-1-2 <br> .tk-rgrid-u-md-1-4 <br>
</p> </layout:box> </layout:block> <layout:block xs="1-1" sm="1-2" md="1-4">
<layout:box styling="bg-paper-dark"> <p> .tk-rgrid-u-1 <br> .tk-rgrid-u-sm-1-2
<br> .tk-rgrid-u-md-1-4 <br> </p> </layout:box> </layout:block> <layout:block
xs="1-1" sm="1-2" md="2-4"> <layout:box styling="bg-paper"> <p> .tk-rgrid-u-sm-1-
2 <br> .tk-rgrid-u-md-2-4 <br> </p> <layout:grid> <layout:block sm="1-1" md="1-
2"> <layout:box styling="bg-paper-light"> <p> .tk-rgrid-u-1 <br> .tk-rgrid-u-md-
1-2 <br> </p> </layout:box> </layout:block> <layout:block sm="1-1" md="1-2">
<layout:box styling="bg-paper-dark"> <p> .tk-rgrid-u-1 <br> .tk-rgrid-u-md-1-2
<br> </p> </layout:box> </layout:block> </layout:grid> </layout:box>
</layout:block> </layout:grid>
```

Justify Content

Example

Long content

item 1

item2

item 3

```
<layout:grid> <layout:block styling="center-content bg-raspberry text-white"
md="1-2"> <p>Long content</p> <br><br><br><br><br><br></layout:block>
<layout:block md="1-2"> <div styling="stretch-content"> <layout:block
styling="center-content bg-strawberry text-white" md="1-2">item 1</layout:block>
<layout:block styling="center-content bg-carrot text-white" md="1-
2">item2</layout:block> <layout:block styling="center-content bg-pumpkin text-
white full-width">item 3</layout:block> </div> </layout:block> </layout:grid>
```

Hide

You can hide the layout by using hide or show at your desire size point `styling="hide-xs"`

xl

Example

hide @xs-width

hide @sm-width

hide @md-width

hide @lg-width

show @xl-width

md

Example

hide @xs-width

hide @sm-width

hide @lg-width

hide @xl-width

show @md-
width

xs

Example

hide @sm-width

hide @md-width

hide @lg-width

hide @xl-width

show @xs-width

```

<layout:grid> <layout:block md="1-5" styling="hide-xs bg-concrete-light">
<layout:box>hide @xs-width</layout:box> </layout:block> <layout:block md="1-5"
styling="hide-sm bg-concrete"> <layout:box>hide @sm-width</layout:box>
</layout:block> <layout:block md="1-5" styling="hide-md bg-concrete-dark">
<layout:box>hide @md-width</layout:box> </layout:block> <layout:block md="1-5"
styling="hide-lg bg-asphalt-light"> <layout:box>hide @lg-width</layout:box>
</layout:block> <layout:block md="1-5" styling="hide-xl bg-asphalt">
<layout:box>hide @xl-width</layout:box> </layout:block> </layout:grid>
<layout:grid> <layout:block md="1-5" styling="show-xs bg-pink-light">
<layout:box>show @xs-width</layout:box> </layout:block> <layout:block md="1-5"
styling="show-sm bg-pink"> <layout:box>show @sm-width</layout:box>
</layout:block> <layout:block md="1-5" styling="show-md bg-pink-dark">
<layout:box>show @md-width</layout:box> </layout:block> <layout:block md="1-5"
styling="show-lg bg-lavender"> <layout:box>show @lg-width</layout:box>
</layout:block> <layout:block md="1-5" styling="show-xl bg-lavender-dark">
<layout:box>show @xl-width</layout:box> </layout:block> </layout:grid>

```

Alignments

Default (Stretch)

A horizontal row of six boxes. The first five are labeled 'Child' and are of equal height. The sixth is labeled 'Extra Long Child Content' and is taller than the others, with its text wrapped over three lines.

Align All Items Top (`.tk-rgrid-align-top`)

A horizontal row of six boxes. The first five are labeled 'Child' and are of equal height. The sixth is labeled 'Extra Long Child Content' and is taller. All boxes are aligned to the top edge.

Align All Items Bottom (`.tk-rgrid-align-bottom`)

A horizontal row of six boxes. The first five are labeled 'Child' and are of equal height. The sixth is labeled 'Extra Long Child Content' and is taller. All boxes are aligned to the bottom edge.

Align All Items Middle (`.tk-rgrid-align-middle`)

A horizontal row of six boxes. The first five are labeled 'Child' and are of equal height. The sixth is labeled 'Extra Long Child Content' and is taller. All boxes are aligned to the middle edge.

Stretch All Items (`.tk-rgrid-align-stretch`)

A horizontal row of nine boxes. The first eight are labeled 'Child' and are of equal height. The ninth is labeled 'Extra Long Child Content' and is taller, with its text wrapped over three lines.

Item Alignment Overrides

Child

.tk-rgrid-align-self-start

.tk-rgrid-align-self-middle

.tk-rgrid-align-self-end

Extra
Long
Child
Content

Justify Items

Default, Align items to the start (`.tk-rgrid-justify-start`)

Child

Child

Child

Child

Child

Child

Child

Child

Child

Child

Align items to the end of the container (`.tk-rgrid-justify-end`)

Child

Child

Child

Child

Child

Child

Child

Child

Child

Child

Center items (`.tk-rgrid-justify-center`)

Child

Child

Child

Child

Child

Child

Child

Child

Child

Child

Even spacing around items (`.tk-rgrid-justify-around`)

Child

Child

Child

Child

Child

Child

Child

Child

Child

Child

Even spacing between items (`.tk-rgrid-justify-between`)

Child

Child

Child

Child

Child

Child

Child

Child

Child

Child

Combination

Child

Child

.tk-rgrid-align-self-middle

.tk-rgrid-align-self-end

Child

Long
Child
Content

Child

Container Direction

Row – Default (`.tk-rgrid-direction-row`)

1

2

3

4

5

Row Reverse (`.tk-rgrid-direction-row-reverse`)

5

4

3

2

1

Column (`.tk-rgrid-direction-col`)

1

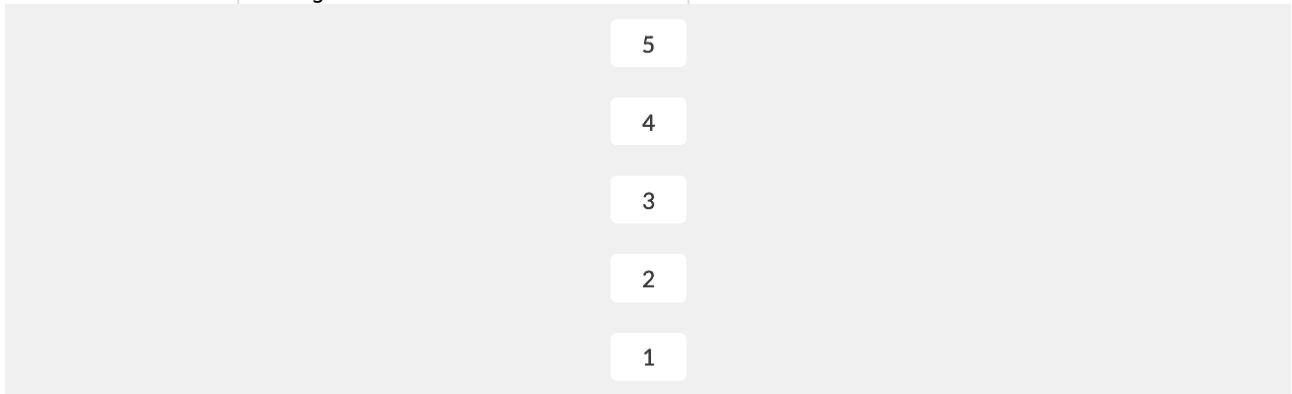
2

3

4

5

Column Reverse (`.tk-rgrid-direction-col-reverse`)



Offsets

`.tk-rgrid-u-1`
`.tk-rgrid-u-sm-1-2`
`.tk-rgrid-u-md-1-4`
`.tk-rgrid-u-offset-sm-1-2`
`.tk-rgrid-u-offset-md-1-4`

`.tk-rgrid-u-1`
`.tk-rgrid-u-sm-1-2`
`.tk-rgrid-u-md-1-4`
`.tk-rgrid-u-offset-sm-1-2`
`.tk-rgrid-u-offset-md-1-4`

Icons

Twigkit provide a handful of useful icons for you to use.

 icon-doc
 icon-doc-text
 icon-doc-inv
 icon-doc-text-inv

 icon-file-pdf
 icon-file-word
 icon-file-excel
 icon-file-powerpoint
 icon-file-image
 icon-file-archive
 icon-file-audio
 icon-file-video
 icon-file-code

 icon-folder-empty
 icon-folder-open-empty
 icon-folder
 icon-folder-open

 icon-user-o
 icon-user
 icon-user-circle
 icon-user-circle-o
 icon-user-times
 icon-user-plus
 icon-address-book-o
 icon-address-book
 icon-address-card-o
 icon-address-card
 icon-id-card-o

 icon-twitter
 icon-facebook
 icon-linkedin
 icon-youtube
 icon-google
 icon-skype
 icon-slack
 icon-whatsapp
 icon-stackoverflow
 icon-twitter-squared
 icon-facebook-squared
 icon-linkedin-squared
 icon-youtube-squared
 icon-gplus-squared

 icon-heart
 icon-heart-empty
 icon-star-empty
 icon-star-half-alt
 icon-star
 icon-bookmark-empty
 icon-bookmark
 icon-thumbs-up
 icon-thumbs-down
 icon-thumbs-up-alt
 icon-thumbs-down-alt
 icon-flag-empty
 icon-flag
 icon-flag-checkered
 icon-flag-1

 icon-tag
 icon-tags

 icon-location
 icon-pin
 icon-lightbulb
 icon-direction

 icon-comment
 icon-chat
 icon-chat-empty

 icon-resize-full
 icon-resize-small
 icon-television
 icon-move
 icon-resize-full-alt
 icon-zoom-in
 icon-zoom-out
 icon-window-maximize
 icon-window-minimize

☒ icon-check
☐ icon-check-empty

 icon-chart-bar
 icon-chart-pie
 icon-chart-area
 icon-chart-line
 icon-chart-line-1

 icon-cog
 icon-share
 icon-link
 icon-link-ext
 icon-attach
 icon-code
 icon-export
 icon-upload
 icon-download
 icon-upload-cloud
 icon-download-cloud
 icon-clone
 icon-lock
 icon-lock-open-alt
 icon-trash
 icon-trash-empty

 icon-info-circled
 icon-help-circled
 icon-ok-circled
 icon-plus-circled
 icon-minus-circled
 icon-cancel-circled
 icon-info
 icon-help
 icon-ok
 icon-plus
 icon-minus
 icon-cancel
 icon-attention

 icon-home
 icon-search
 icon-filter
 icon-sliders
 icon-signal
 icon-ellipsis-vert
 icon-ellipsis
 icon-sort-name-up
 icon-sort-name-down
 icon-sort-number-up
 icon-sort-number-down
 icon-sort-alt-up
 icon-sort-alt-down
 icon-table
 icon-sitemap
 icon-menu
 icon-th-large
 icon-th
 icon-th-list
 icon-align-justify

 icon-angle-up
 icon-angle-down
 icon-angle-left
 icon-angle-right
 icon-up-dir
 icon-down-dir
 icon-left-dir
 icon-right-dir
 icon-up-open-1
 icon-down-open-1
 icon-left-open-1
 icon-right-open-1
 icon-up-open
 icon-down-open
 icon-left-open
 icon-right-open

 icon-up-big
 icon-down-big
 icon-left-big
 icon-right-big

 icon-bell
 icon-bell-alt
 icon-bell-off-empty
 icon-bell-off

 icon-calendar
 icon-clock
 icon-clock-1
 icon-history

 icon-mail
 icon-inbox

 icon-shopping-basket
 icon-box
 icon-sticky-note
 icon-sticky-note-o

 icon-stethoscope
 icon-beaker
 icon-thermometer
 icon-thermometer-3
 icon-thermometer-2
 icon-thermometer-

quarter

 icon-thermometer-0

Input

Standard

Example

Search...

```
<div styling="input"> <input type="text" placeholder="Search..."> </div>
```

Disabled

Example

Search...

```
<input type="text" placeholder="Search..." disabled="">
```

Icon

Example

Search...

```
<div styling="icon input"> <input type="text" placeholder="Search..."> <i  
class="icon-search3 icon"></i> </div>
```

Size

Example

Search extra small...

Search small...

Search medium...

Search large...

Search extra large...

```
<input type="text" placeholder="Search extra small..." styling="xs"> <input  
type="text" placeholder="Search small..." styling="sm"> <input type="text"  
placeholder="Search medium..." styling="md"> <input type="text"  
placeholder="Search large..." styling="lg"> <input type="text"  
placeholder="Search extra large..." styling="xl">
```

Loader

Rotating spinner

Example



```
<widget:spinner></widget:spinner>
```

Message

Interruptor

Example

I'm an Interruptor

Interruptors are smaller than banners, but larger than alertboxes.
They're about drawing your attention to a CTA of some sort.

Button

```
<div styling="interruptor"> <h4>I'm an Interruptor</h4> <p>Interruptors are  
smaller than banners, but larger than alertboxes.<br> They're about drawing your  
attention to a CTA of some sort.</p> <p><a styling="button small  
disabled">Button</a></p> </div>
```

Alert Box

Example

Did you mean the shop [kitchensinks.co.uk](#)?

Something you can probably ignore has happened.

Something bad has happened.

Something slightly worrying has happened.

Something good has happened.

```
<div styling="alertbox"> Did you mean the shop <a href="">kitchensinks.co.uk</a>?
<a href="" class="close">x</a> </div> <div styling="alertbox alertbox-secondary">
Something you can probably ignore has happened. <a href="" class="close">x</a>
</div> <div styling="alertbox alertbox-negative"> Something bad has happened. <a
href="" class="close">x</a> </div> <div styling="alertbox alertbox-warning">
Something slightly worrying has happened. <a href="" class="close">x</a> </div>
<div styling="alertbox alertbox-positive"> Something good has happened. <a
href="" class="close">x</a> </div>
```

Pagination

Default

Example

1 2 3 4 5 next

```
<search:pagination response="response" max-pages="5"></search:pagination>
```

Centered

Example

1 2 3 4 5 next

```
<search:pagination response="response" max-pages="5" styling="pagination-wrapper-center"></search:pagination>
```

Right

Example

1 2 3 4 5 next

```
<search:pagination response="response" max-pages="5" styling="pagination-wrapper-right"></search:pagination>
```

Panel

Default

Example

PANEL HEADER

Lorem ipsum dolor sit amet, consectetur adipisicing elit. A blanditiis commodi corporis, culpa ea eius error molestias necessitatibus numquam odit, officia possimus quae quasi quis quisquam quos rerum unde, voluptas.

Panel footer

```
<div styling="panel"> <h3 styling="title">Panel Header</h3> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. A blanditiis commodi corporis, culpa ea eius error molestias necessitatibus numquam odit, officia possimus quae quasi quis quisquam quos rerum unde, voluptas.</p> <br> <div styling="footer"> <p>Panel footer</p> </div> </div>
```

Using a result

Example

MELBOURNE

Sir John Carew Eccles

"for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane"

```
<search:result result="response | result:5"> <layout:grid> <layout:block md="1-2"> <div styling="stretch-content"> <div styling="center aligned panel center-content"> <search:field name="born_city" styling="title"></search:field> <search:field name="full_name" styling="description" value-element="h3"></search:field> <div styling="footer"> <search:field name="motivation" styling="description"></search:field> </div> </div> </div> </layout:block> </layout:grid> </search:result>
```


Progress Bars

Default

Example

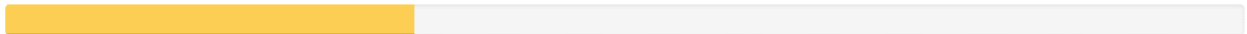


```
<div styling="progress-bar"> <div styling="progress-bar-fill" style="width: 66%">
</div> </div>
```

Background Color

All background colors can be use to change the color of the bar

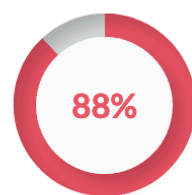
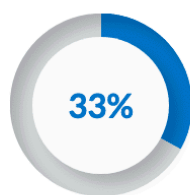
Example



```
<div styling="progress-bar"> <div styling="progress-bar-fill bg-sunflower"
style="width: 33%"></div> </div>
```

Radial

Example



```
<layout:grid> <layout:block sm="1-2"> <progress:radial percentage="33">
</progress:radial> </layout:block> <layout:block sm="1-2"> <progress:radial
percentage="88" fill-class="tk-stl-bg-raspberry" percentage-class="tk-stl-text-
raspberry"></progress:radial> </layout:block> </layout:grid>
```

Result List

Default

The default result list shows all the fields

Example

id	937
firstname	Bob
surname	Dylan
born	1941-05-24T00:00:00
born_lat	46.8
born_lon	-92.1
born_coords	46.8,-92.1
Country of birth	USA
born_countrycode	US
born_city	Duluth, MN
gender	Male
facialhair	moustache
year	2016
Prize Category	literature
share	1
motivation	"for having created new poetic expressions within the great American song tradition"
age	75
	75
type	laureate
award_id	937-2016-literature
prize_id	2016-literature
version	1548651151257239600
score	1
full_name	Bob Dylan
photo_url	assets/avatars/a.jpg
photo	assets/img2.jpg
details_url	/#/laureates/937
affiliation_address	,

```
<search:result-list response="smallResponse"></search:result-list>
```

List

Full width list of results

Example



Bob Dylan

Year 2016 Gender Male Prize Category literature

"for having created new poetic expressions within the great American song tradition"

<http://nobel.divedeeper.io/#/laureates/937> 

```
<search:result-list response="smallResponse"> <search:result> <layout:grid>
<layout:block md="1-4" padding-right="1">  </layout:block> <layout:block md="3-4"> <search:field
name="full_name" styling="title"></search:field> <search:field name="year"
label="Year" styling="label-inline"></search:field> <search:field name="gender"
label="Gender" styling="label-inline"></search:field> <search:field name="Prize
Category" styling="label-inline"></search:field> <br> <search:field
name="motivation" styling="description"></search:field> <search:field
name="details_url" styling="url description" prefix="http://nobel.divedeeper.io"
urlfield="details_url" url-prefix="http://nobel.divedeeper.io"></search:field>
</layout:block> </layout:grid> </search:result> </search:result-list>
```

Cards

Result cards that can be shown in a row

Example



Bob Dylan

Year 2016 Gender Male

Prize Category literature
"for having created new poetic expressions within the great American song tradition"

<http://nobel.divedeeper.io/#/laureates/937>



Karl Ferdinand Braun

Year 1909 Gender Male

Prize Category physics
"in recognition of their contributions to the development of wireless telegraphy"

<http://nobel.divedeeper.io/#/laureates/14>



Luis F. Leloir

Year 1970 Gender Male

Prize Category chemistry
"for his discovery of sugar nucleotides and their role in the biosynthesis of carbohydrates"

<http://nobel.divedeeper.io/#/laureates/239>



Bernardo Alberto Houssay

Year 1947 Gender Male

Prize Category medicine
"for his discovery of the part played by the hormone of the anterior pituitary lobe in the metabolism of sugar"

<http://nobel.divedeeper.io/#/laureates/345>



Sir Frank Macfarlane Burnet

Year 1960 Gender Male

Prize Category medicine
"for discovery of acquired immunological tolerance"

<http://nobel.divedeeper.io/#/laureates/369>



Sir John Carew Eccles

Year 1963 Gender Male

Prize Category medicine
"for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane"

<http://nobel.divedeeper.io/#/laureates/375>



Barry J. Marshall

Year 2005 Gender Male

Prize Category medicine
"for their discovery of the bacterium *Helicobacter pylori* and its role in gastritis and peptic ulcer disease"

<http://nobel.divedeeper.io/#/laureates/789>



J. Robin Warren

Year 2005 Gender Male

Prize Category medicine
"for their discovery of the bacterium *Helicobacter pylori* and its role in gastritis and peptic ulcer disease"

<http://nobel.divedeeper.io/#/laureates/790>



Brian P. Schmidt

Year 2011 Gender Male

Prize Category physics
"for the discovery of the accelerating expansion of the Universe through observations of distant supernovae"

<http://nobel.divedeeper.io/#/laureates/865>



Victor Franz Hess

Year 1936 Gender Male

Prize Category physics
"for his discovery of cosmic radiation"

<http://nobel.divedeeper.io/#/laureates/42>

```
<search:result-list response="response" styling="cards-md-2 cards-xl-3">
<search:result> <layout:grid> <layout:block md="1-4" padding-right="1">  </layout:block> <layout:block
md="3-4"> <search:field name="full_name" styling="title"></search:field>
<search:field name="year" label="Year" styling="label-inline"></search:field>
<search:field name="gender" label="Gender" styling="label-inline"></search:field>
<search:field name="Prize Category" styling="label-inline"></search:field> <br>
<search:field name="motivation" styling="description"></search:field>
<search:field name="details_url" styling="url description"
prefix="http://nobel.divedeeper.io" urlfield="details_url" url-
prefix="http://nobel.divedeeper.io" target="_blank"></search:field>
</layout:block> </layout:grid> </search:result> </search:result-list>
```

Card Emphasis

A result card with a little more border

Example



Bob Dylan

Year 2016 Gender Male Prize

Category literature

"for having created new poetic expressions
within the great American song tradition"

<http://nobel.divedeeper.io/#/laureates/937>

```
<search:result-list response="smallResponse" styling="cards-md-2"> <search:result
styling="card-emphasis"> <layout:grid> <layout:block md="1-4" padding-right="1">
 </layout:block>
<layout:block md="3-4"> <search:field name="full_name" styling="title">
</search:field> <search:field name="year" label="Year" styling="label-inline">
</search:field> <search:field name="gender" label="Gender" styling="label-
inline"></search:field> <search:field name="Prize Category" styling="label-
inline"></search:field> <br> <search:field name="motivation"
styling="description"></search:field> <search:field name="details_url"
styling="url description" prefix="http://nobel.divedeeper.io"
urlfield="details_url" url-prefix="http://nobel.divedeeper.io" target="_blank">
</search:field> </layout:block> </layout:grid> </search:result> </search:result-
list>
```

Card Shadow

Add a small drop shadow

Example



Bob Dylan

Year 2016 Gender Male Prize

Category literature

"for having created new poetic expressions within the great American song tradition"

<http://nobel.divedeeper.io/#/laureates/937> 

```
<search:result-list response="smallResponse" styling="cards-md-2"> <search:result
styling="card-shadow"> <layout:grid> <layout:block md="1-4" padding-right="1">
 </layout:block>
<layout:block md="3-4"> <search:field name="full_name" styling="title">
</search:field> <search:field name="year" label="Year" styling="label-inline">
</search:field> <search:field name="gender" label="Gender" styling="label-
inline"></search:field> <search:field name="Prize Category" styling="label-
inline"></search:field> <br> <search:field name="motivation"
styling="description"></search:field> <search:field name="details_url"
styling="url description" prefix="http://nobel.divedeeper.io"
urlfield="details_url" url-prefix="http://nobel.divedeeper.io" target="_blank">
</search:field> </layout:block> </layout:grid> </search:result> </search:result-
list>
```

Card Step

A result card with a slightly raised edge

Example



Bob Dylan

Year 2016 Gender Male Prize

Category literature

"for having created new poetic expressions within the great American song tradition"

<http://nobel.divedeeper.io/#/laureates/937> 

```

<search:result-list response="smallResponse" styling="cards-md-2"> <search:result
styling="card-step"> <layout:grid> <layout:block md="1-4" padding-right="1">  </layout:block> <layout:block
md="3-4"> <search:field name="full_name" styling="title"></search:field>
<search:field name="year" label="Year" styling="label-inline"></search:field>
<search:field name="gender" label="Gender" styling="label-inline"></search:field>
<search:field name="Prize Category" styling="label-inline"></search:field> <br>
<search:field name="motivation" styling="description"></search:field>
<search:field name="details_url" styling="url description"
prefix="http://nobel.divedeeper.io" urlfield="details_url" url-
prefix="http://nobel.divedeeper.io" target="_blank"></search:field>
</layout:block> </layout:grid> </search:result> </search:result-list>

```

Table

Comparison Tables

Different options for displaying tables on small screens. In terms of CSS, examples 1 and 2 are identical. Example 3 stacks, so is perhaps unsuitable for comparisons. Example 4 is more of a hybrid that is in line with our original sketches. Examples three and four both rely on `data-title` for the labels.

Tested in latest Chrome, FireFox, Safari and Opera. IE10 and later works. IE9 should be possible with some effort.

Tables

Vertical Comparison

	Henson Pictures	Sony Pictures	Warner Bros.	Walt Disney Studios
URL	henson.com	sonypictures.com	warnerbros.com	disney.com
CEO	Brian Henson	Thomas Rothman	Kevin Tsujihara	Alan Horn
Headquarters	Los Angeles, California	Culver City, California	Burbank, California	Burbank, California
Founded	November 20, 1958	August 7, 1991	April 4, 1923	November 05, 1936

Horizontal Comparison

	Henson Pictures	Sony Pictures	Warner Bros.	Walt Disney Studios
URL	henson.com	sonypictures.com	warnerbros.com	disney.com
CEO	Brian Henson	Thomas Rothman	Kevin Tsujihara	Alan Horn
Headquarters	Los Angeles, California	Culver City, California	Burbank, California	Burbank, California
Founded	November 20, 1958	August 7, 1991	April 4, 1923	November 05, 1936

Stacked

Name	URL	CEO	Headquarters	Founded
Henson Pictures	henson.com	Brian Henson	Los Angeles, California	November 20, 1958
Sony Pictures	sonypictures.com	Thomas Rothman	Culver City, California, United States	August 7, 1991
Warner Bros.	warnerbros.com	Kevin Tsujihara	Burbank, California	April 4, 1923
Walt Disney Studios	disney.com	Alan Horn	Burbank, California	November 05, 1936
Universal Studios	universalstudios.com	Ronald Meyer	Universal City, California	April 10, 1912
United Artists	uamg.com	Mark Burnett	Beverly Hills, California	February 5, 1919
Metro-Goldwyn-Mayer	mgm.com	Gary Barber	Los Angeles, California	April 17, 1924

Stacked Comparison

Name	URL	CEO	Headquarters	Founded
Henson Pictures	henson.com	Brian Henson	Los Angeles, California	November 20, 1958
Sony Pictures	sonypictures.com	Thomas Rothman	Culver City, California, United States	August 7, 1991
Warner Bros.	warnerbros.com	Kevin Tsujihara	Burbank, California	April 4, 1923
Walt Disney Studios	disney.com	Alan Horn	Burbank, California	November 05, 1936
Universal Studios	universalstudios.com	Ronald Meyer	Universal City, California	April 10, 1912
United Artists	uamg.com	Mark Burnett	Beverly Hills, California	February 5, 1919
Metro-Goldwyn-Mayer	mgm.com	Gary Barber	Los Angeles, California	April 17, 1924

Combination Comparison

	Henson Pictures	Sony Pictures	Warner Bros.	Walt Disney Studios
URL	henson.com	sonypictures.com	warnerbros.com	disney.com
CEO	Brian Henson	Thomas Rothman	Kevin Tsujihara	Alan Horn
Headquarters	Los Angeles, California	Culver City, California	Burbank, California	Burbank, California
Founded	November 20, 1958	August 7, 1991	April 4, 1923	November 05, 1936

Tabs and Toggle Panes

Default

The standard toggle pane tabs

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="panes"></toggle:controls>
<toggle:panes id="panes"> <toggle:pane> <h2>Pane 1</h2> </toggle:pane>
<toggle:pane> <h2>Pane 2</h2> </toggle:pane> <toggle:pane> <h2>Pane 3</h2>
</toggle:pane> </toggle:panes>
```

Line

Underlined tabs

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="line" styling="line">
</toggle:controls> <toggle:panes id="line"> <toggle:pane> <h2>Pane 1</h2>
</toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane> <toggle:pane>
<h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Pills

Rounded tabs

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="pills" styling="pill">
</toggle:controls> <toggle:panes id="pills"> <toggle:pane visible="true">
<h2>Pane 1</h2> </toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane>
<toggle:pane> <h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Simple

No styling

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="simple" styling="simple">
</toggle:controls> <toggle:panes id="simple"> <toggle:pane visible="true">
<h2>Pane 1</h2> </toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane>
<toggle:pane> <h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Bordered

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:panes id="bordered"> <toggle:pane visible="true"> <h2>Pane 1</h2>
</toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane> <toggle:pane>
<h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Line Bordered

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="line-b" styling="line bordered"></toggle:controls> <toggle:panes id="line-b"> <toggle:pane visible="true"> <h2>Pane 1</h2> </toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane> <toggle:pane> <h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Pills Bordered

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="pills-b" styling="pill bordered"></toggle:controls> <toggle:panes id="pills-b"> <toggle:pane visible="true"> <h2>Pane 1</h2> </toggle:pane> <toggle:pane> <h2>Pane 2</h2> </toggle:pane> <toggle:pane> <h2>Pane 3</h2> </toggle:pane> </toggle:panes>
```

Simple Bordered

Example

Pane 1 Pane 2 Pane 3

Pane 1

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="simple-b" styling="simple bordered"></toggle:controls> <toggle:panes id="simple-b" visible="true">
<toggle:pane visible="true"> <h2>Pane 1</h2> </toggle:pane> <toggle:pane>
<h2>Pane 2</h2> </toggle:pane> <toggle:pane> <h2>Pane 3</h2> </toggle:pane>
</toggle:panes>
```

Tag

Default / Flat

Example

Police

Martial Arts

Comedy

Corruption

Action

```
<ol styling="tag-list tag-capsules"> <li> <a>Police</a> <span class="icon-cross2 icon"></span> </li> <li> <a>Martial Arts</a> </li> <li> <a>Comedy</a> </li> <li>
<a>Corruption</a> </li> <li> <a>Action</a> </li> </ol>
```

Gradient

Example

Police

Martial Arts

Comedy

Corruption

Action

```
<ol styling="tag-list tag-capsules tag-capsules-gradient"> <li> <a>Police</a>
</li> <li> <a>Martial Arts</a> </li> <li> <a>Comedy</a> </li> <li>
<a>Corruption</a> </li> <li> <a>Action</a> </li> </ol>
```

Timeline

Default

Example

October 26, 2016
1:22pm – 1:37pm

Lot Shipped

From: Warehouse 3, Row 6, Level -2
To: XY Distribution Pharma, Texas
Docket Ref: dsg246475g

October 25, 2016
3:22pm – 8:22am

Lot Shipped

From: Warehouse 3, Row 6, Level -2
To: XY Distribution Pharma, Texas
Docket Ref: dsg246475g

```
<layout:grid> <layout:block> <!-- Entry --> <div styling="timeline-entry">
<layout:grid> <layout:block medium="7-24" large="1-3" styling="timeline-time">
<layout:box padding-right="2"> <h2>October 26, 2016</h2> <h4>1:22pm – 1:37pm</h4>
</layout:box> </layout:block> <layout:block medium="17-24" large="14-24"
styling="timeline-content"> <layout:box padding-left="2"> <div styling="panel
text-left"> <h3>Lot Shipped</h3> <p>From: Warehouse 3, Row 6, Level -2</p> <p>To:
XY Distribution Pharma, Texas</p> <p>Docket Ref: dsg246475g</p> </div>
</layout:box> </layout:block> </layout:grid> </div> <!-- Entry --> <div
styling="timeline-entry"> <layout:grid> <layout:block medium="7-24" large="1-3"
styling="timeline-time"> <layout:box padding-right="2"> <h2>October 25, 2016</h2>
<h4>3:22pm – 8:22am</h4> </layout:box> </layout:block> <layout:block medium="17-
24" large="14-24" styling="timeline-content"> <layout:box padding-left="2"> <div
styling="panel text-left"> <h3>Lot Shipped</h3> <p>From: Warehouse 3, Row 6,
Level -2</p> <p>To: XY Distribution Pharma, Texas</p> <p>Docket Ref:
dsg246475g</p> </div> </layout:box> </layout:block> </layout:grid> </div>
</layout:block> </layout:grid>
```

Tooltip

Positions

Tooltip position variations

Example

Top Left

Top

Top Right

Bottom Left

Bottom

Bottom Right

Add users to your feed

Left

Right

```
<button tooltip="Add users to your feed" styling="tooltip-top-left button"> Top Left </button> <button tooltip="Add users to your feed" styling="tooltip-top button"> Top </button> <button tooltip="Add users to your feed" styling="tooltip-top-right button"> Top Right </button> <hr> <button tooltip="Add users to your feed" styling="tooltip-bottom-left button"> Bottom Left </button> <button tooltip="Add users to your feed" styling="tooltip-bottom button"> Bottom </button> <button tooltip="Add users to your feed" styling="tooltip-bottom-right button"> Bottom Right </button> <hr> <button tooltip="Add users to your feed" styling="tooltip-left button"> Left </button> <button tooltip="Add users to your feed" styling="tooltip-right button"> Right </button>
```

Colors

Tooltips can be in different colors

Example

Add users to your feed

Black

Error

Info

Success

Warning

```
<button tooltip="Add users to your feed" styling="tooltip-inverted button"> Black </button> <button tooltip="Add users to your feed" styling="tooltip-error tooltip-top-right button"> Error </button> <button tooltip="Add users to your feed" styling="tooltip-bottom-left tooltip-info button"> Info </button> <button tooltip="Add users to your feed" styling="tooltip-bottom tooltip-success button"> Success </button> <button tooltip="Add users to your feed" styling="tooltip-bottom-right tooltip-warning button"> Warning </button>
```

Typography

Default

Standard Button

Example

H1 - 2.18em

H2 - 1.64em

H3 - 1.45em

H4 - 1.33em

H5 - 1em

H6 - 0.83em

```
<h2>H1 - 2.18em</h2> <h2>H2 - 1.64em</h2> <h3>H3 - 1.45em</h3> <h4>H4 - 1.33em</h4> <h5>H5 - 1em</h5> <h6>H6 - 0.83em</h6>
```

Small – Secondary

Example

App Studio Framework

App Studio Framework

App Studio Framework

App Studio Framework

App Studio Framework

App Studio Framework

```
<h2>App Studio <small>Framework</small></h2> <h2>App Studio <small>Framework</small></h2> <h3>App Studio <small>Framework</small></h3> <h4>App Studio <small>Framework</small></h4> <h5>App Studio <small>Framework</small></h5> <h6>App Studio <small>Framework</small></h6>
```


Paragraph

Example

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting

```
<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s</p>
<p>when an unknown printer took a galley of type and scrambled it to make a type
specimen book. It has survived not only five centuries, but also the leap into
electronic typesetting</p>
```

Highlight

Can be highlighted

Example

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

```
<p>Lorem Ipsum is simply <mark>dummy text</mark> of the printing and typesetting
industry.</p>
```

FAQ

The articles in this section answer frequently asked questions about various aspects and functionality of App Studio.

Aspects of App Studio

- *Devices and Browsers*
- *Performance*
- *Responsive Design*
- *Solr Empty Facets*
- *Space for Log Files*
- *Versions*

How-to

- *Adding Twitter and YouTube Results*
- *Changing the Color Scheme*
- *Changing the Logo*
- *Dynamically Adding Icons To Fields*
- *Removing the Security Module*
- *Setting up a Mock Platform Response*

App Studio Versions

How are these products versioned?

- **Appkit** is versioned according to updates to the code base.
- **App Studio Wizard** is included in the Fusion UI. As a result, the App Studio Wizard version always matches the version of Fusion it is running on.

Starting in Fusion 5, the App Studio Wizard is no longer included in the Fusion UI. However, you can use *App Studio Enterprise* to create apps for Fusion 5.
- **App Studio Enterprise** versions are loosely tied to the version of the Appkit framework it includes. An update from **4.8.0** to **4.9.0** may not include any changes to App Studio Enterprise.

What about compatibility?

- Any version of **Appkit** can be used with Fusion 4.x. However, only Appkit 4.6.0+ can be used with Fusion 5.y.
- **App Studio Wizard** and **App Studio Enterprise** are used alternatively according to availability and preference.
- **App Studio Enterprise** can be used with any version of Fusion or Appkit.

Can these products be updated independently?

- **Appkit** can be *updated* independently, assuming there are no version *compatibility* issues.
- **App Studio Wizard** is included as part of the Fusion UI in Fusion 4.x. It is updated when Fusion is updated.
- **App Studio Enterprise** is released when a new version of Appkit is released. As a result, App Studio Enterprise itself does not need to be updated. To move to the latest code base, simply update the Appkit version within the project.

Devices and Browser Support

What browsers can be supported? For example IE 11 or 9?

Any modern browser will be perfectly fine. For Internet Explorer, we recommend at least version 11. Appkit uses progressive enhancements so for older browsers the application will still work but in a feature limited fashion.

How does AppKit support responsive design?

All standard Appkit templates are responsively designed and will work correctly on tablets and mobiles with full feature parity.

What is the process we must follow to accomplish responsive design?

As long as the existing templates are not compromised, responsive design will work out of the box. The user interface being adapted is a customization of a fully responsive template.

Empty Facets in Solr

How do I avoid displaying empty facet values in my Appkit visualization?

By default Solr will return all the facet values for a given facet field, regardless of whether they match any documents in the result sets.

This behavior is controlled by the Solr facet.mincount configuration parameter (<https://cwiki.apache.org/confluence/display/solr/Faceting#Faceting-Thefacet.mincountParameter>) which defaults to 0.

Setting the facet.mincount parameter to 1 will ensure that only values which match at least one document will be returned in the facet list.

This can be done by adding a custom parameter at query time or by modifying the search handler

(<https://cwiki.apache.org/confluence/display/solr/RequestHandlers+and+SearchComponents+in+SolrConfig#RequestHandlersandSearchComponentsinSolrConfig-SearchHandlers>) in the `solrconfig.xml` file for the relevant collection.

Performance

What kind of performance testing has been done on Appkit and can results be shared?

Appkit is routinely profiled and performance tested. This includes load and capacity testing for concurrent users as well as time taken for requests to be processed. Through years of development and deployments each component will function to a very high performance specification as evidenced by several high load public facing sites including Vodafone, Financial Times, and Thomson–Reuters. Another Appkit customer TES Global (<https://www.tes.co.uk> (<https://www.tes.co.uk>)) experiences peak loads of 100 requests per second.

We track the performance of Appkit in numerous scenarios across releases to ensure there is no degradation in performance. In nearly every installation the performance bottleneck and largest proportion of time taken during the request–response cycle is due to the search platform. Tuning of the search platform and infrastructure, for example horizontal scaling is recommended in large scale deployments with concurrent users.

Is there a process to optimize performance?

The performance of a given application tends to be more bound by the search infrastructure. Appkit is normally only approximately 5–10% of the overall request lifecycle with the majority of the time being spent on I/O within the search engine.

Appkit is a stateless system, so we advocate building out the application as per user requirements and then load testing that to expected loads x 3. Performance will vary more by features selected and infrastructure/network performance, but due to Appkit's stateless nature you can add as many instances as required.

Responsive Design

How does AppKit support responsive design?

All standard Appkit templates are responsively designed and will work correctly on tablets and mobiles with full feature parity.

What is the process we must follow to accomplish responsive design?

As long as the existing templates are not compromised, responsive design will work out of the box. The user interface being adapted is a customization of a fully responsive template.

Space for Log Files

How much space should you allow for Appkit logs?

Less than 50 MB for Appkit logs should be more than sufficient in production, after you reduce the log levels to INFO in `logback.xml`.

However this is entirely configurable. You can set up rolling daily logs, only holding back the last X. See the Logback documentation for more information (<http://logback.qos.ch/manual/appenders.html>).

Deployment

This article describes general app deployment with App Studio.

You can choose how to deploy your search application:

- Deploy your search application in the *Fusion Web Apps service*.

This is the recommended deployment method for production environments.
- Deploy your search application in *standalone mode*, which is best for evaluation, development, demos, and small production environments.

Building from Source

Appkit can either be deployed as a simple web application archive (.war) file – see *Installing Appkit* – or as a Java Web Application built using Maven (<http://maven.apache.org>), as described in this article.

Install Java 7+ JDK, Git and Maven for your platform

1. Download (<https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>) and install the Oracle (SUN) Java SE JDK 8.
2. Install npm 5.0+ (<https://www.npmjs.com/get-npm>).
3. Configure Maven by saving the following `settings.xml` file in your Maven configuration folder within your App Studio Enterprise project (`/bin/settings.xml`). See the Maven Settings Reference (<http://maven.apache.org/settings.html>) for more information.

```
<?xml version="1.0"?> <!-- Credentials for Twigkit repositories --> <settings
xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/profile-1.1.0.xsd"> <servers> <server>
<id>twigkit.com</id> <username>evaluator</username>
<password>AP6YhQQWq5cqu85Eu5JwMznJk48</password> </server> <server>
<id>twigkit.com-releases</id> <username>evaluator</username>
<password>AP6YhQQWq5cqu85Eu5JwMznJk48</password> </server> <server>
<id>twigkit.com-snapshots</id> <username>evaluator</username>
<password>AP6YhQQWq5cqu85Eu5JwMznJk48</password> </server> </servers>
<profiles> <profile> <id>twigkit-default</id> <!-- Artifact repositories -->
<repositories> <repository> <id>twigkit.com</id> <name>Twigkit</name>
<url>https://twigkit.jfrog.io/twigkit/repo</url> <releases> </releases>
<snapshots> </snapshots> </repository> </repositories> <!-- Plugin repositories
--> <pluginRepositories> <pluginRepository> <id>twigkit.com</id>
<name>Twigkit</name> <url>https://twigkit.jfrog.io/twigkit/repo</url>
<releases> <enabled>true</enabled> <updatePolicy>never</updatePolicy>
<checksumPolicy>warn</checksumPolicy> </releases> <snapshots>
<enabled>true</enabled> <updatePolicy>always</updatePolicy>
<checksumPolicy>warn</checksumPolicy> </snapshots> </pluginRepository>
</pluginRepositories> </profile> </profiles> <activeProfiles>
<activeProfile>twigkit-default</activeProfile> </activeProfiles> </settings>
```

Download App Studio Enterprise

Go to the *App Studio Enterprise* download page and download the latest copy.

Run the web application

After you have downloaded the Git project, you can run the application using the included start scripts. Enter this command from the project folder:

```
./app-studio start
```

Or, for Windows:

```
app-studio.bat start
```

After the application has started, you can access it from your browser at `http://localhost:8080/`.

Package the web application

To package the web application for deployment (rather than run it locally), you can run this command from the project folder:

```
./app-studio package
```

Or, for Windows:

```
app-studio.bat package
```

This will generate a web application archive file (.war) in the `target/` folder. You can then deploy the generated package as described in *Deploying an Appkit .war file*.

Appkit also supports creating self-starting applications which bundle the application server. When you package the application a self-starting archive is also created:

In the `/target` folder you will find both the WAR file and the self-starting one, for example, `app-studio-enterprise.jar` (note the .jar suffix). To run the application use:

```
java -jar app-studio-enterprise.jar
```

Use build profiles to package the web application for different runtime environments

Often the only differences between applications deployed to different environments are in the resources, that is, configuration settings, security configuration, or in the license key that is used. To facilitate these differences Appkit also provides the capability to configure which runtime environment you want to deploy to when packaging an application for distribution.

To configure an application for different environments, the resources to be deployed to a particular environment are stored in `src/main/profiles/<environment>` in the source code tree. The assumption being that the main source tree will continue to contain all the resources intended for the `production` build of the application (that is, `src/main/resources`). For example, to configure an application for use in three different environments, in this case `dev`, `staging`, and eventually `production`, the source tree could be set up as follows:

```
/src /main /profiles /dev /conf spring-security.xml twigkit.lic /staging  
twigkit.lic /java /resources << here is where the resources will be stored for  
the production build
```

To package an application with a particular build profile, the profile name is given with the `-P` flag. For example, to deploy to the `dev` environment you would use this command:

```
./app-studio package -P dev
```

Prior to packaging, the contents of the build profile directory are merged with the resources under the main resources directory, overwriting any existing files, and including any additional files. This does not change any of the resources stored under the main resources directory, intended for the `production` build. The profile resources are only merged with main resources in the pre-packaging phase of the build after they are in the `target/classes` directory.

After the application has been packaged, the war and jar files associated with that particular application profile will be stored under the application's `dist/<environment>` directory where `<environment>` is the name given to the build profile. For example:

```
/dist /dev <application-name>.jar <application-name>.war /staging <application-  
name>.jar <application-name>.war <application-name>-1.0-SNAPSHOT.jar << both this  
and the war file below are for the production build <application-name>.war
```

Here, you can see that the application has been packaged for three different environments. One for the `dev` environment, another for the `staging` environment, and finally one for the `production` environment. The `<application-name>` would be replaced by the name of the application. The application version number is appended to the production jar file.

To test a profile locally, you can use its self-starting archive JAR file. To run the application, navigate to the directory the JAR file is in and use this command:

```
java -jar <application-name>.jar
```

Again replacing `<application-name>` with the actual name of the application.

Develop a Maven build profile by overlaying configurations

Together with configuring different build profiles for different runtime environments, Appkit also offers the ability to *overlay configurations* via the Jetty Maven plugin. This can be very useful when you want to test whether specific values can be used in the build profile prior to deployment.

By default, when a build profile is run via the Jetty Maven plugin, a default overlay will also be used if it is placed in `src/dev/resources/conf`:

```
mvn clean jetty:run -Pdevelopment -Dtwigkit.profile=dev
```

However, if the path to the overlay is different it can still be picked up by adding the system parameter `-Dtwigkit.conf.overlay:`

```
mvn clean jetty:run -Pdevelopment -Dtwigkit.profile=dev -  
Dtwigkit.conf.overlay="file:///src/dev/my_resources/conf"
```

In terms of merge order, the build profile is merged first followed by the overlay. So you will see in the final loaded configuration that the overlay takes precedence.

The overlay is purely meant for development. It will not be used when an app is deployed (via `mvn clean package`). During this phase, any resource under the build profile will overwrite the equivalent resource in the main app.

Search Platforms

Appkit integrates with a number of search engines, data warehouses and web service endpoints, even surfacing information from multiple sources simultaneously. The platform abstraction covers both search and storage sides, meaning that Appkit will implement for each platform the necessary protocols to save or index content for any platform. A number of platform adapters are available out-of-the-box with Appkit, and *adding new ones* is a simple exercise.

This is a summary of all types of data source currently supported by Appkit. Each data source has an associated set of connectors, that are responsible for brokering communications between the end-user and the underlying search endpoint. In each case, the corresponding search platform is typically configured through platform configuration files, all of which share a number of *common attributes*.

Search engines

- *Fusion*
- *Solr*
- *SolrCloud*
- *Elasticsearch*

Other data sources

- *Wolfram|Alpha*
- *Wikipedia*
- *Pipl*
- *YouTube*
- *Twitter*
- *Google Custom Search*

Related how-to articles

- *Fusion Impersonation via a Service Account*
- *Set Up Active Directory Federation Services (ADFS) Authentication*
- *Set Up Facebook Authentication*
- *Set Up Google Authentication*
- *Set Up Office 365 Authentication*

Solr

The Solr search adapter brokers all communication with the Solr search engine (<http://lucene.apache.org/solr/>). To work with SolrCloud deployments, read about our *SolrCloud* search adapter.

Setup

To add Solr search connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.solr.runtime</artifactId> <version>${project.parent.version}</version> </dependency>
```

Usage

Assuming you have added the correct dependencies to your project (see above), the Solr platform configuration can be accessed by pointing to it in the `search:platform` tag. In this case the platform is placed in a variable named `platform` using the `var` attribute, and points to a platform configuration in `../conf/platforms/solr/solr.conf`:

```
<search:platform var="platform" conf="platforms.solr"></search:platform>
```

platforms/solr/solr.conf

In this example platform configuration, we set some general settings for the platform:

```
name: twigkit.search.solr.Solr resultIDField: id defaultQuery: *:* url:
http://100.200.300.400:8983/solr/myCollection defaultFacets: age,weight
```

Required attributes

url (java.lang.String)

The URL of the Solr Server. Must be specified using the full URL (for example, `http://localhost:8983/solr/collection` – `localhost:8983` will not work). If a value for `url` is found in the configuration, this will take precedence over other parameters found for `protocol`, `host`, `port`, and `collection`.

protocol (java.lang.String) The protocol used to send requests to the Solr Server, for example, `https` in `https://localhost:8983/solr/collection`.
Default: `http`

host (java.lang.String)

The domain name of the Solr server, for example, `localhost` in `http://localhost:8983/solr/collection`.

[Use of `host` as the URI of the Solr Server is deprecated.]

port (java.lang.String)

The port used to access the Solr Server, for example, `8983` in `https://localhost:8983/solr/collection`.
Default: `8983`

collection (java.lang.String)

The Solr collection being accessed, for example, `collection` in `https://localhost:8983/solr/collection`.

As mentioned above for the `host` attribute, if a non-empty value for `url` is found in the configuration, this will take precedence over other attributes found for `protocol`, `host`, `port`, and `collection`. If no `url` attribute is found, and a value for `host` is found that begins with `http`, `https`, `file` or `/` then this will be taken as the full URL. Finally, if the latter condition is also not met, the URL will be built from attributes `protocol`, `host`, `port`, and `collection` if present.

Optional attributes**requestHandler (java.lang.String)**

Name of the Request Handler to use. Use this to specify an alternate Request Handler to use for example, the `/dismax` request handler. Note you can override this by specifying a `'custom'` attribute on the query.

requestMethod (java.lang.String)

Which HTTP request method to use (GET or POST).

strategy (java.lang.String)

Whether to cluster multiple hosts (value should be either `combine`, or `round-robin`). To enable sharding, set strategy to `combine`.

groupField (java.lang.String)

Setting this enables Solr Result Grouping or Field Collapsing on the given field. This will create a Result for each Group returned. Matching documents for the group will be available as related results.

timeOut (java.lang.Integer)

Time out for platform query requests (in milliseconds).
Default: 5000

ignoreAppliedFilters (java.lang.Boolean)

If this is set to false, facet filters that have already been applied to the Query are added to the list of filters for the facet.
Default: true

backwardsCompatible (java.lang.Boolean)

For Solr 1.x (for example, 1.4) set `backwardsCompatible` to true.
Default: false

highlight (java.lang.Boolean)

Whether to enable hit highlighting (hl parameter) in Solr.
Default: true

Attributes shared across platforms

name (java.lang.String) The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:
`aliases="firstnamelastname=name,countryofresidence=country"`

defaultQuery (java.lang.String)

Default query to use when none is specified. For Solr use `:` to bring back all items (for example, for 'zero term search').

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.
Default: true

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result. Should correspond to the `<uniqueKey>` element in the Solr schema.xml.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Hit highlighting

Hit highlighting is enabled by default, and assumes that the unique `uniqueKey` field in the Solr `schema.xml` file is named `id`. If your `uniqueKey` is set to a different field highlighting will not work unless you set the `resultIDField` key in the Appkit configuration to the right field.

```
resultIDField = yourUniqueKey
```

Appkit expects that Solr's default highlighting tags of `<em></em>` are used. Ensure that this has not been set to something different using `hl.simple.pre` and `hl.simple.post` settings in the Solr or request handler configurations.

Configuring the Hierarchical Facet Service

To correctly display a taxonomy as a hierarchical facet you must configure the requisite Appkit service to retrieve the data structure that is configured in your Solr index.

First you must create a service configuration:

```
/conf /services /hierarchical /hierarchical.conf
```

The file 'hierarchical.conf' should be configured similar to this example:

```
name: twigkit.search.solr.service.facet.GreedySolrHierarchicalFacetService
platform: platforms.solr parents-suffix: _parent
```

platform

This specifies the configured platform where the taxonomy has been indexed.

parents-suffix

The suffix that, if appended to the facet's fieldname, designates the field containing parent node data. The suffix will depend on how the taxonomy data has been indexed.

This service configuration is what is used when adding the hierarchical facet to your HTML page as in this example:

```
<search:facet facet-name="location" show="6" show-more="100" max-characters="35"
collapsible="true"> <facet:hierarchical title="location" facet-name="location"
platform="myHierarchicalPlatform" query="myHierarchicalQuery">
</facet:hierarchical> </search:facet>
```

Indexing Hierarchical Taxonomies

For efficient hierarchical faceting in Solr you should index the nodes in your hierarchy as a multi-valued field using a depth (or level) prefix. This follows Solr best practice for hierarchical faceting, as described here:

<http://wiki.apache.org/solr/HierarchicalFaceting>
(<http://wiki.apache.org/solr/HierarchicalFaceting>)

As an example, suppose you have three documents with these values for the `location` field:

```
Doc#1: Europe > Germany > Berlin Doc#2: Europe > France > Paris Doc#3: North  
America > United States > California > San Francisco
```

These three documents fall into this taxonomy:

```
Europe (Doc#1, Doc#2) | | + France (Doc#1) | | | +- Paris (Doc#1) | +- Germany  
(Doc#2) | +- Berlin (Doc#2) North America (Doc#3) | +- United States (Doc#3) | +  
California (Doc#3) | +- San Francisco (Doc#3)
```

To represent this taxonomy structure in your index, you would add these values for the (multivalued) `location` field:

```
Doc#1: location: 0/Europe 1/Europe/Germany 2/Europe/Germany/Berlin Doc#2:  
location: 0/Europe 1/Europe/France 2/Europe/France/Paris Doc#3: location: 0/North  
America 1/North America/United States 2/North America/United States/California  
3/North America/United States/California/San Francisco
```

The way to read this is that Doc#1 matches the `Europe` level-0 (or root) category *and* the `Europe > Germany` level-1 category, and so forth. Through this indexing scheme, you can get facet counts for all documents that belong to the `Europe > Germany` category (at level 1) by querying for "1/Europe/Germany" as a Solr facet prefix, but equally all documents that as a result belong to `Europe` at a broader level. Finally this allows us to also get facet counts for all documents with any `location` value by requesting "0/" as a facet prefix, for example.

Optimising the retrieval of child nodes

When you display a hierarchical facet in Appkit, Appkit initially only shows the top-level nodes in the taxonomy and asynchronously fetch nodes further down the tree by using Solr facet prefix queries as each node gets expanded. This greatly reduces the size of the taxonomy that would have to be fetched initially, if you had to request the whole tree and render it on the page all at once. This means that, irrespective of how deep and wide your taxonomy is, it can be represented in the user interface in a performant manner.

However, when using only one categorisation field, Appkit does not know whether to show a link to expand each node unless it looks ahead and does another Solr query to check whether a given node has any children. To address this, you can further augment the information Appkit indexes for each document that is tagged with hierarchical categories, so that Appkit can quickly look up both a single tree level (for example, `facet.prefix =`

0/) and also determine which of the nodes at a given level have children (using a single facet query).

More specifically, for each hierarchical facet `my_facet`, you create an additional meta-facet named `my_facet_parents` that contains information about the taxonomy categories that have children (that is, those nodes that are not leaf nodes in the hierarchy). Using the example from above, you would index `location` in exactly the same way as before. In addition, you would index these terms for the new `location_parents` meta-facet:

```
Doc#1: location: 0/Europe 1/Europe/Germany 2/Europe/Germany/Berlin
location_parents: 0/Europe 1/Europe/Germany Doc#2: location: 0/Europe
1/Europe/France 2/Europe/France/Paris location_parents: 0/Europe 1/Europe/France
Doc#3: location: 0/North America 1/North America/United States 2/North
America/United States/California 3/North America/United States/California/San
Francisco location_parents: 0/North America 1/North America/United States 2/North
America/United States/California
```

The new `location_parents` field has all the same index terms as `location` except *the last level is omitted*. The intended meaning is that Doc#1 is indexed with the term `0/Europe` in `location_parents` because it contains `location` values that are the children of Europe, and so forth.

With this scheme in place, Appkit will automatically generate and run a single facet query to retrieve all top-level nodes and identify those nodes that have children.

Query Suggestions

Solr provides the means to suggest documents or terms to the end-user as he is typing search terms in the search box. It can suggest terms based on a dictionary file, terms indexed in a specific field or suggest entire documents based on the occurrence of the search terms in a specific field. For more details on how to implement these features, refer to the suggester (<https://cwiki.apache.org/confluence/display/solr/Suggester>) as well as the terms component (<https://cwiki.apache.org/confluence/display/solr/The+Terms+Component>).

Suggester

As an example here, these are things that you must add to your `solrconfig.xml` file to get suggestions based on matching against a `title` field, as well as looking up terms in your main `content` field.

```
<searchComponent name="suggest" class="solr.SuggestComponent"> <!--Suggestion
based on matches in the title. This will return the entire title field as a
suggestion--> <lst name="suggester"> <str name="name">titleMatchSuggester</str>
<str name="lookupImpl">FuzzyLookupFactory</str> <str
name="dictionaryImpl">DocumentDictionaryFactory</str> <str
name="field">title</str> <str name="suggestAnalyzerFieldType">text_general</str>
<str name="buildOnStartup">true</str> </lst> <!--Suggestion based on keyword
matches in the content field. This will only return the individual words matching
the input string--> <lst name="suggester"> <str
name="name">freeTextContentSuggester</str> <str
name="lookupImpl">FreeTextLookupFactory</str> <str
name="dictionaryImpl">DocumentDictionaryFactory</str> <str
name="field">content</str> <str
name="suggestFreeTextAnalyzerFieldType">text_general</str> <str
name="suggestAnalyzerFieldType">text_general</str> <str
name="buildOnStartup">true</str> </lst> </searchComponent> <requestHandler
class="org.apache.solr.handler.component.SearchHandler" name="/suggest"> <lst
name="defaults"> <str name="suggest">true</str> <str name="suggest.count">5</str>
<str name="suggest.dictionary">titleMatchSuggester</str> <str
name="suggest.dictionary">freeTextContentSuggester</str> </lst> <arr
name="components"> <str>suggest</str> </arr> </requestHandler>
```

After this is set up and you have restarted your Solr engine, the query `http://localhost:8983/solr/sample/suggest?q=al` (replace 'al' with the string the end-user has typed) will bring back suggestions for both suggesters in this format:

```
<response> <lst name="responseHeader"> <int name="status">0</int> <int
name="QTime">3</int> </lst> <lst name="suggest"> <lst name="titleMatchSuggester">
<lst name="al"> <int name="numFound">1</int> <arr name="suggestions"> <lst> <str
name="term">allpicsppt</str> <long name="weight">0</long> <str name="payload"/>
</lst> </arr> </lst> </lst> <lst name="freeTextContentSuggester"> <lst name="al">
<int name="numFound">4</int> <arr name="suggestions"> <lst> <str
name="term">all</str> <long name="weight">14243018808291072</long> <str
name="payload"/> </lst> <lst> <str name="term">alternative</str> <long
name="weight">4069433945226020</long> <str name="payload"/> </lst> <lst> <str
name="term">aligned</str> <long name="weight">2034716972613010</long> <str
name="payload"/> </lst> <lst> <str name="term">alzheimer's</str> <long
name="weight">2034716972613010</long> <str name="payload"/> </lst> </arr> </lst>
</lst> </response>
```

This response basically suggests one document with a title that contains the keyword typed by the end user, as well as suggesting search terms that exist in the index.

Accessing the Solr Suggest component in Appkit

Appkit provides two different services to access Solr suggestions depending on whether or not backwards compatibility (set via the `backwardsCompatible` attribute) is enabled.

Backwards compatibility is disabled

When backwards compatibility is disabled, use the `SolrQuerySuggestionService` to access suggestions. In `resources/conf/services/suggestions`, create a configuration file (for example, `querySuggestions.conf`) to contain the configuration parameters for the suggestion service. Here is an example of how this might look:

```
name: twigkit.search.solr.suggestions.SolrQuerySuggestionService source:
platforms.solr parenthesis: true title: Staff custom: count[5] doctype: reference
```

Both the `name` and `source` attribute are required. The `name` indicates which suggestion service is used. The `source` gives the id for accessing the platform configuration attributes.

The remaining attributes shown here are optional. If `parenthesis` is set to true, the query value, equated with the URL parameter `suggest.q`, will be surrounded by parenthesis. The `title` attribute gives a title to the list of suggestions. Custom parameters can also be set on the query. For example, `count[5]` translates to `count=5` on the query URL.

Backwards compatibility is enabled

When backwards compatibility is enabled, use the `SolrSuggestionsParserService` to access suggestions. In `resources/conf/services/suggestions` create a configuration file (for example, `querySuggestions.conf`) to contain the configuration parameters for the suggestion service. Here is an example of how this might look:

```
name: twigkit.search.solr.suggestions.SolrSuggestionsParserService source:
platforms.fusion suggester: titleMatchSuggester parenthesis: true title: Staff
custom: count[5] doctype: reference
```

As above, both the `name` and `source` attributes are required. The `source` attribute here refers to a Fusion platform configuration, which must have the `backwardsCompatible` attribute set to true. The other required attribute for this service is `suggester`. This is the name of the Solr suggester as defined in the `solrconfig.xml`. For example, using the example above, `suggester` could be set to `titleMatchSuggester` or `freeTextContentSuggester`.

The remaining attributes shown here are optional and are the same as described above for when backwards compatibility is disabled.

Terms component

The Solr terms component (<https://cwiki.apache.org/confluence/display/solr/The+Terms+Component>) "provides access to the indexed terms in a field and the number of documents that match each term". This can be used, for example, to find all terms in the Solr index that match an expansion of the given query term. For example, "car" might return CARRY, CARRIED, CARE, CARL etc.

Accessing the Solr Terms component in Appkit

To access the Terms component in Appkit, use the `SolrQueryTermsService`. In `resources/conf/services/suggestions` create a configuration file (for example,

queryTerms.conf) to contain the configuration parameters for the terms suggestion service. Here is an example of how this might look:

```
name: twigkit.search.solr.suggestions.SolrQueryTermsService source:
platforms.solr.suggestions.terms term-fields: Symbol title: Suggested phrase
doctype: reference
```

Both the `name` and `source` attributes are required as well as the `term-fields` attribute. The `name` indicates which suggestion service is used. The `source` gives the configuration name for accessing the Solr platform's configuration attributes (in the example shown above the platform configuration would be stored in `resources/conf/platforms/solr/suggestions/terms.conf`. Within this platform configuration the `requestHandler` attribute may be defined. However, by default, this is set to `/terms`.

The `term-fields` attribute is equivalent to the `terms.fl` parameter required by the Solr Terms Component. It is a comma-separated list of fields from which to retrieve terms. In the example above, only one field, `Symbol`, is used.

The `title` attribute gives a title to the group of suggestions that are returned by Solr. This title can be used to group suggestions in the UI.

Referring to the suggestion and/or terms service in the search:box

Finally, to ensure your application hooks into the suggestion and/or terms service when it is run, place this line inside your `<search:box>`:

```
<query:suggestions completion-service="services.suggestions.querySuggestions"
action="/search"></query:suggestions> <query:suggestions completion-
service="services.suggestions.queryTerms" action="/search"></query:suggestions>
```

With this in place, when you type a query into the search box, a drop-down list of query suggestions and/or terms will be shown that you can click on to fire a query.

SolrCloud

The SolrCloud search module brokers all communication with a SolrCloud cluster (<https://cwiki.apache.org/confluence/display/solr/SolrCloud>). See also documentation on the Solr[Solr platform] for further background.

Setup

To add SolrCloud search connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.solr.cloud</artifactId> <version>${project.parent.version}
</version> </dependency>
```

Usage

Assuming you have added the correct dependencies to your project (see above), the Solr platform configuration can be accessed by pointing to it in the `search:platform` tag. In this case the platform is placed in a variable named `platform` using the `var` attribute, and points to a platform configuration in `../conf/platforms/solr/solr.conf`:

```
<search:platform var="platform" conf="platforms.solr"></search:platform>
```

platforms/solr/solr.conf

In this example platform configuration, we set some general settings for the platform:

```
name: twigkit.search.solr.cloud.SolrCloud resultIDField: id defaultQuery: *:~
zookeeper-host: cld-dapp-slr61:2181,cld-dapp-slr62:2181,cld-dapp-slr63:2181
defaultFacets: age,weight default-collection: mycollection
```

Required attributes

`zookeeper-host` (`java.lang.String`)

The client endpoint of the zookeeper quorum containing the cloud state. The full specification for this string is one or more comma separated `HOST:PORT` values, followed by an optional `chroot` value that starts with a forward slash. Using a `chroot` allows multiple applications to coexist in one ensemble. For full details, see the ZooKeeper documentation (<https://zookeeper.apache.org/doc/r3.2.2/zookeeperAdmin.html>).

Here are some examples:

- `host1:2181`
- `host1:2181,host2:2181,host3:2181/mysolrchroot`
- `zoo1.example.com:2181,zoo2.example.com:2181,zoo3.example.com:2181`

Optional attributes

`groupField` (`java.lang.String`)

Setting this enables Solr Result Grouping or Field Collapsing on the given field.

backwardsCompatible (java.lang.Boolean)

For Solr 1.x (for example, 1.4) set backwardsCompatible to true.

Default: `false`

ignoreAppliedFilters (java.lang.Boolean)

If this is set to false, facet filters that have already been applied to the Query are added to the list of filters for the facet.

Default: `true`

requestMethod (java.lang.String)

Which HTTP request method to use (GET or POST).

highlighting (java.lang.Boolean)

Whether to enable hit highlighting (hl parameter) in Solr.

Default: `true`

max-query-resend-attempts (java.lang.Integer)

The maximum number of times a query is re-sent after a ZooKeeper session expires.

Default: 1

Attributes shared across platforms**default-collection (java.lang.String)**

Specifies the default Solr collection for the request.

name (java.lang.String)

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstname.lastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstname.lastname=name,countryofresidence=country"`

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result. Should correspond to the `<uniqueKey>` element in the Solr `schema.xml`.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Use of the fields parameter in the Appkit query tag with the SolrCloud platform

By default, on all platforms, if the optional `fields` parameter is not used in the `<search:query>` tag, all fields will be returned in the response. This is not the case however with the SolrCloud platform. On this platform, if the `fields` parameter is omitted no fields will be returned in the response. This is to work around an issue identified with the SolrCloud platform that does not allow fields given by Appkit to be overridden by those defined in an invariants section of the SolrCloud request handler. To return all fields in the case, the fields parameter would must contain a wildcard character, `fields="*"`.

Elasticsearch

Setup

To add Elasticsearch connectors to a Maven project, you must add one of the following dependencies to your project's `pom.xml` file, depending on which version of Elastic you are using:

- For pre-V5 Elasticsearch, add this dependency:

```
<dependency>
  <groupId>twigkit</groupId>
  <artifactId>twigkit.elasticsearch.v2</artifactId>
  <version>${project.parent.version}</version>
</dependency>
```

- For Elasticsearch v5, add this dependency:

```
<dependency>
  <groupId>twigkit</groupId>
  <artifactId>twigkit.elasticsearch.v5</artifactId>
  <version>${project.parent.version}</version>
</dependency>
```

- For Elasticsearch v6, add this dependency:

```
<dependency>
  <groupId>twigkit</groupId>
  <artifactId>twigkit.elasticsearch.v6</artifactId>
  <version>${project.parent.version}</version>
</dependency>
```

For projects that are currently using the non-version-numbered dependency below, we recommend that you update the dependency to one of the version-numbered ones above.

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.elasticsearch</artifactId> <version>${project.parent.version}</version> </dependency>
```

Usage

This example shows how to configure an Elasticsearch platform on a JSP page.

```
<search:platform var="platform" conf="platforms.elastic"></search:platform>
```

This is an example platform configuration that would be at `platforms/elastic/elastic.conf`, with a number of attributes exposed:

```
name: twigkit.search.elasticsearch.ElasticSearch timeout: 30000 resultIDField: id
highlight: true defaultQuery: *: * # Elasticsearch host in the format
my.organisation.com host: localhost # Elasticsearch TCP port (defaults to 9300)
#port: # Elastic cluster cluster: elasticsearch # Elastic index index: test
#Security Settings for Shield #user-name: #password:
```

Required attributes

host (java.lang.String)

The URI of the Elasticsearch Server. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

preserveBoosting (java.lang.Boolean)

If true, ensures the query is 'rewritten' by Elasticsearch so that field boosting is preserved.

Default: `false`

Optional attributes

clientTransportSniff (java.lang.Boolean)

Whether to sniff for other nodes on the same cluster allowing for more fault-tolerant and efficient search. Might be required if the Elasticsearch instances are distributed.

Default: `false`

cluster (java.lang.String)

The name of the cluster to use (required if using `clientTransportSniff="true"` for sniffing nodes on the cluster allowing for more fault-tolerance and efficient searching)

Default: `''`

type (java.lang.String)

The name of the data type to use, which will be defined by a mapping of fields in the Elasticsearch instance

Default: `''`

index (java.lang.String)

The search index, for example, 'search' for an index named search.

Default: `''`

highlight (java.lang.String)

Fields to apply highlighting to. Specified in a comma separated list.

facetSize (java.lang.String)

Determines the total number of all facets to be returned – the Elasticsearch default is 10 which is too small for most applications.

applyFiltersToFacets (java.lang.Boolean)

If true: facets returned will be restricted by the filters applied in the query so will match only those with matching documents in the result set.

Default: `true`

shard-size (java.lang.Integer)

For use with terms aggregations, this parameter decides how many terms the co-ordinating node will request from each shard.

min-doc-count (java.lang.Long)

For use with terms aggregations, terms will only be returned if they match more than this number of hits.

shard-min-doc-count (java.lang.Long)

For use with terms aggregations, terms will only be considered if their local shard frequency is higher than this value.

show-term-doc-count-error (java.lang.Boolean)

For use with terms aggregations, setting this value to true will return an error value for each term, which represents the worst case error in the document count.

fields-type (java.lang.String)

Available with Elastic Search 5 adapter only. Setting this value to `source` will enable the use of source filtering as described here

(<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-source-filtering.html>). Default value is `stored` and uses the legacy `stored_fields` filtering described here (<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-stored-fields.html>).

query-term-fields (java.lang.String)

Available with Elastic Search 5 adapter only. The query can also search against multiple fields. These fields can be provided via the "query-term-fields" parameter here

(https://www.elastic.co/guide/en/elasticsearch/reference/5.2/query-dsl-query-string-query.html#_multi_field). Default value is `[]` that searches against all fields.

Attributes shared across platforms**name (java.lang.String)**

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstnamelastname=name,countryofresidence=country"`

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Security with Elastic Shield

You might use the Elastic Shield plugin to Elasticsearch for securing access to your Elasticsearch infrastructure. You can easily configure the Elasticsearch search platform to handle this.

To add an Elasticsearch search platform that is secured by Elastic Shield

1. Add this dependency to your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.elasticsearch.shield</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Configure the credentials for accessing Elasticsearch secured by the Shield plugin. Specify `user-name` and `password` for the config definition.

```
** `*user-name (java.lang.String)*` + The userName to supply ** `*password
(java.lang.String)*` + The password to supply
```

Security with Elastic X-Pack

☐ This functionality is available in post-v5 Elasticsearch.

You might use Elastic X-Pack for securing access to your Elasticsearch infrastructure. You can easily configure the Elasticsearch search platform to handle this.

To add an Elasticsearch search platform that is secured by Elastic X-Pack

Configure the credentials for accessing Elasticsearch secured by Elastic X-Pack. Specify `user-name` and `password` for the config definition.

- **`user-name (java.lang.String)`**
The userName to supply
- **`password (java.lang.String)`**
The password to supply

Google Custom Search

The Google Custom Search Engine (CSE) platform module brokers all communication with a Google Custom Search Engine (<https://cse.google.co.uk/cse/>).

Setup

To add Google CSE connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.google.cse</artifactId> <version>${project.parent.version}</version> </dependency>
```

Required attributes

apiKey (`java.lang.String`)

The unique ID of the CSE (much like an API key).

Optional attributes

client (`java.lang.String`)

The CSE 'client' query parameter – which front-end to use.

dateFields (`java.lang.String`)

Which fields in the index represent dates. This is a comma separated list.

numericFields (`java.lang.String`)

Which fields in the index represent numbers. This is a comma separated list.

output (`java.lang.String`)

Return output with (xml) or without (xml_no_dtd) Google's DTD.

Default: `xml_no_dtd`

Attributes shared across platforms

host (`java.lang.String`)

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

name (`java.lang.String`)

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (`java.lang.String`)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstnamelastname=name,countryofresidence=country"`

defaultQuery (`java.lang.String`)

Default query to use when none is specified.

defaultFacets (`java.lang.String`)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (`java.lang.Boolean`)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (`java.lang.Boolean`)

Apply advanced linguistics such as stemming or lemmatization if supported by the

Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Lucidworks Fusion

The Fusion search adapter brokers all communication with the Lucidwork's Fusion platform (<https://lucidworks.com/products/fusion/>) using its Apollo REST API to query Solr.

Setup

To add Fusion search connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.fusion</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

Usage

Assuming you have added the correct dependencies to your project (see above), the Fusion platform configuration can be accessed by pointing to it in the `search:platform` tag. In this case the platform is placed in a variable named `platform` using the `var` attribute, and points to a platform configuration in `../conf/platforms/fusion/search.conf`:

```
<search:platform var="platform" conf="platforms.fusion.search"></search:platform>
```

platforms/fusion/search.conf

In this example platform configuration, we set some general settings for the platform:

```
name: twigkit.search.fusion.Fusion timeout: 30000 resultIDField: id highlight:  
true defaultQuery: *: * query-profile: my-profile
```

With a Fusion platform, we also need to set some additional parameters in the following files:

services/api/fusion.conf

This file controls which Fusion server to use:

```
# Hostname of your Fusion server host: myHostname # Port of your Fusion server  
port: 6764 # Protocol of your Fusion server - e.g. http or https protocol: http  
# Optional: Fusion Authorization mode - native, serviceaccount, or jwt. # If not  
configured, this will default to the first configured option in the order JWT >  
Service Account > Native # Native auth requires no config here, but does require  
the twigkit.security.provider.fusion module to be loaded. # auth: native  
# Service Account config userName: admin password: password # JWT config #  
issuer: # key: # groups: # Optional: If a JWT subject is configured, all requests  
to Fusion will be sent as the JWT realm user whose name matches the subject #  
subject:
```

platforms/fusion/social.conf

This file configures the Lucidworks Appkit Social Module:

```
# For every Fusion App, a corresponding YOUR_FUSION_APP_user_prefs collection is
automatically created. # Appkit uses this collection for storing collaboration
user-data. # Name of the collection to store user data in collection: Supp-
Demo_user_prefs # Name of the Query & Index pipelines to use - _system is the
Fusion default # pipeline: _system # Enable writing new Social entities into the
platform readOnly: false # Prevent users from searching this platform directly
webservice-enabled: false
```

message/service/fusion.conf

This file configures how Signals are sent to Fusion:

```
# Fusion Signals config # Name of the Query profile to send Signals through
query-profile: Supp-Demo # The index pipeline to send Signals to # signals-index-
pipeline: _signals_ingest # Whether to commit Signals to disk inside Fusion
immediately or not # commit: true # Whether to index to Fusion asynchronously or
not # async: true
```

Required attributes

query-profile (java.lang.String)

The Fusion query profile being accessed. If you have created an app via the Fusion UI called MyAwesomeApp, then you will have a default Query Profile called MyAwesomeApp as well – this is a good starting point.

Applied in: configuration

Optional attributes

requestHandler (java.lang.String)

Name of the Request Handler to use. Use this to specify an alternate Request Handler to use for example, the '/dismax' request handler. Note you can override this by specifying a 'custom' attribute on the query.

requestMethod (java.lang.String)

Which HTTP request method to use (GET or POST).

strategy (java.lang.String)

Whether to cluster multiple hosts (value should be either combine, or round-robin). To enable sharding, set strategy to combine.

groupField (java.lang.String)

Setting this enables Solr Result Grouping or Field Collapsing on the given field. This will create a Result for each Group returned. Matching documents for the group will be available as related results.

timeOut (java.lang.Integer)

Time out for platform query requests (in milliseconds).

Default: 5000

ignoreAppliedFilters (java.lang.Boolean)

If this is set to false, facet filters that have already been applied to the Query are added to the list of filters for the facet.

Default: true

backwardsCompatible (java.lang.Boolean)

For Solr 1.x (for example, 1.4) set backwardsCompatible to true.

Default: false

highlight (java.lang.Boolean)

Whether to enable hit highlighting (hl parameter) in Solr.

Default: true

Attributes shared across platforms**aliases (java.lang.String)**

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstname.lastname` as `name` and `countryofresidence` as `country`, use these mappings:

```
aliases="firstname.lastname=name,countryofresidence=country"
```

defaultQuery (java.lang.String)

Default query to use when none is specified. For Solr use `:` to bring back all items (for example, for 'zero term search').

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: true

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: true

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: true

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

Applied in: configuration

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result. Should correspond to the `<uniqueKey>` element in the Solr schema.xml.

Applied in: JSP tag, configuration

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Applied in: JSP tag, configuration

Hit highlighting

Hit highlighting is enabled by default, and assumes that the unique `uniqueKey` field in the Solr `schema.xml` file is named `id`. If your `uniqueKey` is set to a different field highlighting will not work unless you set the `resultIDfield` key in the Appkit configuration to the right field.

```
resultIDField = yourUniqueKey
```

Appkit expects that Solr's default highlighting tags of `<em></em>` are used. Ensure that this has not been set to something different using `hl.simple.pre` and `hl.simple.post` settings in the Solr or request handler configurations.

Related articles

- *Fusion Impersonation via a Service Account*
- *Query Terms and Suggestions*

Offline

At times, it is not always possible to have access to your search device. To facilitate this, Appkit offers the ability to record responses and save them to disk when you do have access. Then, you can use those responses to carry on working offline.

Recording responses

By booting your application with the system variable `-Dtwigkit.offline.recording=true`, every search response served by the backend device will be serialized to a JSON file written to disk. By default, the location of the saved response JSON file relative to the application root will be as follows:

```
/offline-data /<platform.id> response_<encoded query url>.json
```

Here, `<platform.id>` is either the configuration name of the platform (for example, `platforms.fusion`) or the platform ID, and `<encoded query url>` is an encoded form of the query URL that was used to generate the response. The default offline root folder, `offline-data`, can be overridden by setting the system variable `-Dtwigkit.offline.recording.path` to a relative or absolute file path.

For example, suppose you submit a query to the `platforms.solr` platform instance, with `-Dtwigkit.offline.recording.path=my-data`. Then the system will save any response serialized as a JSON file to:

```
my-data platforms.solr response_1234567.json
```

If a default query (with no query or filter terms) is run, this response will be written to `response.json` and saved in the appropriate platform folder.

Working offline

The Offline Search platform is a search interface that serves static content by matching search queries with JSON data files read from disk. Each JSON data file contains a serialized Appkit response object, independent of the particular search engine that produced the original search results. See the section above for information on how to capture and store search response data.

To work offline with recorded responses, the first step is to add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>  
<artifactId>twigkit.offline.search</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

After this is done, the next step is to update the configuration file of the platform you had been working online with in order to tell it that you are now working offline. The best way to do this is to use the *configuration overlay*. For example, if you previously recorded a number of responses from a Solr device, you could create or update the configuration file, `solr.conf`, in `src/dev/resources/conf/platforms/solr/` to have these entries:

```
name: twigkit.search.offline.OfflineSearch path: file:///data-files/platforms.solr
```

Here, `path` defines the location of the offline response JSON files. This can either be a relative path (for example, `./data-files`), resolved relative to the directory from which the web server is booted, or an absolute path (for example, `/var/data-files`).

Pipl

The Pipl search platform adds support to Appkit for the Pipl search engine for detailed people results from a range of sources.

Setup

To add Pipl connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.pipl</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

Required attributes

`apiKey (java.lang.String)`

The API Key for programmatic access to Pipl.

Attributes shared across platforms

`host (java.lang.String)`

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

`name (java.lang.String)`

The display name of this Platform instance. Used for identifying different platforms in the user interface.

`aliases (java.lang.String)`

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:
`aliases="firstnamelastname=name,countryofresidence=country"`

`defaultQuery (java.lang.String)`

Default query to use when none is specified.

`defaultFacets (java.lang.String)`

Default facets to request when none are specified. This is a comma separated list.

`spellCheck (java.lang.Boolean)`

Spellcheck the Query term if supported by the Platform.
Default: `true`

`expandQuery (java.lang.Boolean)`

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.
Default: `true`

`autoCorrect (java.lang.Boolean)`

Whether to auto-correct and resubmit futile queries (queries with zero results).
Default: `true`

`fileTypeField (java.lang.String)`

Set which field contains information about file type (mime type).

`resultIDField (java.lang.String)`

Set which field represents the unique identifier for a given result.

`pageLimit (java.lang.Long)`

A limitation to which page the platform will offset.

Shared Platform Attributes

This is a list of shared attributes common to all *search platforms*.

Where applicable, default values are given, for example, the default value for the `spellcheck` attribute is set to `true`. An indication is also given of where these attributes can be applied. For example, some attributes can be applied either in the JSP tag or a configuration file (for example, `gsa.conf`). Other attributes can only be applied in the JSP tag or in the configuration file.

On some platforms, the values following `Applied in:` might not be exactly the same as those shown below. For confirmation, also refer to the list of standard attributes specified on that platform page.

Attributes shared across platforms

host (java.lang.String)

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

name (java.lang.String)

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstname.lastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstname.lastname=name,countryofresidence=country"`

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fields (java.lang.String)

Names of fields to request. This is a comma separated list. Fields set on the query take precedence over fields set here.

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

webservice-enabled (java.lang.String)

Whether the platform web service should be accessible (defaults to 'true'). Set to 'false' if you do not want to expose the web service for the platform to authenticated users. Useful when a workflow platform implements additional security trimming so the underlying platform web service should be switched off to avoid unauthorized access to a wider set of documents.

Twitter

The platform adapter for Twitter allows Appkit applications to search Twitter or retrieve a user's timeline using the Twitter API.

Setup

To add Twitter connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.twitter</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

Usage

This example shows how to configure a Twitter platform.

```
name: twigkit.search.twitter.Twitter oAuthConsumerKey: foo oAuthConsumerSecret:  
foo oAuthAccessToken: foo oAuthAccessSecret: foo
```

Required attributes

`oAuthConsumerKey (java.lang.String)`

OAuth consumer key for the Twitter API.

`oAuthConsumerSecret (java.lang.String)`

OAuth consumer secret for the Twitter API.

`oAuthAccessToken (java.lang.String)`

OAuth access token for the Twitter API.

`oAuthAccessSecret (java.lang.String)`

OAuth access secret for the Twitter API.

Optional attributes

`userTimeline (java.lang.String)`

User timeline to retrieve.

Attributes shared across platforms

`host (java.lang.String)`

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

`name (java.lang.String)`

The display name of this Platform instance. Used for identifying different platforms in the user interface.

`aliases (java.lang.String)`

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstnamelastname=name,countryofresidence=country"`

`defaultQuery (java.lang.String)`

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform. Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Wikipedia

The Wikipedia platform adds support to Appkit for the Wikipedia search engine.

Setup

To add Wikipedia connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.wikipedia</artifactId> <version>${project.parent.version}</version> </dependency>
```

Usage

This example shows how to configure a Wikipedia search endpoint. You can define the search endpoint through a configuration node (say `conf/platforms/wikipedia.conf`). For this you must include `name: twigkit.search.wikipedia.Wikipedia`, as shown here:

```
name: twigkit.search.wikipedia.Wikipedia defaultQuery: my_default_query
```

Attributes shared across platforms

host (java.lang.String)

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

name (java.lang.String)

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

```
aliases="firstnamelastname=name,countryofresidence=country"
```

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Wolfram|Alpha

The Wolfram|Alpha platform adds support to Appkit for the Wolfram|Alpha (<http://www.wolframalpha.com/>) computational knowledge engine.

Setup

To add Wolfram|Alpha connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.wolframalpha</artifactId> <version>${project.parent.version}</version> </dependency>
```

Usage

Assuming you have added the correct dependencies to your project (see above), the Wolfram Alpha platform configuration can be accessed by pointing to it in the `search:platform` tag. In this case the platform is placed in a variable named `platform` using the `var` attribute, and points to a platform configuration in `../conf/platforms/wa/wa.conf`:

```
<search:platform var="platform" conf="platforms.wa"></search:platform>
```

platforms/solr/solr.conf

In this example platform configuration, we set some general settings for the platform:

```
name: twigkit.search.wolframalpha.WolframAlpha appId: ABCD-EFGH
```

Required attributes

appId (`java.lang.String`)
Application ID API key.

Optional attributes

async (`java.lang.Boolean`)
Whether to request data in 'asynchronous' mode or not. In asynchronous mode, only the definitions of existing properties are returned in the first response, with links to further information or computations.
Default: `false`

Attributes shared across platforms

host (`java.lang.String`)
The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

name (`java.lang.String`)
The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

`aliases="firstnamelastname=name,countryofresidence=country"`

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

YouTube

The YouTube platform lets Appkit pull results from YouTube, which is often useful for presenting 'related videos'. The search can be scoped to a particular YouTube Channel if required.

Setup

To add YouTube connectors to a Maven project, you must add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.youtube</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Usage

This example shows how to configure a Youtube search endpoint. You can define the search endpoint through a configuration node (say `conf/platforms/youtube.conf`). For this you must include `name: twigkit.search.youtube.YouTube`, as shown here:

```
name: twigkit.web.youtube.YouTube apiKey: foo channelId: bar
```

Required attributes

apiKey (java.lang.String)

API Key for the application to access YouTube.

Optional attributes

channelId (java.lang.String)

The ID of the channel to restrict the search to.

Attributes shared across platforms

host (java.lang.String)

The URI of the platform. Must be specified using the full URI (for example, `http://localhost:1234`; `localhost:1234` will not work).

name (java.lang.String)

The display name of this Platform instance. Used for identifying different platforms in the user interface.

aliases (java.lang.String)

Manage mappings from field names to aliases (use a comma-separated list). All references to the field (via this platform) in results, facets, filters, and query strings are mapped. For example, to refer to a field in the index named `firstnamelastname` as `name` and `countryofresidence` as `country`, use these mappings:

```
aliases="firstnamelastname=name,countryofresidence=country"
```

defaultQuery (java.lang.String)

Default query to use when none is specified.

defaultFacets (java.lang.String)

Default facets to request when none are specified. This is a comma separated list.

spellCheck (java.lang.Boolean)

Spellcheck the Query term if supported by the Platform.

Default: `true`

expandQuery (java.lang.Boolean)

Apply advanced linguistics such as stemming or lemmatization if supported by the Platform.

Default: `true`

autoCorrect (java.lang.Boolean)

Whether to auto-correct and resubmit futile queries (queries with zero results).

Default: `true`

fileTypeField (java.lang.String)

Set which field contains information about file type (mime type).

resultIDField (java.lang.String)

Set which field represents the unique identifier for a given result.

pageLimit (java.lang.Long)

A limitation to which page the platform will offset.

Filters

Filters are AngularJS based filters that can be implemented in the markup by piping the filter name to a data object, which transforms the object. See *Filter Syntax* for more information.

- *actual*
- *actualValue*
- *dateFormat*
- *deepFilter*
- *display*
- *displayValue*
- *encodeURIComponent*
- *facet*
- *fact*
- *field*
- *filterCount*
- *filterValue*
- *formatAsCssName*
- *getUriForQuery*
- *hasFilter*
- *htmlToPlainText*
- *maxCharacters*
- *numberFormat*
- *orderByValue*
- *placeholder*
- *property*
- *pyramidSort*
- *rangeDateFormat*
- *result*
- *roundedNumber*
- *translate*
- *trusted*

Filter Syntax

This article describes the syntax of a single Appkit search filter to include in an HTTP request.

```
<filter-expression> ::= <field-expression> <group-expression> <value-expression>
<display-expression> <field-expression> ::= <exclusion-expression> <field-
optional-expression> <field-name> <exclusion-expression> ::= "-" | "" <field-
optional-expression> ::= "~" | "" <group-expression> ::= "(" <group-number> ")"
<group-number> ::= <digit> | <digit> <group-number> <field-name> ::= <field-name-
prefix> <field-name-suffix> <field-name-prefix> ::= <letter> | "_" <field-name-
suffix> ::= <field-symbol> | <field-symbol> <field-name-suffix> <field-symbol>
::= <letter> | <digit> | "_" | "-" | "." | ":" | " " | "" <value-expression> ::=
<value-bracket-left> <filter-value> <value-bracket-right> <value-bracket-left>
::= "[" | "(" <value-bracket-right> ::= "]" | ")" <filter-value> ::= <value> |
<range-value> <value> ::= <string-expression> | <numeric-value> | <date-value>
<string-expression> ::= <partial-match> <string-value> <partial-match> <string-
value> ::= "'" <text> "'" | "\"" <text> "\"" <partial-match> ::= '|' | '*' <numeric-
value> ::= <floating-point> | <integer> <number-sequence> ::= <digit> | <number-
sequence> <digit> <integer> ::= "0" | <positive-digit> <number-sequence>
<floating-point> ::= <integer> | <integer> "." <number-sequence> <date-value> ::=
<four-digits> "-" <two-digits> "-" <two-digits> "T" <two-digits> ":" <two-
digits> ":" <two-digits> <two-digits> ::= <digit> <digit> <four-digits> ::= <two-
digits> <two-digits> <range-value> ::= <value> "<value>" | "<value>" | "<value>"
<value> "<value>" <display-expression-opt> ::= "" | "[" <display-expression> "]"
<display-expression> ::= "[" "" <display-value> "" "]" | "[" "" <display-
value> "" "]"
```

Where:

```
<letter> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y"
| "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" |
"m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
; <positive-digit> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<digit> ::= "0" | <positive-digit>
```

Here:

- `<group-number>` is an integer value denoting a grouping of filter constraints
- `<field-name>` is the name of the field to apply the constraint to. A valid field name is one that matches the regular expression `[A-Za-z_][A-Za-z_0-9\-\.\:]`
- `<display-value>` is an optional label to attach to the filter
- `<filter-value>` is the actual constraint to be applied to the field named `<field-name>`

actual

Description

Renders fields actual value

Usage

In HTML Template Binding

```
{{ actual_expression | actual }}
```

In JavaScript

```
$filter('actual')(string)
```

Parameters

Param	Type	Details
string	string	The field to get the actual value from

actualValue

Description

Renders fields actual value

Usage

In HTML Template Binding

```
{{ actualValue_expression | actualValue }}
```

In JavaScript

```
$filter('actualValue')(string)
```

Parameters

Param	Type	Details
string	string	The field to get the actual value from

dateFormat

Description

Formats `date` to a string based on the requested `format`.

`format` string can be composed of the following elements:

- `'yyyy'`: 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)
- `'yy'`: 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)
- `'y'`: 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)
- `'MMMM'`: Month in year (January-December)
- `'MMM'`: Month in year (Jan-Dec)
- `'MM'`: Month in year, padded (01-12)
- `'M'`: Month in year (1-12)
- `'LLLL'`: Stand-alone month in year (January-December)
- `'dd'`: Day in month, padded (01-31)
- `'d'`: Day in month (1-31)
- `'EEEE'`: Day in Week, (Sunday-Saturday)
- `'EEE'`: Day in Week, (Sun-Sat)
- `'HH'`: Hour in day, padded (00-23)
- `'H'`: Hour in day (0-23)
- `'hh'`: Hour in AM/PM, padded (01-12)
- `'h'`: Hour in AM/PM, (1-12)
- `'mm'`: Minute in hour, padded (00-59)
- `'m'`: Minute in hour (0-59)
- `'ss'`: Second in minute, padded (00-59)
- `'s'`: Second in minute (0-59)
- `'sss'`: Millisecond in second, padded (000-999)
- `'a'`: AM/PM marker
- `'Z'`: 4 digit (+sign) representation of the timezone offset (-1200-+1200)
- `'ww'`: Week of year, padded (00-53). Week 01 is the week with the first Thursday of the year
- `'w'`: Week of year (0-53). Week 1 is the week with the first Thursday of the year
- `'G'`, `'GG'`, `'GGG'`: The abbreviated form of the era string (e.g. 'AD')
- `'GGGG'`: The long form of the era string (e.g. 'Anno Domini')
- `'medium'`: equivalent to `'MMM d, y h:mm:ss a'`
- `'short'`: equivalent to `'M/d/yy h:mm a'`
- `'fullDate'`: equivalent to `'EEEE, MMMM d, y'`
- `'longDate'`: equivalent to `'MMMM d, y'`
- `'mediumDate'`: equivalent to `'MMM d, y'`
- `'shortDate'`: equivalent to `'M/d/yy'`
- `'mediumTime'`: equivalent to `'h:mm:ss a'`
- `'shortTime'`: equivalent to `'h:mm a'`
- `'relative'`: displays the time since/until the date

`format` string can contain literal values. These need to be escaped by surrounding with single quotes (e.g. `"h 'in the morning'"`). In order to output a single quote, escape it - i.e., two single quotes in a sequence (e.g. `"h 'o''clock'"`).

Usage

In HTML Template Binding

```
{{ dateFormat_expression | dateFormat:format[:timezone] }}
```

In JavaScript

```
$filter('dateFormat')(date, format[, timezone])
```

Parameters

Param	Type	Details
date	Date number string	The date value to format.
format	string	The format to render the date value in.
timezone (optional)	string	Timezone to be used for formatting. It understands UTC/GMT and the continental US time zone abbreviations, but for general use, use a time zone offset, for example, '+0430' (4 hours, 30 minutes east of the Greenwich meridian) If not specified, the timezone of the browser will be used.

deepFilter

Description

Filters array by deep object properties

Usage

In HTML Template Binding

```
{{ deepFilter_expression | deepFilter }}
```

In JavaScript

```
$filter('deepFilter')(string)
```

Parameters

Param	Type	Details
string	string	The field to get the actual value from

display

Description

Renders fields actual value

Usage

In HTML Template Binding

```
{{ display_expression | display }}
```

In JavaScript

```
$filter('display')(string)
```

Parameters

Param	Type	Details
string	string	The field to get the actual value from

displayValue

Description

Renders fields display value

Usage

In HTML Template Binding

```
{{ displayValue_expression | displayValue }}
```

In JavaScript

```
$filter('displayValue')(string)
```

Parameters

Param	Type	Details
string	string	The field to get the display value from

encodeURIComponent

Description

Renders fields actual value

Usage

In HTML Template Binding

```
{{ encodeURIComponent_expression | encodeURIComponent }}
```

In JavaScript

```
$filter('encodeURIComponent')(string)
```

Parameters

Param	Type	Details
string	string	The value to encode

facet

Description

Returns a result from the response

Usage

In HTML Template Binding

```
{{ facet_expression | facet:facetName }}
```

In JavaScript

```
$filter('facet')(response, facetName)
```

Parameters

Param	Type	Details
response	object	The response object to get the facet from
facetName	string	The facet name

fact

Description

Returns the first fact of the provided fact name from the response. If no fact is found it will return null.

Usage

In HTML Template Binding

```
{{ fact_expression | fact:fact-name }}
```

In JavaScript

```
$filter('fact')(response, fact-name)
```

Parameters

Param	Type	Details
response	object	The response object to get the fact from
fact-name	string	The fact name

field

Description

Returns a result from the response

Usage

In HTML Template Binding

```
{{ field_expression | field[:field] }}
```

In JavaScript

```
$filter('field')(result[, field])
```

Parameters

Param	Type	Details
result	string	The result object to get the field from.
field (optional)	number	The name of the field to retrieve

filterCount

Description

Returns a result from the response

Usage

In HTML Template Binding

```
{{ filterCount_expression | filterCount[:value] }}
```

In JavaScript

```
$filter('filterCount')(facet[, value])
```

Parameters

Param	Type	Details
facet	string	The facet object to get the filter from
value (<i>optional</i>)	number	The value of the filter to get

filterValue

Description

Returns the facet filter that has the requested value

Usage

In HTML Template Binding

```
{{ filterValue_expression | filterValue[:value] }}
```

In JavaScript

```
$filter('filterValue')(facet[, value])
```

Parameters

Param	Type	Details
facet	string	The facet object to get the filter from
value (<i>optional</i>)	number	The value of the filter to get

formatAsCssName

Description

Renders a string in a friendly name for CSS.

Usage

In HTML Template Binding

```
{{ formatAsCssName_expression | formatAsCssName }}
```

In JavaScript

```
$filter('formatAsCssName')(string)
```

Parameters

Param	Type	Details
string	string	The string to format.

getUrlForQuery

Description

Returns the query parameters for a query object

Usage

In HTML Template Binding

```
{{ getUrlForQuery_expression | getUrlForQuery }}
```

In JavaScript

```
$filter('getUrlForQuery')(query)
```

Parameters

Param	Type	Details
query	Object	The query object

hasFilter

Description

Returns the first filter object from the query or facet that matches the name and optionally the value also

Usage

In HTML Template Binding

```
{{ hasFilter_expression | hasFilter:field[:value] }}
```

In JavaScript

```
$filter('hasFilter')(query, field[, value])
```

Parameters

Param	Type	Details
query	string	The query object
field	number	The name of the filter
value (<i>optional</i>)	number	The value of the filter

htmlToPlainText

Description

Strip HTML from a string.

Usage

In HTML Template Binding

```
{{ htmlToPlainText_expression | htmlToPlainText }}
```

In JavaScript

```
$filter('htmlToPlainText')(string)
```

Parameters

Param	Type	Details
string	string	The value to strip the html from.

maxCharacters

Description

Renders fields actual value

Usage

In HTML Template Binding

```
{{ maxCharacters_expression | maxCharacters:maxCharacters }}
```

In JavaScript

```
$filter('maxCharacters')(string, maxCharacters)
```

Parameters

Param	Type	Details
string	string	The field to get the actual value from
maxCharacters	number	Number of characters to display

numberFormat

Description

Renders a number in the format specified.

Usage

In HTML Template Binding

```
{{ numberFormat_expression | numberFormat[:format] }}
```

In JavaScript

```
$filter('numberFormat')(number[, format])
```

Parameters

Param	Type	Details
number	number	The number to format.
format (optional)	string	string for the numbers used using the Java DecimalFormat pattern. Default: '#,##0.###'

orderByValue

Description

Orders values from facet or results

Usage

In HTML Template Binding

```
{{ orderByValue_expression | orderByValue }}
```

In JavaScript

```
$filter('orderByValue')([desc])
```

Parameters

Param	Type	Details
desc (<i>optional</i>)	boolean	Whether it should be descending or not.

placeholder

Description

Renders a placeholder while a field is null.

Usage

In HTML Template Binding

```
{{ placeholder_expression | placeholder:placeholder }}
```

In JavaScript

```
$filter('placeholder')(input, placeholder)
```

Parameters

Param	Type	Details
input	string	The input to check if it is null
placeholder	string	The value to display while input is null

property

Description

Get value from an array based on the value of a property.

Usage

In HTML Template Binding

```
{{ property_expression | property:value:returnProperty }}
```

In JavaScript

```
$filter('property')(property, value, returnProperty)
```

Parameters

Param	Type	Details
property	string	The name of the property to filter on.
value	string	The value of the property to filter on.
returnProperty	string	The property to return.

pyramidSort

Description

Sorts values in a pyramid fashion

Usage

In HTML Template Binding

```
{{ pyramidSort_expression | pyramidSort:field }}
```

In JavaScript

```
$filter('pyramidSort')(items, field)
```

Parameters

Param	Type	Details
items	object	Array of values
field	string	The field to sort on.

rangeDateFormat

Description

Formats `date` to a string based on the requested `format`.

If a `from` and `to` value are present the from value will be used for formatting a display. If only a from value is present in the filter the from value will be formatted and prefixed with a `>` character. If only a to value is present then that value will be formatted and prefixed with a `<`

`format` string can be composed of the following elements:

- `'yyyy'`: 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)
- `'yy'`: 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)
- `'y'`: 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)
- `'MMMM'`: Month in year (January-December)
- `'MMM'`: Month in year (Jan-Dec)
- `'MM'`: Month in year, padded (01-12)
- `'M'`: Month in year (1-12)
- `'LLLL'`: Stand-alone month in year (January-December)
- `'dd'`: Day in month, padded (01-31)
- `'d'`: Day in month (1-31)
- `'EEEE'`: Day in Week, (Sunday-Saturday)
- `'EEE'`: Day in Week, (Sun-Sat)
- `'HH'`: Hour in day, padded (00-23)
- `'H'`: Hour in day (0-23)
- `'hh'`: Hour in AM/PM, padded (01-12)
- `'h'`: Hour in AM/PM, (1-12)
- `'mm'`: Minute in hour, padded (00-59)
- `'m'`: Minute in hour (0-59)
- `'ss'`: Second in minute, padded (00-59)
- `'s'`: Second in minute (0-59)
- `'sss'`: Millisecond in second, padded (000-999)
- `'a'`: AM/PM marker
- `'Z'`: 4 digit (+sign) representation of the timezone offset (-1200-+1200)
- `'ww'`: Week of year, padded (00-53). Week 01 is the week with the first Thursday of the year
- `'w'`: Week of year (0-53). Week 1 is the week with the first Thursday of the year
- `'G'`, `'GG'`, `'GGG'`: The abbreviated form of the era string (e.g. 'AD')
- `'GGGG'`: The long form of the era string (e.g. 'Anno Domini')
- `'medium'`: equivalent to `'MMM d, y h:mm:ss a'`
- `'short'`: equivalent to `'M/d/yy h:mm a'`
- `'fullDate'`: equivalent to `'EEEE, MMMM d, y'`
- `'longDate'`: equivalent to `'MMMM d, y'`
- `'mediumDate'`: equivalent to `'MMM d, y'`
- `'shortDate'`: equivalent to `'M/d/yy'`
- `'mediumTime'`: equivalent to `'h:mm:ss a'`
- `'shortTime'`: equivalent to `'h:mm a'`
- `'relative'`: displays the time since/until the date

`format` string can contain literal values. These need to be escaped by surrounding with single quotes (e.g. `"h 'in the morning'"`). In order to output a single quote, escape it - i.e., two single quotes in a sequence (e.g. `"h 'o''clock'"`).

Usage

In HTML Template Binding

```
{{ rangeDateFormat_expression | rangeDateFormat:format }}
```

In JavaScript

```
$filter('rangeDateFormat')(value, format)
```


Parameters

Param	Type	Details
value	object	Range filters value object (eg. filter.val)
format	string	The format to render the date value in.

result

Description

Returns a result from the response

Usage

In HTML Template Binding

```
{{ result_expression | result[:index] }}
```

In JavaScript

```
$filter('result')(response[, index])
```

Parameters

Param	Type	Details
response	string	The response object to get the result from.
index (<i>optional</i>)	number	The index of the result to retrieve, index starts at 0. (Defaults to 0).

roundedNumber

Description

Renders a number rounded, eg. 1234 = 1.2k

Usage

In HTML Template Binding

```
{{ roundedNumber_expression | roundedNumber }}
```

In JavaScript

```
$filter('roundedNumber')(value)
```

Parameters

Param	Type	Details
value	number	The number to round.

translate

Description

Translate a term using the Translation Service

Usage

In HTML Template Binding

```
{{ translate_expression | translate[:bundle] }}
```

In JavaScript

```
$filter('translate')(term[, bundle])
```

Parameters

Param	Type	Details
term	string	The term to translate
bundle (<i>optional</i>)	string	The translation bundle to use

trusted

Description

Returns a trusted url. This is required if trying to use an angular expression in the media idea src attribute.

Usage

In HTML Template Binding

```
{{ trusted_expression | trusted }}
```

In JavaScript

```
$filter('trusted')(url)
```

Parameters

Param	Type	Details
url	string	The url to return.

Installation

This section contains various articles on the installation, system requirements, and supported environments of App Studio.

Appkit is most commonly run as a lightweight Java web application that can be deployed in any Java Servlet Container (http://en.wikipedia.org/wiki/Java_Servlet#Container_servers) (application server) such as Tomcat (<http://tomcat.apache.org/>), Glassfish (<http://glassfish.java.net/>), Jetty (<http://www.eclipse.org/jetty/>) depending on your preferences or IT standards. The Appkit application is designed to be stateless (http://en.wikipedia.org/wiki/Stateless_protocol), making deployment and scaling (for example, with load balancers ([http://en.wikipedia.org/wiki/Load_balancing_\(computing\)](http://en.wikipedia.org/wiki/Load_balancing_(computing)))) a straightforward exercise.

Requirements

App Studio is a standalone application with the following requirements:

- Oracle Java JDK (<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>) 1.8 or later 64-bit (x64 only)
- A running instance of *Fusion Server*, with one or more collections of indexed data

We support the following JDKs:

- OpenJDK 1.8 (<https://openjdk.java.net/install/>), 64-bit (x64/x86_64 only)
- AdoptOpenJDK 1.8 (<https://adoptopenjdk.net/releases.html?variant=openjdk8&jvmVariant=hotspot>), 64-bit (x64 only)

For Windows users, the Fusion start and stop logging output found in `$Fusion\var\logs\metrics\metric.log` tends to be more reliable when using AdoptOpenJDK.

Installation

1. Download *App Studio Enterprise*.
2. Copy the App Studio Enterprise zip file to any convenient path.
3. Unzip the file, and follow the instructions provided in the `README.md` file.

Configuring the application server

For development and testing purposes you can deploy the Appkit web application in any modern servlet container and it will run and serve queries as long as the computer has network access to the underlying search engine. This is a non-exhaustive list of servers that have been used to run Appkit applications:

- Tomcat 6+
- JBoss 8+
- Jetty 6+
- GlassFish 4+

Java runtime environment

Appkit requires Java JDK 8 to be installed on the application server. Java EE is not needed.

Memory allocation

You must make sure that the servlet container is configured with access to an adequate amount of memory and heap space. The standard Appkit application does not have specific requirements but specific modules (for example, the *Social Module* have a larger footprint. See the *Troubleshooting* page for more information.

Typical production setup

Your production setup will be a matter of preference because Appkit imposes no specific requirements in terms of topology or infrastructure. Some organizations choose to front the application server (servlet container) with either Apache or another webserver. In a

public-facing scenario, this can yield some gains when serving static content and creates a buffer between the outside world and the execution environment.

Using Windows Integrated Authentication for internal applications

If you would like to take advantage of Integrated Windows Authentication (http://en.wikipedia.org/wiki/Integrated_Windows_Authentication) for single sign on (http://en.wikipedia.org/wiki/Single_sign-on) in your search application, the typical pattern is to front the servlet container (for example, Tomcat) with Microsoft Internet Information Services (<http://www.iis.net/>). In this pattern IIS handles the authentication with Active Directory and then forwards the request to Tomcat where it is picked up by Appkit and used by our *Security Module* for user information, authorization or personalization.

Related articles

- *Building from Source*
- *Deploying a WAR File*

Open Source Libraries

Below is a list of Open Source Libraries that Appkit uses, along with license details.

Open Source Library	License Details
SLF4J	Logging abstraction framework. MIT License.
Log4j (optional)	Logging framework. Apache 2.0 License.
Commons-IO	Utilities. Apache 2.0 License.
Apache Lucene (optional)	Highlighting. Apache 2.0 License.
WR04j (optional)	CSS and JavaScript compiler. Apache 2.0 License.
URLRewriteFilter (optional)	URL routing. BSD License.
CORS Filter	Cross-origin scripting capabilities. Apache 2.0 License.
Sitebricks (optional)	Used to provide configuration to Sitebricks. Apache 2.0 License.
Hibernate	Persistence layer integration. Apache 2.0 License.
Spring Security (optional)	Authentication and authorization provider abstraction. Apache 2.0 License.
JodaTime	Date calculations. Apache 2.0 License.
Jasypt	Encryption/decryption. Apache 2.0 License.
Google Guice	Dependency injection. Apache 2.0 License.
Jersey (optional)	Web services. CDDL 1.1 and GPL 2 with class path exception licenses.
OWASP ESAPI	Security and vulnerability layer. BSD License.
Bouncy Castle License	Cryptography. MIT License.
Apache Derby (optional)	Database. Apache 2.0 License.
Markdownj (optional)	Markdown parsing. Custom license, allows redistribution and use through attribution.
Rome (optional)	RSS parser and generator. Apache 2.0 License.
Jackson	JSON parsing and generation. Apache 2.0 and LGPL licenses.

Open Source Library	License Details
Quartz Scheduler (optional)	Event scheduling for saved searches and alerting. Apache 2.0 License.
Plexus	Application building. Apache 2.0 License.
JUnit	Unit testing – not included in deliverable. IBM Common Public License 0.5.
Atunit	Unit testing – not included in deliverable. Apache 2.0 License.
EasyMock	Unit testing – not included in deliverable. Apache 2.0 License.
ImageIO-Ext (optional)	Rich media transcoding. LGPL 2.1 and BSD licenses.

Software Requirements

This software is required to use App Studio and Appkit to develop, build, and deploy search apps.

Software	Version and Notes
Appkit	A version of Appkit is included in each App Studio release. Appkit is also released independently. For information about upgrading an app to use a later version of Appkit, see link:/how-to/791/upgrade-appkit-in-existing-apps [Upgrade Appkit in existing apps].
Oracle Java 8 JDK or later	Download (https://www.oracle.com/technetwork/java/javase/downloads/index.html) and install the Oracle Java SE JDK 8 or newer for Windows (http://www.oracle.com/technetwork/java/javase/documentation/install-windows-152927.html) or Linux (http://www.oracle.com/technetwork/java/javase/install-linux-self-extracting-138783.html) (it comes pre-installed on Macs).
Node.js 8 or later	Download (https://nodejs.org/en/download/) and install Node.js 8 or later.
npm	The <code>npm</code> package manager is included with Node.js.

Supported Devices and Browsers

What browsers can be supported? For example IE 11 or 9?

Any modern browser will be perfectly fine. For Internet Explorer, we recommend at least version 11. Appkit uses progressive enhancements so for older browsers the application will still work but in a feature limited fashion.

How does AppKit support responsive design?

All standard Appkit templates are responsively designed and will work correctly on tablets and mobiles with full feature parity.

What is the process we must follow to accomplish responsive design?

As long as the existing templates are not compromised, responsive design will work out of the box. The user interface being adapted is a customization of a fully responsive template.

Licensing

This article contains basic licensing details.

App Studio comes with a trial license file, located at `app-studio/app-studio.lic`. This trial license is valid for 30 days. Contact Lucidworks to obtain a commercial license.

To install a commercial license, copy the new license file to `app-studio/app-studio.lic`, overwriting the trial license. Commercial licenses support multiple instances of App Studio. Lucidworks issues a separate license for each App Studio instance.

What is an Appkit license file?

To run an Appkit application, you need an Appkit license file. This is a small encrypted text file that an Appkit application reads on startup. The license file enables the individual Appkit *modules* and *search platforms* that you need for your application.

Types of licenses

Development licenses

To run an app in **development**, you will need a developer license from us. To generate a development license, go to the Manage License Keys page (<https://dev.twigkit.net/settings/keys/>) (accessed via the drop down menu by your logged-in username) and click 'new developer license':

Personal settings

Change your password
License keys

Twigkit Launchpad > Settings >
Manage License Keys

Download and manage your license keys.

New developer license

Valid from	Expires on	Download	Terms and conditions
2015-09-01	2015-11-01	twigkit_dev_2015-11-01.lic	This license is issued to jobloggs@mycompany.com and is valid only in non-production environments

License keys
Change password
Administration settings
Sign out

These are usually time-limited, enable all Appkit features, and not tied to any given hostname, so you can run them and develop locally, and try out all Appkit features.

Production licenses

For **production**, two tiers of licensing are available: *per host* and *enterprise*.

- **Per-host.** The easiest way to get started with Appkit is through our low-cost, host-based licenses. As the name suggests, these licenses are bound to a single machine hostname. The license is purchased to cover an agreed period (usually 1 year). They are bound to specific modules.
- **Enterprise.** Lucidworks also provides enterprise-wide licenses for customers wanting unlimited usage of Appkit throughout their entire organization. These annual licenses include access to source code, greater involvement in our development roadmap, and premium developer support.

For more information on Per-host and Enterprise licensing, contact Lucidworks (<https://lucidworks.com/company/contact/>).

How do I add a license?

License files should be named `app-studio.lic` and stored where the application can read them.

You can specify the location of the license file in the application's configuration tree – specifically, in `src/main/resources/conf/twigkit.conf` you will find this (default) entry

```
license-file: /app-studio.lic
```

This entry can either be *classpath-relative* (as given above) or given as an *absolute file path* (with a `"file://"` prefix, for example, `file:///var/opt/lics/app-studio.lic`).

When the license location is given as a relative path, the application tries to look up the license file on the runtime class path. So you can place the license in for example, the `src/main/resources` folder in your application, which will work. If no license file is found on the class path, it looks in the home directory of the user that spun up the JVM. The latter is useful for development, by having a single license on the dev workstation.

License errors and problems

If you are seeing license errors or exceptions (like `'twigkit.license.LicenseException: null at twigkit.license.PLM.isValid'`), it could be that you have either mis-configured your license (for example, it is not named `app-studio.lic`), or it is being used on the wrong host or not visible by the application.

Contact Lucidworks Support (<https://support.lucidworks.com/hc/en-us/requests/new>) and provide your hostname, license file, and full details of the error in the Appkit logs, and we will be able to advise further.

Styles

This article lists the valid values for the `styling` attribute supported by many App Studio markup tags.

Global styles

These styles can be applied to any element.

Flex utilities

- `stretch-content`: Uses flex box to stretch the child content
See the `grid` and `panel` examples in the gallery.
- `stretch-row`: Same as `stretch-content` but with auto height for use with rows
- `center-content`: Center child content vertically and horizontally
See the `grid` and `panel` examples in the gallery.
- `flex-nowrap`: Prevent flex wrapping
- `flex-grow`: Flex grow an element

Width utilities

- `width-1`: 100%
- `width-2-3`: 66%
- `width-1-2`: 50%
- `width-1-3`: 33%
- `width-1-4`: 25%

Display utilities

- `inline-center`
- `block`
- `inline`
- `inline-block`
- `pull-left`: float left
See the `button` example in the gallery.
- `pull-right`: float right
See the `button` example in the gallery.
- `floated left`: float element left
- `floated right`: float element right
- `border`: Add 1px border to element
- `border right`: Add 1px right border to element
- `border left`: Add 1px left border to element
- `border bottom`: Add 1px bottom border to element
- `border top`: Add 1px top border to element
- `centercontent`: adds padding that pushes in from page edges.
- `centercontent-inner`: adds borders and box-shadows etc. – should be used inside `centercontent`
- `centercontent-inner-shadow`: Same as `centercontent` but with a drop shadow
- `fade`: Fades out an element, add the class in to fade back in
- `bg-stretch`: Stretch a background image to the full element height and width

Headers

- `header header-fixed`: position fixed header
- `header header-shadow`: header with drop shadow
- `header header-gradient`: header with gradient
- `header header-border`

Sizes

- `xs`: Make the font-size the smallest – also works for making most elements smaller
- `sm`
- `md`

- lg
- xl
- break-text-all: Make text wrap mid word
- break-text-normal: Text wraps at best fit

Border radius

- border-radius: Adds rounded corners
- border-radius-top-left: Adds rounder corner to top left
- border-radius-top-right
- border-radius-bottom-left
- border-radius-bottom-right

Typography

- text-center
- text-right
- text-left
- capitalize
- uppercase
- lowercase

Colors

See these *colors* in the gallery.

- bg-lgrey: Light grey background
- bg-mgrey
- bg-dgrey
- bg-charcoal
- bg-black
- bg-white
- text-white: White text
- bg-lbgrey
- bg-mbgrey
- bg-dbgrey
- bg-blue
- bg-sblue
- bg-yellow
- bg-orange
- bg-red
- bg-pink
- bg-purple
- bg-green
- bg-primary
- bg-primary-light
- bg-primary-dark
- bg-primary-contrast
- text-primary
- text-primary-contrast
- bg-secondary
- bg-secondary-light
- bg-secondary-dark
- bg-secondary-contrast
- text-secondary
- text-secondary-light
- text-secondary-dark
- text-secondary-contrast
- bg-tertiary
- bg-tertiary-light
- bg-tertiary-dark
- bg-tertiary-contrast
- text-tertiary

- text-tertiary-light
- text-tertiary-dark
- text-tertiary-contrast
- bg-raspberry
- bg-raspberry-light
- bg-raspberry-dark
- text-raspberry
- text-raspberry-light
- text-raspberry-dark
- bg-strawberry
- bg-strawberry-light
- bg-strawberry-dark
- text-strawberry
- text-strawberry-light
- text-strawberry-dark
- bg-carrot
- bg-carrot-light
- bg-carrot-dark
- text-carrot
- text-carrot-light
- text-carrot-dark
- bg-pumpkin
- bg-pumpkin-light
- bg-pumpkin-dark
- text-pumpkin
- text-pumpkin-light
- text-pumpkin-dark
- bg-sunflower-light
- bg-sunflower
- bg-sunflower-dark
- text-sunflower
- text-sunflower-light
- text-sunflower-dark
- bg-leaf
- bg-leaf-light
- bg-leaf-dark
- text-leaf
- text-leaf-light
- text-leaf-dark
- bg-grass
- bg-grass-light
- bg-grass-dark
- text-grass
- text-grass-light
- text-grass-dark
- bg-emerald
- bg-emerald-light
- bg-emerald-dark
- text-emerald
- text-emerald-light
- text-emerald-dark
- bg-mint
- bg-mint-light
- bg-mint-dark
- text-mint
- text-mint-light
- text-mint-dark
- bg-aqua
- bg-aqua-light
- bg-aqua-dark
- text-aqua
- text-aqua-light

- text-aqua-dark
- bg-sky
- bg-sky-light
- bg-sky-dark
- text-sky
- text-sky-light
- text-sky-dark
- bg-royal
- bg-royal-light
- bg-royal-dark
- text-royal
- text-royal-light
- text-royal-dark
- bg-lavender
- bg-lavender-light
- bg-lavender-dark
- text-lavender
- text-lavender-light
- text-lavender-dark
- bg-pink
- bg-pink-light
- bg-pink-dark
- text-pink
- text-pink-light
- text-pink-dark
- bg-paper
- bg-paper-light
- bg-paper-dark
- text-paper
- text-paper-light
- text-paper-dark
- bg-concrete
- bg-concrete-light
- bg-concrete-dark
- text-concrete
- text-concrete-light
- text-concrete-dark
- bg-asphalt
- bg-asphalt-light
- bg-asphalt-dark
- text-asphalt
- text-asphalt-light
- text-asphalt-dark
- bg-duckegg
- bg-duckegg-light
- bg-duckegg-dark
- text-duckegg
- text-duckegg-light
- text-duckegg-dark
- bg-depth-dblue
- text-depth-dblue
- bg-depth-lblue
- text-depth-lblue

Switches

Usage:

```
<input type="checkbox" styling="checkbox-switch"> <label>Switch Label</label>
```

- checkbox-switch: Default switch
- Add styles to label:

```
<input type="checkbox" styling="checkbox-switch"> <label  
styling="square">Switch Label</label>
```

- square
- flat

Switch colors

Also added to the label * positive * negative * warning * info * secondary * tertiary

Right-hand-side panel

Adds a wikipedia-esque smart panel * rhspanel * rhspanel-shadow

Tooltips

- tooltip: Adds a tooltip using the aria-label attribute for the tooltip content
- tooltip-top-left: Position tooltip
- tooltip-top-right
- tooltip-top
- tooltip-bottom-left
- tooltip-bottom-right
- tooltip-bottom
- tooltip-left
- tooltip-right
- tooltip-top
- tooltip-small
- tooltip-medium
- tooltip-large
- tooltip-error
- tooltip-warning
- tooltip-info
- tooltip-success
- tooltip-always: The tooltip is always shown
- tooltip-rounded: Rounded corners
- tooltip-no-animate
- tooltip-bounce

Hero banner

This style is useful for landing pages.

- hero-banner

Label

- label

Buttons

- `button`: Default button
- `button-hover`: Hover state always active
- `button-secondary`: Secondary color
- `button-green`
- `button-blue`
- `button-white`
- `button-disabled`
- `button-hidden`
- `block`: Full width

Forms

- `checkbox`
- `radio`
- `input-rounded`: Applies only to inputs
- `group`: Form input group
- `form-stacked`: Stacked form – should be applied to the form
- `form-aligned`
- `labelled-input`: Labelled input e.g. an input with a dollar sign attached as a prefix
- `labelled-input label-right`: Labelled input e.g. an input with a percentage sign attached as a suffix
- `styled-checkbox`: CSS checkbox – should be applied to a checkbox input with a label as the next sibling

Lists

These styles can be applied to ordered (`ol`) or unordered (`ul`) HTML elements.

- `striped`
- `lined`

Grid utilities

- `hide-xs`: Hide on XS size screens
- `show-xs`: Show on XS size screens
- `show-sm`
- `show-md`
- `show-lg`
- `show-xl`
- `show-xs`
- `hide-xs`
- `hide-sm`
- `show-sm`
- `hide-md`
- `show-md`
- `hide-lg`
- `show-lg`
- `hide-xl`
- `show-xl`

Component styles

These styles can be applied to specific tags.

Field styles

Styles for the `search:field` tag:

- `title`
- `url`
- `label-left`
- `label-left-always`

- `label-inline`
- `label-above`
- `description`
- `image-top`

Image

Styles for the `media:image` tag:

- `square-crop`
- `circle-crop`

Facet

This style can be applied to the `search:facet` tag:

- `switches`: Render switches instead of checkboxes.
Unlike the default checkboxes, only one switch can be selected at a time.

Facet list

Styles for `search:facet-list` tag:

- `shadow`
- `shadow-far`
- `recessed`
- `step`
- `simple`
- `wrappedheader`
- `hidemetadata`
- `indented`

Search tabs and nav bars

These styles are for the `search:tabs` tag.

- `navbar`: Navbar style
- `navbar navbar-shadow`
- `navbar navbar-centered`
- `navbar navbar-tabs`: Tab style of navbar
- `navbar navbar-light`: Light version
- `tabs`: Tabs style

Comment list

Styles for the `collaborate:comment-list` tag:

- `hide-avatars`: Hides the comment avatars

Pagination

Styles for `search:pagination` tag.

- `pagination-wrapper-left`: Align left
- `pagination-wrapper-center`
- `pagination-wrapper-right`

Alert box

Styles for `search:alert` tag.

- `alertbox`: Style element as an alert box
- `alertbox-secondary`: Secondary color
- `alertbox-positive`
- `alertbox-negative`
- `alertbox-warning`

- `alertbox-info`

Interruptors

Apply to any element or use the `block:interruptor` tag.

- `interruptor`: Style element as an attention grabber
See the `message` example in the gallery.

Scroll to

Styles for the `widget:scroll-to` tag:

- `bottom-left`: Position bottom left of the page
- `bottom-right`
- `bottom-center`

Toggle panes

Styles to be used like:

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="pills" styling="pill">
</toggle:controls> <toggle:panes id="pills"> <toggle:pane> ...
```

See `toggle:controls`.

- `tabs-shadow`: Adds shadow to tabs
- `collapsible`: Allows tabs to be collapsed by clicking the active tab
- `pill`: Pill style
- `line`
- `simple`
- `bordered`

Cards

- `basic-card`: Basic card style
- `card`: Default card style

Card results

The card widths are auto-generated. By default a maximum of 6 result cards can be shown on one row. By setting `@maxCardsPerRow` this can be increased or decreased.

On a `search:result-list` add the card styling in the following format:

```
cards-{size}-{numberPerRow}
```

For example:

```
<search:result-list styling="cards-sm-2 cards-lg-4">
```

This will show 2 cards per row on small screens, 4 on large screens.

Card styles

Styles for the `search:result` tag:

- `card-emphasis`
- `card-shadow`
- `card-step`

Popover

Styles for `widget:popover` tag:

- `popover-wrapper`: Popover style
- `popover-large`: Large popover
- `popover-medium`
- `popover-small`

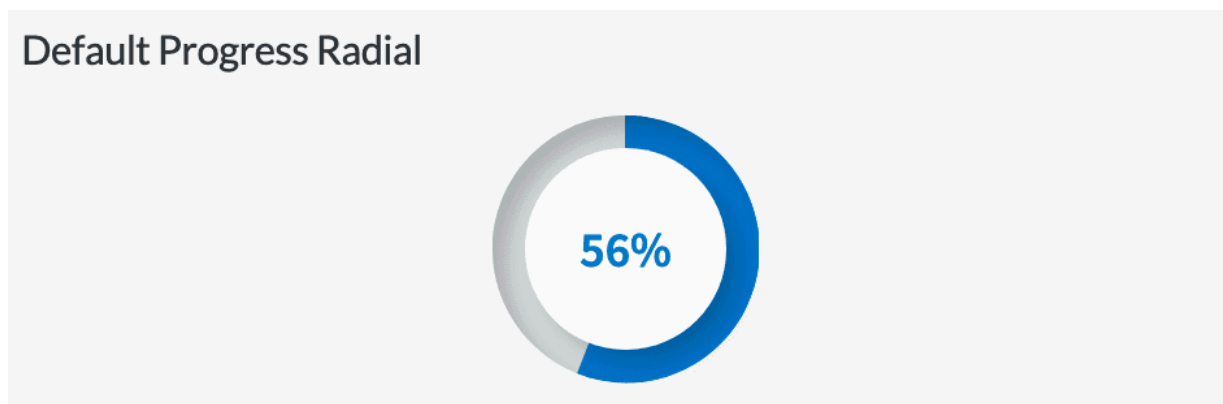
Tags

The App Studio markup language defines a set of HTML-like elements, or *tags*, for defining search queries to the backend and visualizing the search results and facets, handling all the complexities of connecting to Fusion.

For example, adding a radial progress bar is as simple including the following tag:

```
<progress:radial percentage="56"> </progress:radial>
```

This results in the following:



Over 150 tags are provided in the Appkit framework.

advanced:button

Description

Creates a button for triggering the advanced search.

Usage

as element:

```
<advanced:button [button-label="{string}"] [search-within-checkbox="{boolean}"]  
[search-within-label="{boolean}"] [search-within-default-value="{boolean}"]  
[action="{string}"]> </advanced:button>
```

Parameters

Param	Type	Details
button-label (optional)	string	Label for the button
search-within-checkbox (optional)	boolean	Show the 'search within' option as a checkbox. This enables the query to retain the filters already applied when checked. If not checked the search is reset when a query is entered and the form submitted. If you want this functionality but don't want a checkbox showing, you can use this attribute set to be false, and set <code>search-within-default-value</code> to be true – this uses a hidden field instead of a checkbox.
search-within-label (optional)	boolean	The label for the 'search within' checkbox.
search-within-default-value (optional)	boolean	Whether to enable 'search within' by default. The default is option is 'false'.
action (optional)	string	Page the search should take place on, results will not change automatically when an action is set.

advanced:date-range

Description

Creates an advanced search date range picker. Must be used within a `<search:advanced>` tag. A filter for the field specified is only set once both dates have been set.

Usage

as element:

```
<advanced:date-range field="{string}" values="{string}" [date-format="{string}"]  
[min-view="{string}"]> </advanced:date-range>
```

Parameters

Param	Type	Details
field	string	The field to apply the filter to.
values	string	The checkbox values to display. These can be represented (comma separated) as field=displayName combinations and must match the actual value of a filter.
dateFormat (optional)	string	The display date format of the dates. Defaults to 'medium'.
minView (optional)	string	The minimum view of the date picker to display. Values: 'year/month/day/hour/minute'. Default 'minute'.

Example

Source

```
<search:advanced> <search:box></search:box> <advanced:date-range  
fields="source=Source,Material"></advanced:date-range> </search:advanced>
```

Result

From

To

▼

▼

+

advanced:text

Description

Creates an advanced search input field. Must be used within a <search:advanced> tag. The input value will be set as the filter value for the field specified.

Usage

as element:

```
<advanced:text field="{string}" label="{string}" placeholder="{string}" multiple>
</advanced:text>
```

Parameters

Param	Type	Details
field	string	The field to apply the filter to.
label	string	The input label.
placeholder	string	The input placeholder.
multiple	boolean	Allows additional input fields to be created by the user. Default true.

block:interruptor

Description

Creates an interruptor box with the transcluded content

Usage

as element:

```
<block:interruptor> </block:interruptor>
```

Example

Source

```
<block:interruptor> <h2>Heads Up</h2> <p>I'm interrupting you!</p>
</block:interruptor>
```

Result

EXAMPLE

Heads Up
I'm interrupting you!

```
<block:interruptor>
  <h2>Heads Up</h2>
  <p>I'm interrupting you!</p>
</block:interruptor>
```

chart:credit

Description

To display a credit on the chart in the bottom right hand corner use the Chart Credit tag with the text and href attributes. This tag must be nested in the *Chart Display* tag.

Usage

as element:

```
<chart:credit [text="{string}"] [href="{string}"]> </chart:credit>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
text (optional)	string	The text for the credits label.
href (optional)	string	Url to redirect the user to when they click the credit.

Example

Source

```
<chart:display response="response" facet-name="test" type="column"> <chart:credit  
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> </chart:display>
```

```
angular.module('lightning') .controller('ExampleController',  
 ['$scope', '$timeout', 'ResponseService', function($scope,$timeout,ResponseService)  
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { test: { filters: [ {  
val: { dsp: 'dsp1', act: 1 }, count: 1, }, { val: { dsp: 'dsp2', act: 2 }, count:  
2, } ] }, }, results: [ { result: { fields: { label: 'label1', value: 1 } }, }, {  
result: { fields: { label: 'label2', value: 2 } }, hits: { act: ['Hits', 264] }  
} ] } ResponseService.setResponse('response',$scope.response); }]);
```


Result

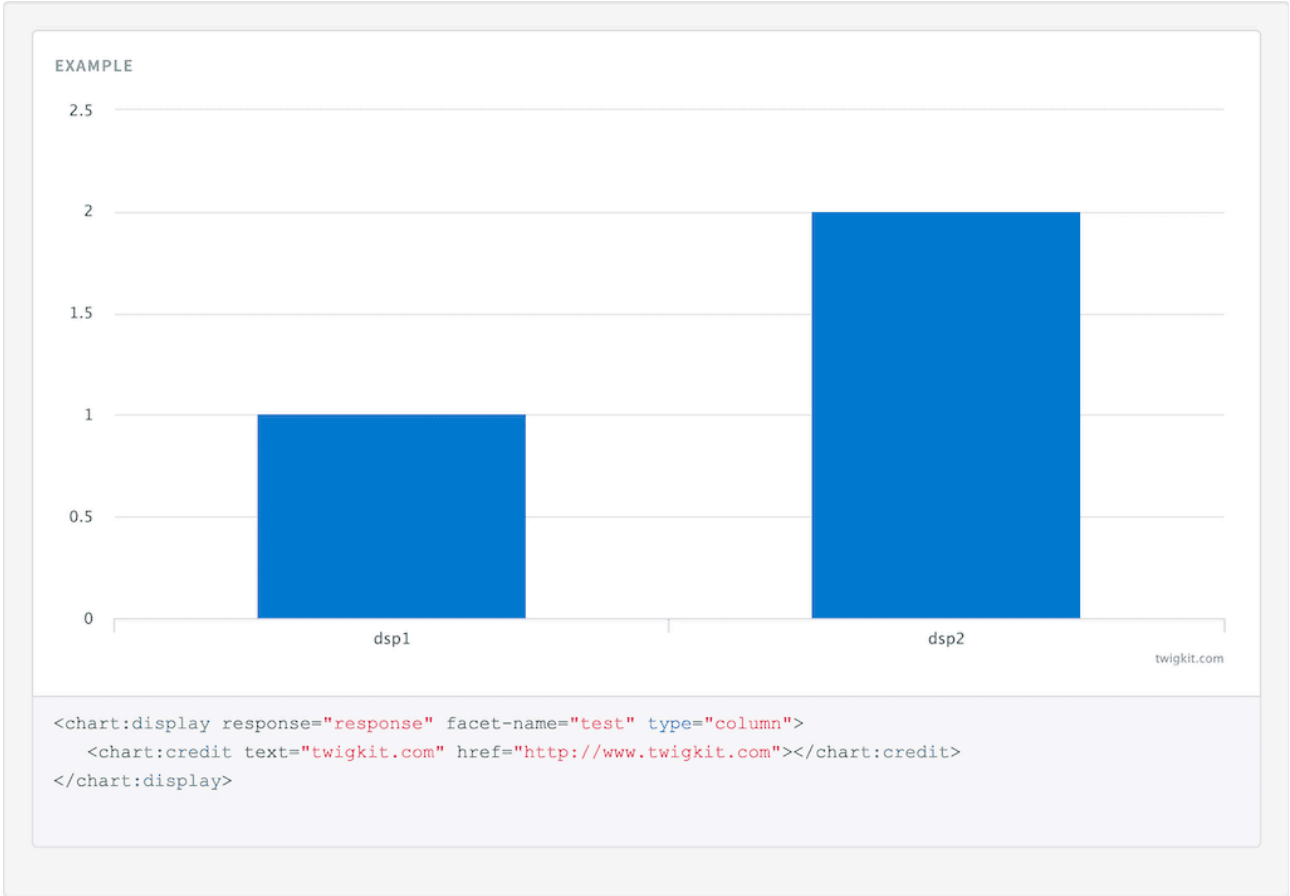


chart:display

Description

The Chart Display Tag displays the facet provided in a chart. The body of the Chart Display tag can contain a *Series Facet Tag* for rendering multiple series using facets and also a *Series Result List Tag* for rendering multiple series using results. Along with this there are other tags to customise the chart, these include the *Chart Tooltip Tag* for modifying the tooltip, *Chart Legend Tag* for modifying the legend, *Chart Credit Tag* for adding a credit to the chart and the *Chart Export Tag* for exporting the chart.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see [here](http://www.highcharts.com/docs/chart-design-and-style/style-by-css).

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

The following tags can be nested within the chart display tag.

- *Series Facet Tag*
- *Series Result List Tag*
- *Series Time Tag*
- *Chart Credit Tag*
- *Chart Export Tag*
- *Chart Legend Tag*
- *Chart Tooltip Tag*

Usage

as element:

```
<chart:display response="{string}" facet-name="{string}" type="{string}" [title="{string}"] [subtitle="{string}"] [height="{string}"] [xtitle="{string}"] [ytitle="{string}"] [stacking="{string}"] [radar-or-polar="{string}"] [xlabel-rotation="{number}"] [max-characters="{number}"] [inner-size="{number}"] [invert-axes="{boolean}"] [show-tooltips="{boolean}"] [query="{object}"] [options="{object}"] [max-labels="{number}"] [show-labels="{boolean}"] [show-count="{boolean}"] [show-data-labels="{boolean}"] [three-d="{boolean}"] [websocket="{string}"] [show-value="{boolean}"] [select-series="{boolean}"] [select="{string}"] [single-filter-mode="{boolean}"]> </chart:display>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	The name of the response.
facet-name	string	Name of facet.
type	string	Chart type (use type of 'time' when using the series time tag).
title (optional)	string	Chart title
sub title (optional)	string	Chart sub title
height (optional)	string	Chart height. Default:400 (400px but px is not required!)
x title (optional)	string	x axis title
y title (optional)	string	y axis title
stacking (optional)	string	Whether to stack the values of each series on top of each other when using multiple series. 'normal' to stack by value or 'percent'.

Param	Type	Details
radar- or- polar (optional)	string	Set as 'radar' or 'polar' to turn your series into a radar/polar chart.
xlabel- rotation (optional)	number	Rotation of the labels in degrees
max- characters (optional)	number	Number of characters to show in labels. default:250. (Cannot be used with a time series)
inner- size (optional)	number	The of size of the inner diameter of a pie chart. A size greater than 0 renders a donut chart.
invert- axes (optional)	boolean	Whether to invert the axes so that the x axis is vertical and y axis is horizontal. When true, the x axis is reversed by default. If a bar series is present in the chart, it will be inverted automatically. Defaults to false.
show- tool tips (optional)	boolean	Whether to display hover tooltips in the graph. Default:true
query (optional)	object	The search query from which to build links.

Param	Type	Details
options (optional)	object	Extend the current options. Options can be found here. (http://api.highcharts.com/highcharts/)
max-labels (optional)	number	Maximum number of labels to show
show-labels (optional)	boolean	Whether to show labels in the graph, does not affect hover tooltips, default:true
show-count (optional)	boolean	Whether to show the count in the label. default:false
show-data-labels (optional)	boolean	Whether to show the count on the point of the series (Not applicable to pie charts) default:false
three-d (optional)	boolean	Renders the chart in 3d. Only works with column and pie charts.
websocket (optional)	string	URL to receive updates from

Param	Type	Details
show-value (optional)	boolean	Whether to show the value in the middle of the chart when hovered over. This will only work for pie charts with the <code>inner-size</code> attribute set greater than 0.
select-series (optional)	boolean	When set to <code>true</code> you will be able to select the series to show on the chart.
select (optional)	string	How the data should be filtered when clicking on a part of the chart, a valid query must be present. See <i>select example</i> + Valid values:- false User selects a filter by clicking the value; (default) * multiselect Supports selecting multiple filters that get combined with an 'OR'. * multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.
single-filter-mode (optional)	boolean	When in single-filter mode, a facet visualisation will be shown as a single number, in large print, when the facet being rendered has only a single filter value. This is to avoid showing a single bar, column or pie slice when rendering a visualisation over a single data bucket. See <i>single filter mode example</i>

Example

Source

```
<div ng-controller="ExampleController"> <div> <h3 id="example_chart-types">Chart
Types</h3> <toggle:controls labels="Bar, Column, Pie, Donut, Donut 2, Line, Area,
Area Spline" panes="panes"></toggle:controls> <toggle:panes id="panes">
<toggle:pane> <chart:display response="response" facet-name="population"
type="bar" title="Population of Countries" subtitle="Bar Chart"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export
file-name="bar"></chart:export> </chart:display> </toggle:pane> <toggle:pane>
<chart:display response="response" facet-name="population" type="column"
title="Population of Countries" subtitle="Column Chart"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export
file-name="column"></chart:export> </chart:display> </toggle:pane> <toggle:pane>
<chart:display response="response" facet-name="population" type="pie"
title="Population of Countries" subtitle="Pie Chart"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export
file-name="pie"></chart:export> </chart:display> </toggle:pane> <toggle:pane>
<chart:display response="response" facet-name="population" type="pie" inner-
size="80%" show-tooltips="false" show-value="true" title="Population of
Countries" subtitle="Donut Chart"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="donut">
</chart:export> </chart:display> </toggle:pane> <toggle:pane> <chart:display
response="response" facet-name="population" type="donut" title="Population of
Countries" subtitle="Donut Chart 2"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="donut">
</chart:export> </chart:display> </toggle:pane> <toggle:pane> <chart:display
response="response" facet-name="population" type="line" title="Population of
Countries" subtitle="Line Chart"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="line">
</chart:export> </chart:display> </toggle:pane> <toggle:pane> <chart:display
response="response" facet-name="population" type="area" title="Population of
Countries" subtitle="Area Chart"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="area">
</chart:export> </chart:display> </toggle:pane> <toggle:pane> <chart:display
response="response" facet-name="population" type="areaspline" title="Population
of Countries" subtitle="Area Spline Chart"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-
name="areaspline"></chart:export> </chart:display> </toggle:pane> </toggle:panes>
</div> <hr/> <div> <h3 id="example_radar-chart-type">Radar Chart Type</h3>
<toggle:controls labels="Bar, Line, Area" panes="radar"></toggle:controls>
<toggle:panes id="radar"> <toggle:pane> <chart:display response="response" facet-
name="population" type="bar" title="Population of Countries" subtitle="Bar Chart"
radar-or-polar="radar"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="bar">
</chart:export> </chart:display> </toggle:pane> <toggle:pane> <chart:display
response="response" facet-name="population" type="line" title="Population of
Countries" subtitle="Line Chart" radar-or-polar="radar"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export
file-name="line"></chart:export> </chart:display> </toggle:pane> <toggle:pane>
<chart:display response="response" facet-name="population" type="area"
title="Population of Countries" subtitle="Area Chart" radar-or-polar="radar">
<chart:credit text="twigkit.com" href="http://www.twigkit.com"></chart:credit>
<chart:export file-name="area"></chart:export> </chart:display> </toggle:pane>
</toggle:panes> </div> <hr/> <div> <h3 id="example_polar-chart-type">Polar Chart
Type</h3> <toggle:controls labels="Bar, Line, Area" panes="polar">
</toggle:controls> <toggle:panes id="polar"> <toggle:pane> <chart:display
response="response" facet-name="population" type="bar" title="Population of
Countries" subtitle="Bar Chart" radar-or-polar="polar"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export
file-name="bar"></chart:export> </chart:display> </toggle:pane> <toggle:pane>
<chart:display response="response" facet-name="population" type="line"
title="Population of Countries" subtitle="Line Chart" radar-or-polar="polar">
```

```

<chart:credit text="twigkit.com" href="http://www.twigkit.com"></chart:credit>
<chart:export file-name="line"></chart:export> </chart:display> </toggle:pane>
<toggle:pane> <chart:display response="response" facet-name="population"
type="area" title="Population of Countries" subtitle="Area Chart" radar-or-
polar="polar"> <chart:credit text="twigkit.com" href="http://www.twigkit.com">
</chart:credit> <chart:export file-name="area"></chart:export> </chart:display>
</toggle:pane> </toggle:panes> </div> <hr/> <div> <h3 id="example_axis-
titles">Axis Titles</h3> <chart:display response="response" facet-
name="population" type="bar" title="Population of Countries" xtitle="Country"
ytitle="Population"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> <chart:export file-name="column">
</chart:export> </chart:display> </div> <hr/> <div> <h3 id="example_3d">3D</h3>
<chart:display response="response" facet-name="population" type="pie"
title="Population of Countries" three-d="true"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> <hr/>
<chart:display response="response" facet-name="population" type="column"
title="Population of Countries" three-d="true"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> </div> <hr/> <div>
<h3 id="example_label-rotation">Label Rotation</h3> <chart:display
response="response" facet-name="population" type="column" title="Population of
Countries" xlabel-rotation="45"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> <hr/>
<chart:display response="response" facet-name="population" type="column"
title="Population of Countries" xlabel-rotation="-45"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> </chart:display>
</div> <hr/> <div> <h3 id="example_show-labels">Show Labels (false)</h3>
<chart:display response="response" facet-name="population" type="pie"
title="Population of Countries" show-labels="false"> <chart:credit
text="twigkit.com" href="http://www.twigkit.com"></chart:credit> </chart:display>
<hr/> <chart:display response="response" facet-name="population" type="pie"
inner-size="80%" show-value="true" show-tooltips="false" title="Population of
Countries" show-labels="false"> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </div> <div> <h3
id="example_custom">Custom (<a href="http://api.highcharts.com/highcharts/"
target="_blank">Options</a>)</h3> <chart:display response="response" type="line"
title="Population of Countries" xlabel-rotation="-45" options="custom">
<series:facet response="response" name="population" title="Population Present
Day"></series:facet> <series:facet response="response" name="population2000"
title="Population 2000"></series:facet> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> <h4
id="example_custom_zoom">Zoom (Custom)</h4> <p>Custom Object</p> <pre>
{{zoomCustom | json}}</pre> <p>Use your mouse to select the area you wish to zoom
in on</p> <chart:display response="response" type="line" title="Population of
Countries" options="zoomCustom"> <series:facet response="response"
name="population" title="Population Present Day"></series:facet> <series:facet
response="response" name="population2000" title="Population 2000"></series:facet>
<chart:credit text="twigkit.com" href="http://www.twigkit.com"></chart:credit>
</chart:display> </div> <div id="colour-demo"> <h3 id="example_colour">Colour
(since 0.3.0)</h3> <p>Add an ID to the element wrapping your chart and configure
the background colour and series colours like so.</p> <pre> #colour-demo
.highcharts-color-0{ fill:pink; stroke:pink; } #colour-demo .highcharts-color-1{
fill:yellow; stroke:yellow; } #colour-demo .highcharts-background{ fill:#f5f0f0;
} </pre> <p>For more detail on what you can change and set using CSS see <a
href="http://www.highcharts.com/docs/chart-design-and-style/style-by-css"
target="_blank">here.</a></p> <chart:display response="response" type="line"
title="Population of Countries" xlabel-rotation="-45" options="custom">
<series:facet response="response" name="population" title="Population Present
Day"></series:facet> <series:facet response="response" name="population2000"
title="Population 2000"></series:facet> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> </div> <div> <h3
id="example_selectable-series">Selectable Series</h3> <chart:display
response="response" type="line" title="Population of Countries" xlabel-
rotation="-45" select="checkbox"> <series:facet response="response"
name="population" title="Population Present Day" selected-by-default="false">
</series:facet> <series:facet response="response" name="population2000"

```

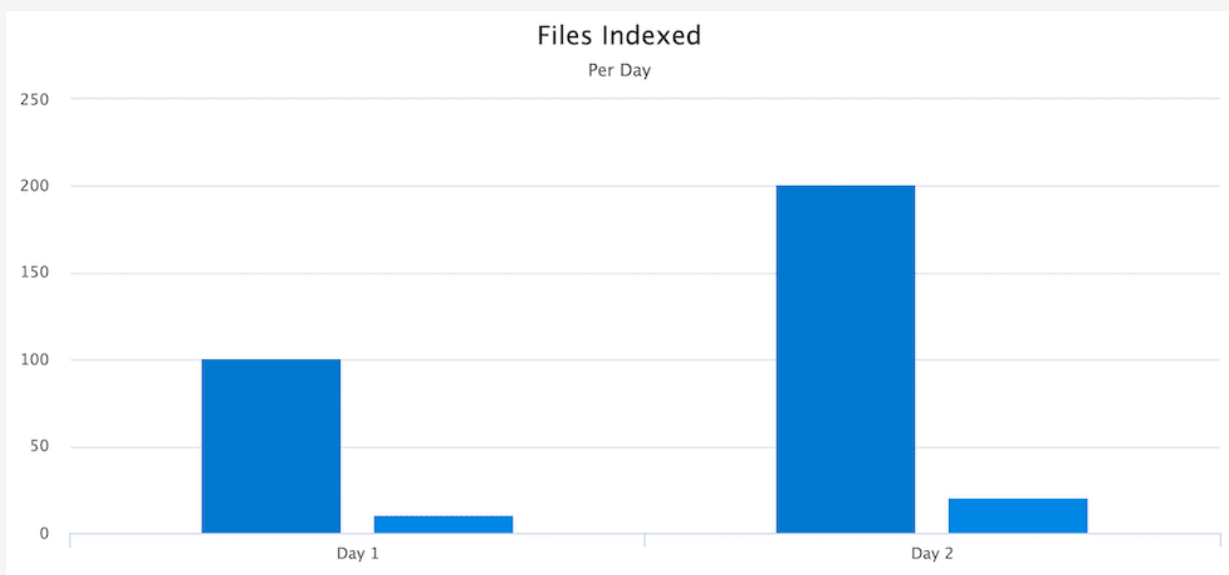
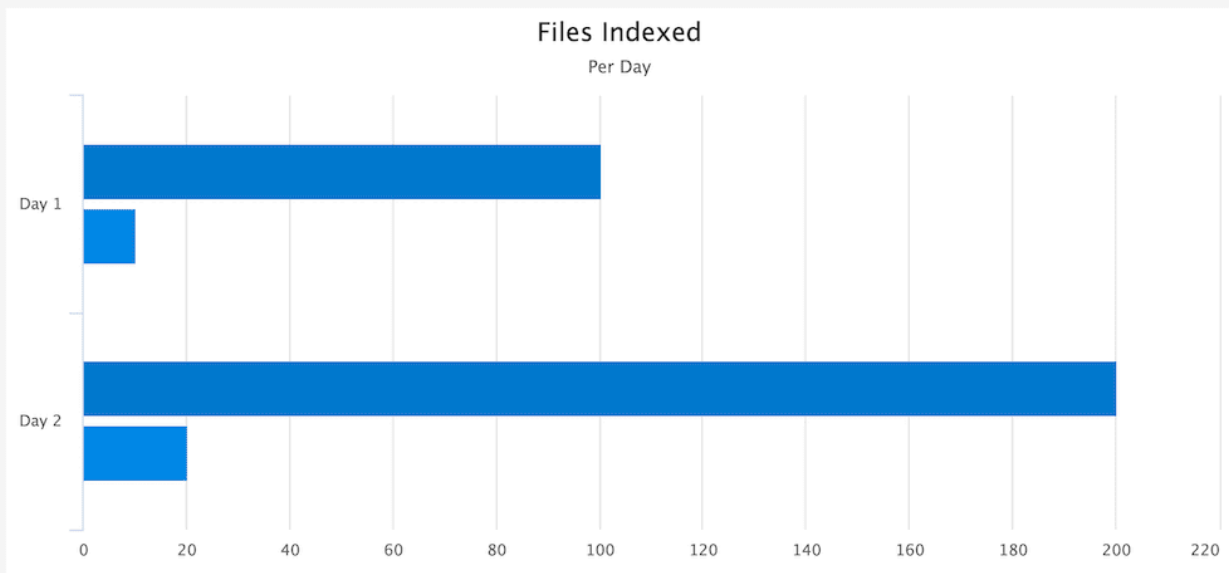


```
title="Population 2000"></series:facet> <chart:credit text="twigkit.com"
href="http://www.twigkit.com"></chart:credit> </chart:display> </div> </div>
```

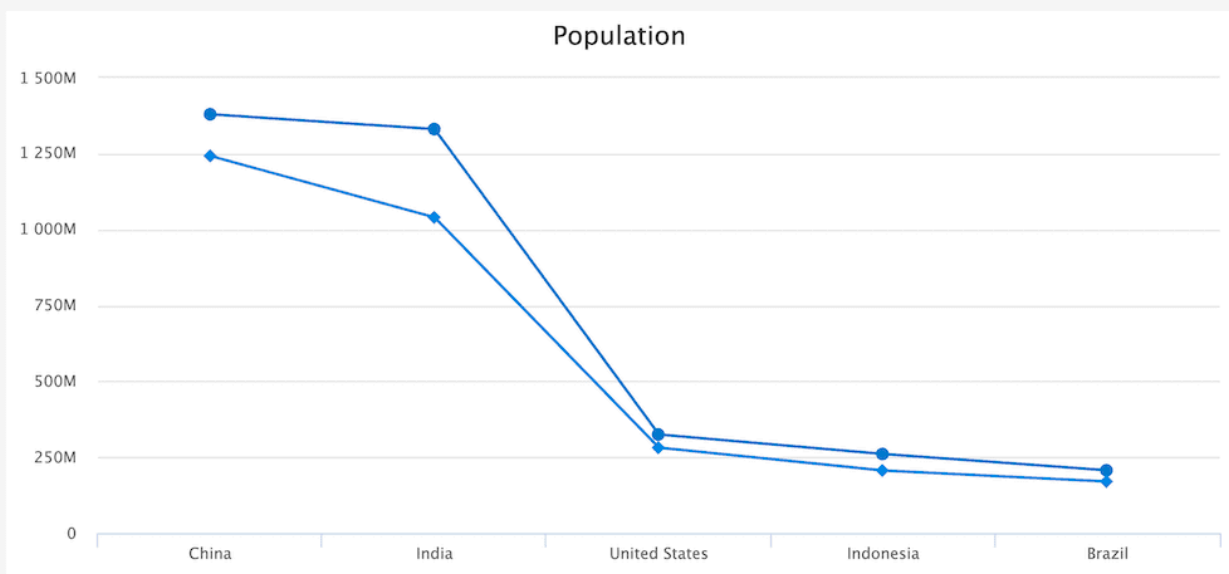
```
#colour-demo .highcharts-color-0{ fill:pink; stroke:pink; colour-demo
.highcharts-color-1{ fill:yellow; stroke:yellow; colour-demo .highcharts-
background{ fill:#f5f0f0;
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout', 'ResponseService', function($scope,$timeout,
ResponseService) { $scope.custom = { tooltip:{ shared:true, split:true }, xAxis:
{ plotLines: [{ value: 3, // Value of where the line will appear width: 1 //
Width of the line }] } }; $scope.zoomCustom = { chart:{ zoomType:'xy' } }
$scope.response = { page: 2, query: { rpp: 50 }, facets: { test: { filters: [ {
val: { dsp: 'dsp1', act: 1 }, count: 1, }, { val: { dsp: 'dsp2', act: 2 }, count:
2, } ] }, population:{ filters: [ { val: 'China', count: 1379790000, }, { val:
'India', count: 1330780000, }, { val: 'United States', count: 324897000, }, {
val: 'Indonesia', count: 260581000, }, { val: 'Brazil', count: 206927000, }, {
val: 'Pakistan', count: 194772000, }, { val: 'Nigeria', count: 186987000, }, {
val: 'Bangladesh', count: 161401000, }, { val: 'Russia', count: 146727405, }, {
val: 'Japan', count: 126693000, }, ] }, population2000:{ filters: [ { val:
'China', count: 1242612300, }, { val: 'India', count: 1040000000, }, { val:
'United States', count: 281421923, }, { val: 'Indonesia', count: 206264595, }, {
val: 'Brazil', count: 170000000, }, { val: 'Pakistan', count: 140000000, }, {
val: 'Nigeria', count: 110000000, }, { val: 'Bangladesh', count: 130000000, }, {
val: 'Russia', count: 147000000, }, { val: 'Japan', count: 127000000, }, ] },
donut: { filters: [ { val: { dsp: 'Short', act: 1 }, count: 1, }, { val: { dsp:
'Long Donut Name(15)', act: 2 }, count: 2, }, { val: { dsp: 'Long Long Donut
Name', act: 10 }, count: 10, }, { val: { dsp: 'Long Long Long Donut Name', act:
20 }, count: 20, }, { val: { dsp: 'Long Long Long Long Donut Name', act: 20 },
count: 20, }, { val: { dsp: 'Long Long Long Long Long Donut Name', act: 20 },
count: 20, }, { val: { dsp: 'Long Long Long Long Long Long Donut Name', act: 20
}, count: 20, } ] }, }, results: [ { result: { fields: { label:{ val:'label1', }
, value:{ val:1, }, anotherValue:{ val:1, }, } }, }, { result: { fields: { label:
{ val:'label2', }, value:{ val:2, }, anotherValue:{ val:100, }, } }, hits: {
act: ['Hits', 264] } } ] }
ResponseService.setResponse('response',$scope.response); }]);
```

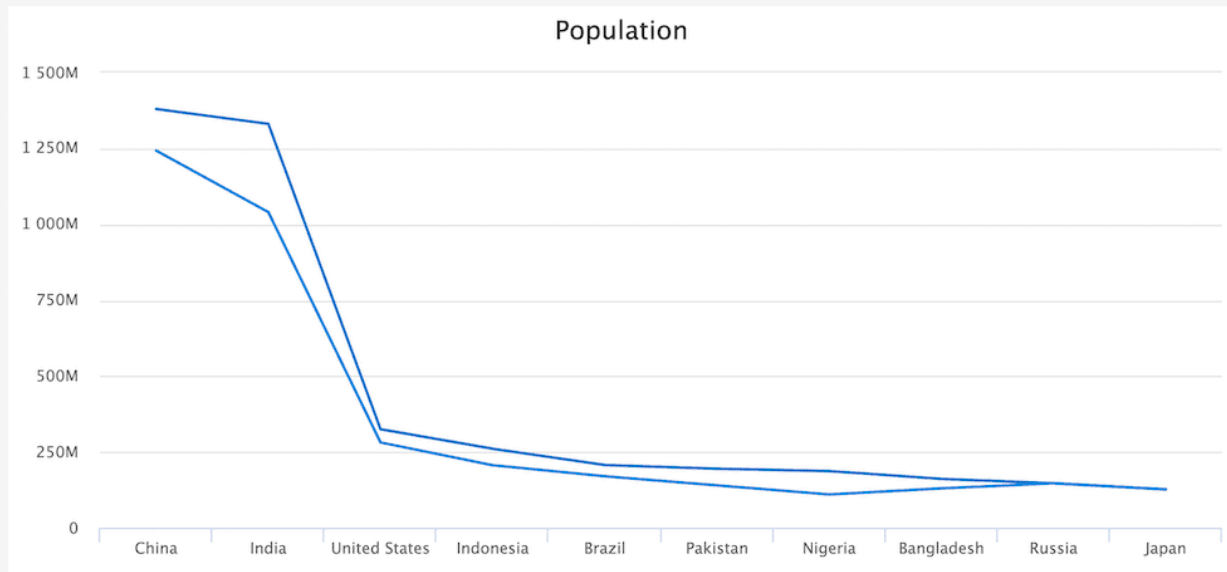
Result



Number To Show Attribute



Show Marker Attribute



Colour (since 0.3.0)

Add an ID to the element wrapping your chart and configure the background colour and series colours like so.

```
#colour-demo .highcharts-color-0{
  fill:pink;
  stroke:pink;
}
#colour-demo .highcharts-color-1{
  fill:yellow;
  stroke:yellow;
}
#colour-demo .highcharts-background{
  fill:#f5f0f0;
}
```

For more detail on what you can change and set using CSS see [here](#).

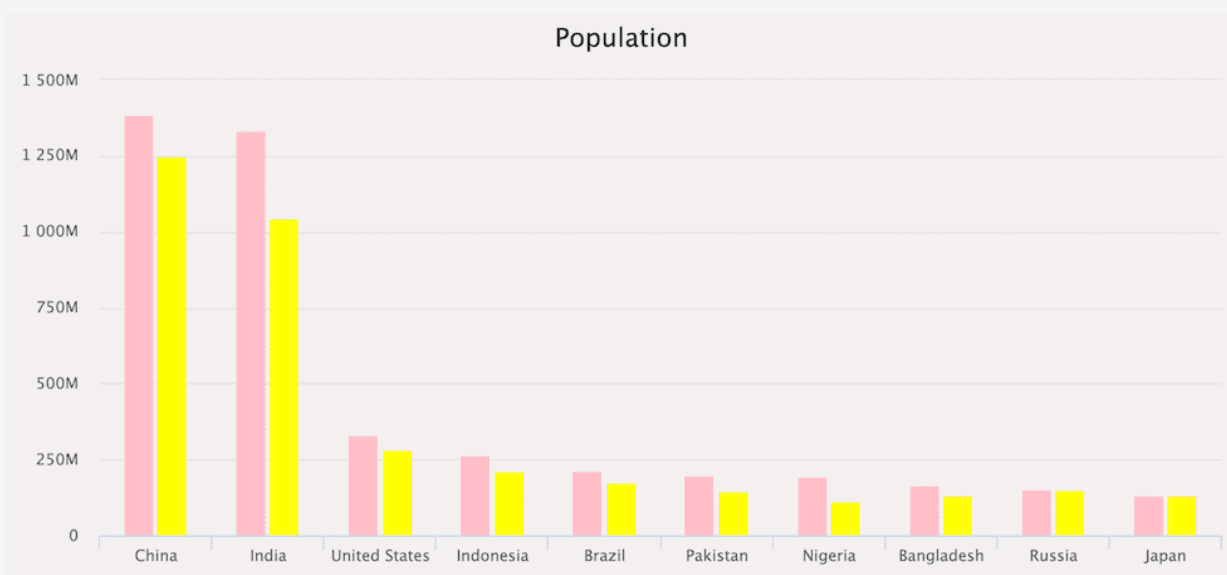


chart:export

Description

To give users the option to export a chart include the Chart Export tag within the Chart Display tag. Adding the tag will enable exporting without adding any attributes. This tag must be nested in the *Chart Display* tag.

Usage

as element:

```
<chart:export [file-name="{string}"] [url="{string}"]> </chart:export>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
file-name (optional)	string	Custom name for the file when downloaded, if nothing is entered it will default to 'chart'.
url (optional)	string	Url of the export server, if left blank it will use the providers export server.

Example

Source

```
<chart:display response="response" facet-name="test" type="column"> <chart:export  
file-name="column"></chart:export> </chart:display>
```

```
angular.module('lightning') .controller('ExampleController',  
 ['$scope', '$timeout', 'ResponseService', function($scope, $timeout, ResponseService)  
 { $scope.response = { page: 2, query: { rpp: 50 }, facets: { test: { filters: [ {  
 val: { dsp: 'dsp1', act: 1 }, count: 1, }, { val: { dsp: 'dsp2', act: 2 }, count:  
 2, } ] }, }, results: [ { result: { fields: { label: 'label1', value: 1 } }, }, {  
 result: { fields: { label: 'label2', value: 2 } }, hits: { act: ['Hits', 264] }  
 } ] } ResponseService.setResponse('response', $scope.response); }]);
```

Result

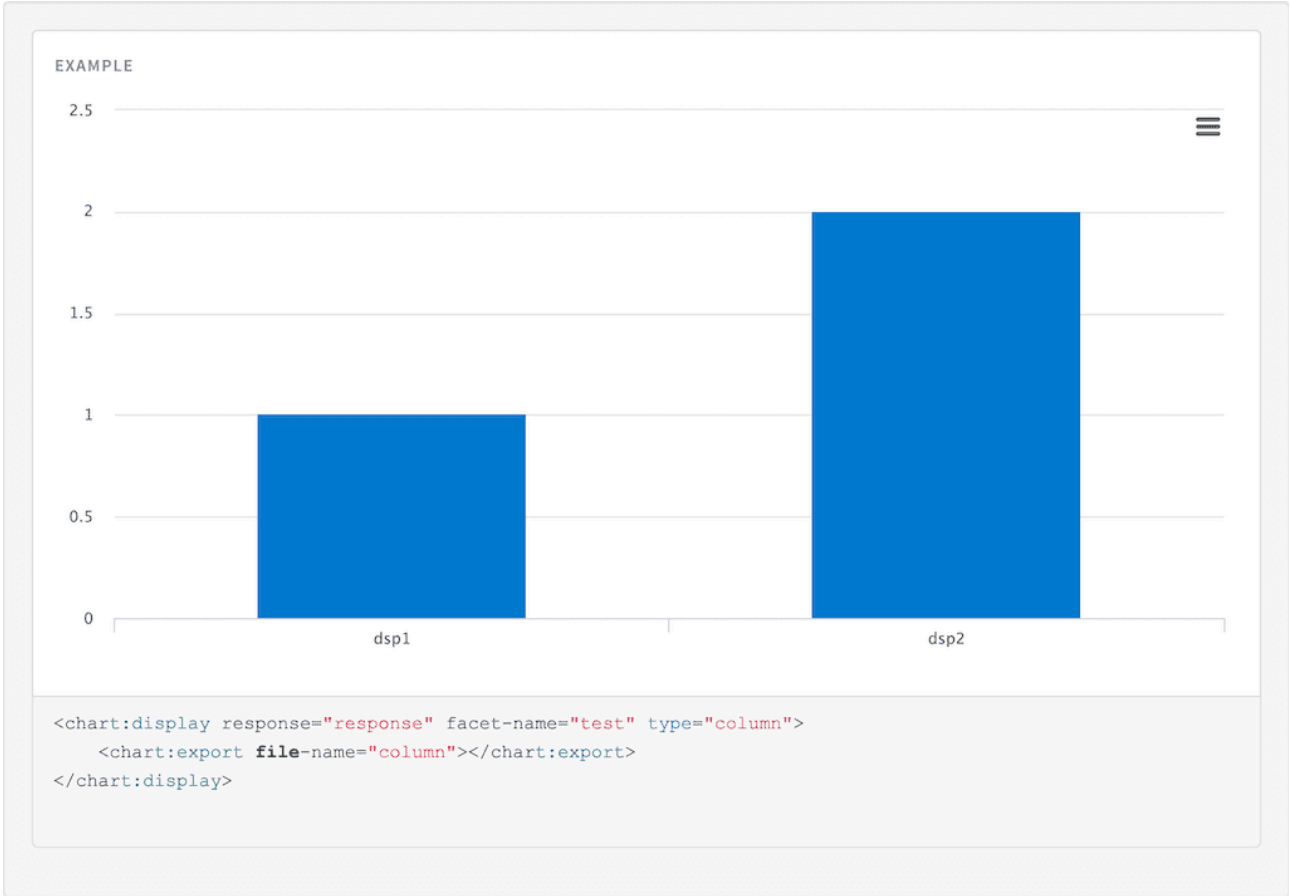


chart:gauge

Description

The chart gauge directive generates a gauge, you can also use the chart export and chart credit directives to add this functionality to your gauge.

Usage

as element:

```
<chart:gauge value="{number}" min="{number}" max="{number}" [stops="{array}"]  
[title="{string}"] [value-suffix="{string}"] [height="{string}"] [options="  
{object}"]> </chart:gauge>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
value	number	Value to be displayed on the gauge.
min	number	The smallest number that can be displayed on the gauge.
max	number	The highest number that can be displayed on the gauge.
stops (optional)	array	The colours to render at each stop measured from 0 to 1. Default [[0.1, '#DF5353'], // red [0.5, '#DDDF0D'], // yellow [0.9, '#55BF3B'] // green]
title (optional)	string	The title of the chart.
value-suffix (optional)	string	Suffix for the value on the gauge.
height (optional)	string	Chart height. Default:400 (400px but px is not required!)
options (optional)	object	Extend the current options. Options can be found here. (http://api.highcharts.com/highcharts/)

Example

Source

```
<div ng-controller="ExampleController"> <chart:gauge value="{{value}}" min="{{min}}" max="{{max}}" title="Speed" value-suffix="mph" stops="[[0.1, '#55BF3B'], [0.5, '#DDDF0D'], [0.9, '#DF5353']]"> <chart:credit text="twigkit.com" href="http://www.twigkit.com"></chart:credit> <chart:export file-name="gauge"></chart:export> </chart:gauge> <form class="tk-stl-form tk-stl-form-aligned"> <fieldset> <div class="tk-stl-control-group"> <label for="value">Value</label> <input id="value" type="number" placeholder="Value" ng-model="value"> <button ng-click="generateNewNumber()" style="button tiny">Generate Random Value</button> </div> <div class="tk-stl-control-group"> <label for="min">Min</label> <input id="min" type="number" placeholder="Min" ng-model="min"> </div> <div class="tk-stl-control-group"> <label for="max">Max</label> <input id="max" type="number" placeholder="Max" ng-model="max"> </div> </fieldset> </form> </div>
```

```
angular.module('lightning') .controller('ExampleController',  
['$scope','$timeout', function($scope,$timeout) { $scope.value = 20; $scope.min = 0; $scope.max = 40 $scope.generateNewNumber = function(){ $scope.value = Math.floor(Math.random() * ($scope.max - $scope.min + 1)) + $scope.min; }; }]);
```

Result

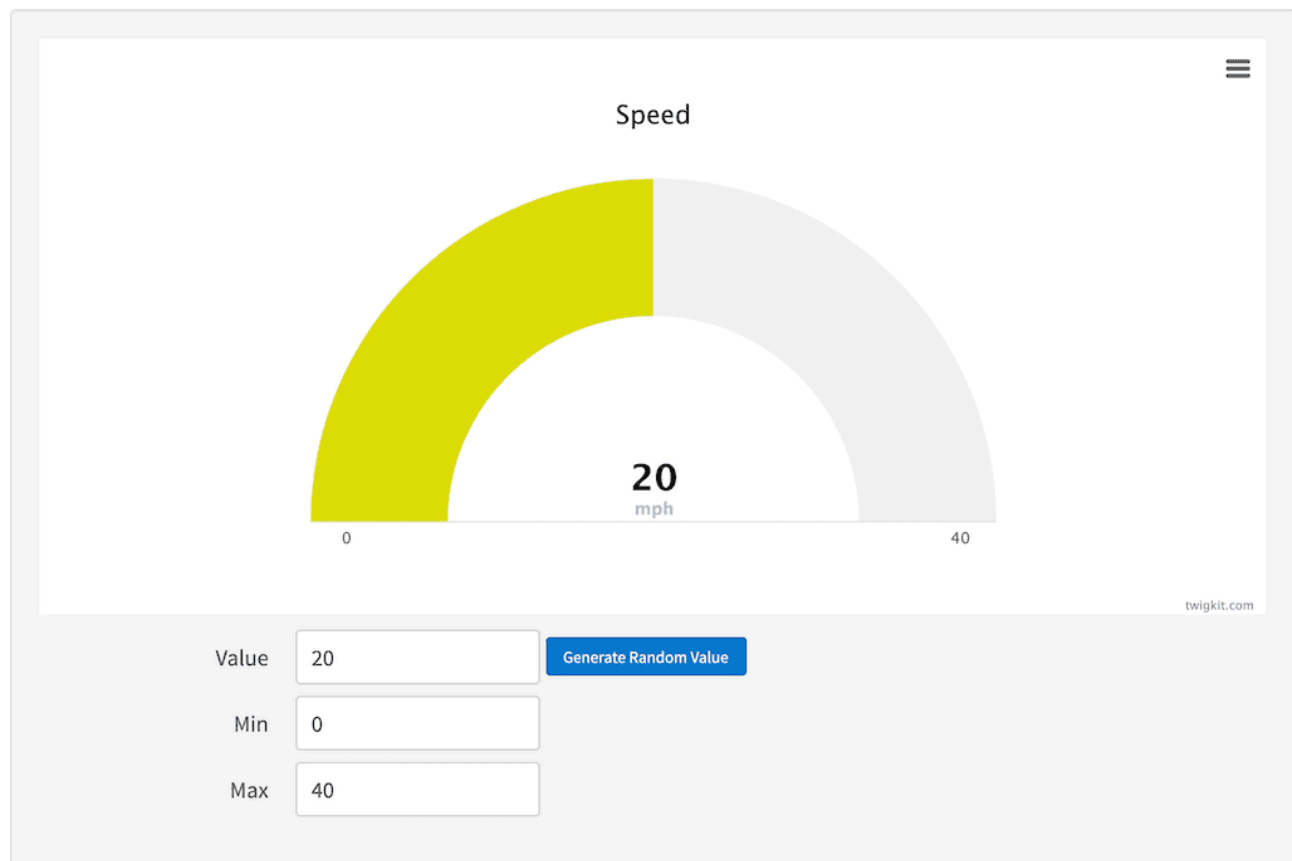


chart:histogram

Description

The Chart Histogram Directive

Usage

as element:

```
<chart:histogram [response="{object}"] [facet-name="{string}"] [query="{string}"]  
[title="{string}"] [show="{number}"] [height="{string}"] [show-count="{boolean}"]  
[max-characters="{number}"] [action="{string}"] [date-format="{string}"]>  
</chart:histogram>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response (optional)	object	A response object
facet-name (optional)	string	Facet name to use from the response object
query (optional)	string	The name of the query to build links from.
title (optional)	string	The title that should be displayed above the histogram.
show (optional)	number	The maximum number of bars to show in the histogram. Defaults to 20.
height (optional)	string	Height of the chart specified by a valid CSS height style value (e.g. '100px' or '7em'). Defaults to '100px'
show-count (optional)	boolean	Whether to display the count in the label for each bar
max-character s (optional)	number	The number of characters to show before truncating the label. Default: 20.
action (optional)	string	The target resource to which the histogram query should submit.
date-format (optional)	string	How to format the date. See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.

Example

Source

```
<div ng-controller="ExampleController"> <h3 id="example_histogram">Histogram</h3>
<chart:histogram response="response" facet-name="test1"></chart:histogram> <hr />
<h3 id="example_histogram-with-date-formatting">Histogram with Date
Formatting</h3> <chart:histogram response="response" facet-name="test3" date-
format="fullDate"></chart:histogram> <hr /> <h3 id="example_large-dateset">Large
Dateset</h3> <chart:histogram response="response" facet-name="date" date-
format="fullDate" show="99" show-count="true"></chart:histogram> <widget:slider
response="response" facet-name="date" date-format="fullDate" sort="false">
</widget:slider> </div>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50, filters:[] }, facets: { test1: {
filters: [ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val: { dsp:
'Day 2', act: 'Day 2', }, count: 200, } ], 'max-count':'200' }, test2: { filters:
[ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 10, }, { val: { dsp: 'Day 2',
act: 'Day 2', }, count: 20, } ], 'max-count':20 }, test3: { filters: [ { val: {
dsp: 1288323623006, act: 1288323623006, }, count: 10, }, { val: { dsp:
1288323723006, act: 1288323723006, }, count: 20, } ], 'max-count':20 }, date:{
filters:[], 'max-count':100 } }, results: [] } var i = 0; while(i < 99){ var
CurrentDate = new Date(); CurrentDate.setMonth(CurrentDate.getMonth() + i);
$scope.response.facets.date.filters.push( { val : { dsp:CurrentDate,
act:CurrentDate }, count:Math.floor(Math.random()*(100-1+1)+1), } ) i++ }
console.log($scope.response.facets.date); ResponseService.setResponse('response',
$scope.response); }]);
```

Result

Histogram



Histogram with Date Formatting



Large Datasets

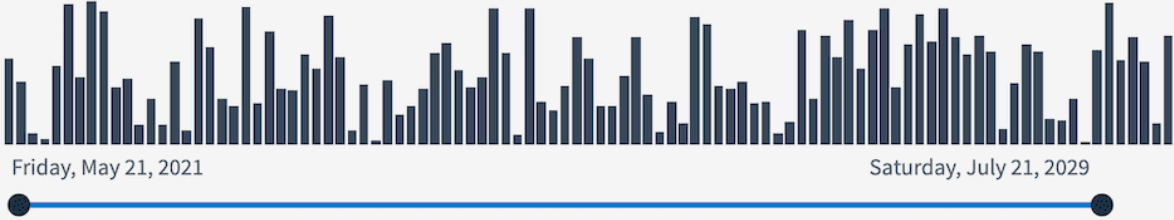


chart:legend

Description

A legend is displayed when the chart has multiple series, with the Chart Legend tag you can customise the legend with a title, border and location on the chart along with other things. This tag must be nested in the *Chart Display tag*.

Usage

as element:

```
<chart:legend [title="{string}"] [align="{string}"] [vertical-align="{string}"]  
[layout="{string}"] [x="{string}"] [y="{string}"]> </chart:legend>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
title (optional)	string	The title of the legend.
align (optional)	string	The horizontal alignment of the legend box within the chart area. Valid values are 'left', 'center' and 'right'.
vertical-align (optional)	string	The vertical alignment of the legend box. Valid values are 'top', 'middle' or 'bottom'.
layout (optional)	string	The layout of the legend items. Can be one of 'horizontal' or 'vertical'.
x (optional)	string	The x offset of the legend relative to its horizontal alignment 'align'. Negative x moves it to the left, positive x moves it to the right.
y (optional)	string	The vertical offset of the legend relative to its vertical alignment 'verticalAlign'. Negative y moves it up, positive y moves it down.

Example

Source

```
<div ng-controller="ExampleController"> <h3 id="example_legend">Legend</h3> <h4
id="example_legend_legend-on-the-right">Legend on the Right</h4> <tk-example>
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
<series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
<chart:legend title="Legend Title" align="right" vertical-align="top"
layout="vertical" y="100"></chart:legend> </chart:display> </tk-example> <tk-
code-wrapper> <code hljs> <chart:display response="response" title="Files
Indexed" subtitle="Per Day"> <series:facet name="test1" type="column"
title="First Pass"></series:facet> <series:facet name="test2" type="column"
title="Second Pass"></series:facet> <chart:legend title="Legend Title"
align="right" vertical-align="top" layout="vertical" y="100"></chart:legend>
</chart:display> </code> </tk-code-wrapper> <hr/> <h4 id="example_legend_legend-
on-the-left">Legend on the Left</h4> <tk-example> <chart:display
response="response" title="Files Indexed" subtitle="Per Day"> <series:facet
name="test1" type="column" title="First Pass"></series:facet> <series:facet
name="test2" type="column" title="Second Pass"></series:facet> <chart:legend
title="Legend Title" align="left" vertical-align="top" layout="vertical" y="100">
</chart:legend> <chart:legend title="Legend Title" align="left" vertical-
align="top" layout="vertical" y="100"></chart:legend> </chart:display> </tk-
example> <tk-code-wrapper> <code hljs> <chart:display response="response"
title="Files Indexed" subtitle="Per Day"> <series:facet name="test1"
type="column" title="First Pass"></series:facet> <series:facet name="test2"
type="column" title="Second Pass"></series:facet> <chart:legend title="Legend
Title" align="right" vertical-align="top" layout="vertical" y="100">
</chart:legend> </chart:display> </code> </tk-code-wrapper> <hr/> <h4
id="example_legend_legend-on-the-bottom">Legend on the Bottom</h4> <tk-example>
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
<series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
<chart:legend title="Legend Title" align="center" vertical-align="bottom"
layout="horizontal"></chart:legend> </chart:display> </tk-example> <tk-code-
wrapper> <code hljs> <chart:display response="response" title="Files Indexed"
subtitle="Per Day"> <series:facet name="test1" type="column" title="First Pass">
</series:facet> <series:facet name="test2" type="column" title="Second Pass">
</series:facet> <chart:legend title="Legend Title" align="center" vertical-
align="bottom" layout="vertical"></chart:legend> </chart:display> </code> </tk-
code-wrapper> <hr/> <h4 id="example_legend_adding-a-border">Adding a Border</h4>
<p>Add an ID or class to the element wrapping your chart and configure the stroke
width to add a border and the value rx to modify the border radius.</p> <pre>
.chart-with-legend-border .highcharts-legend-box { fill: none; stroke-width: 1px;
stroke: black; rx:2; } </pre> <div class="chart-with-legend-border"> <tk-example>
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
<series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
<chart:legend title="Legend Title" align="right" vertical-align="middle"
layout="vertical"></chart:legend> </chart:display> </tk-example> <tk-code-
wrapper> <code hljs> <chart:display response="response" title="Files Indexed"
subtitle="Per Day"> <series:facet name="test1" type="column" title="First Pass">
</series:facet> <series:facet name="test2" type="column" title="Second Pass">
</series:facet> <chart:legend title="Legend Title" align="right" vertical-
align="middle" layout="vertical"></chart:legend> </chart:display> </code> </tk-
code-wrapper> </div> </div>
```

```
.chart-with-legend-border .highcharts-legend-box { fill: none; stroke-width: 1px; stroke: black; rx:2; }
```

```
angular.module('lightning') .controller('ExampleController',  
  ['$scope','$timeout','ResponseService', function($scope,$timeout,  
  ResponseService) { $scope.response = { page: 2, query: { rpp: 50 }, facets: {  
  test1: { filters: [ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val:  
  { dsp: 'Day 2', act: 'Day 2', }, count: 200, } ] }, test2: { filters: [ { val: {  
  dsp: 'Day 1', act: 'Day 1', }, count: 10, }, { val: { dsp: 'Day 2', act: 'Day 2',  
  }, count: 20, } ] }, }, results: [ { result: { fields: { label: 'label1', value:  
  1 } }, }, { result: { fields: { label: 'label2', value: 2 } }, hits: { act:  
  ['Hits', 264] } } ] } ResponseService.setResponse('response',$scope.response);  
  }]);
```


Result

Legend

Legend on the Right

EXAMPLE



```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">  
  <series:facet name="test1" type="column" title="First Pass"></series:facet>  
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>  
  <chart:legend title="Legend Title" align="right" vertical-align="top" layout="vertical"  
y="100"></chart:legend>  
</chart:display>
```

Legend on the Left

EXAMPLE

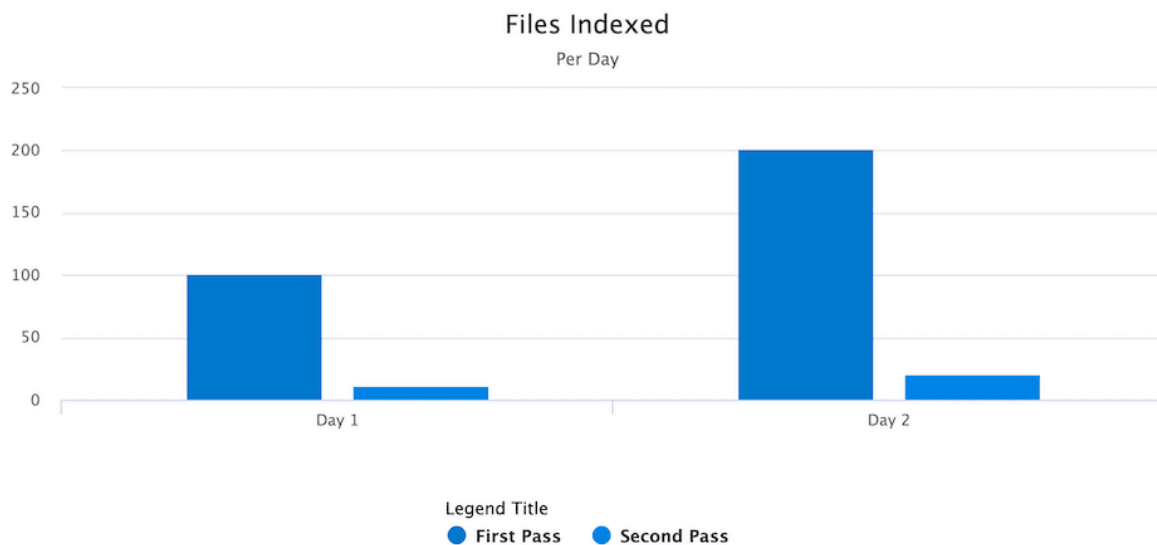


```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">  
  <series:facet name="test1" type="column" title="First Pass"></series:facet>  
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>  
  <chart:legend title="Legend Title" align="right" vertical-align="top" layout="vertical"  
y="100"></chart:legend>  
</chart:display>
```

```
y="100"></chart:legend>
</chart:display>
```

Legend on the Bottom

EXAMPLE



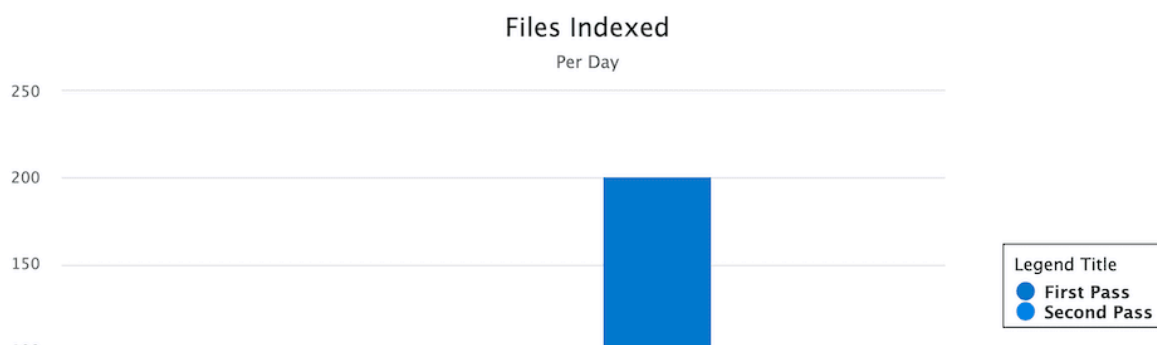
```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
  <series:facet name="test1" type="column" title="First Pass"></series:facet>
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>
  <chart:legend title="Legend Title" align="center" vertical-align="bottom"
  layout="vertical"></chart:legend>
</chart:display>
```

Adding a Border

Add an ID or class to the element wrapping your chart and configure the stroke width to add a border and the value rx to modify the border radius.

```
.chart-with-legend-border .highcharts-legend-box {
  fill: none;
  stroke-width: 1px;
  stroke: black;
  rx:2;
}
```

EXAMPLE





```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
  <series:facet name="test1" type="column" title="First Pass"></series:facet>
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>
  <chart:legend title="Legend Title" align="right" vertical-align="middle"
  layout="vertical"></chart:legend>
</chart:display>
```

chart:tagcloud

Description

The Chart Tag Cloud Directive

Usage

as element:

```
<chart:tagcloud response="{object}" facet-name="{string}" [title="{string}"]  
[show="{number}"] [action="{string}"] [order="{string}"]> </chart:tagcloud>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object
facet-name	string	Facet name to use from the response object
title (optional)	string	The title that should be displayed above the tag cloud.
show (optional)	number	The maximum number of values to show in the tag cloud. Defaults to 15.
action (optional)	string	The target resource to which the tag cloud query should submit.
order (optional)	string	Key to order the data by, accepted values are <code>val</code> , <code>-val</code> , <code>-count</code> , <code>-weight</code> , <code>count</code> and <code>weight</code> .

chart:timeline

Description

The Timeline directive

Usage

as element:

```
<chart:timeline response="{object}" date="{string}" event="{string}" [milestone-  
grouping="{string}"] [orientation="{string}"] [max-width="{string}"] [max-  
height="{string}"] [query="{object}"] [platform="{string}"]> </chart:timeline>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object
date	string	The result field to use as a date for the timeline
event	string	The result field to use for the event content
milestoneGrouping (optional)	string	How to group the milestones i.e. year (yyyy), month (MM)
orientation (optional)	string	The orientation of the timeline. (horizontal or vertical. Default vertical)
maxWidth (optional)	string	Set max width of timeline container
maxHeight (optional)	string	Set max height of timeline container
query (optional)	object	Query object to fetch more results
platform (optional)	string	Platform to query for more results

chart:tooltip

Description

Tooltip Tag

Tooltips display information when you hover over a point or area of the chart, tooltips can be set across the whole chart or specifically to a series using the Chart Tooltip tag. This tag must be nested in the *Chart Display tag* or the series tags (*Series Facet Tag*, *Series Result List Tag* or *Series Time Tag*) if the tooltip is specific to a series.

Shared Tooltip

If you are rendering more than one series you can set the tooltip to be shared, this means it will display information about all the series for the point you've hovered over.

Usage

as element:

```
<chart:tooltip [header-format="{string}"] [point-format="{string}"] [shared="{boolean}"]> </chart:tooltip>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
header-format (optional)	string	Custom formatting of the data point tooltips header. Can make use of expressions '{point.name}', '{point.x}' and '{point.y}'. For example: '{point.x} cm, {point.y} kg'.
point-format (optional)	string	Custom formatting of the data point tooltips. Can make use of expressions '{point.name}', '{point.x}' and '{point.y}'. For example: '{point.x} cm, {point.y} kg'.
shared (optional)	boolean	Whether the tooltip should show the point details of every series (only work when placed within the Chart Display tag).

Example

Source

```
<div ng-controller="ExampleController"> <h3 id="example_tooltip-tag">Tooltip
Tag</h3> <p> When setting a tooltip you have access to two objects called <a
href="http://api.highcharts.com/highcharts/Point" target="_blank">Point</a> and
<a href="http://api.highcharts.com/highcharts/Series" target="_blank">Series</a>.
These can be used by wrapping them in '{}' eg. `{series.name} - {point.y}`. </p>
<tk-example> <chart:display response="response" title="Files Indexed"
subtitle="Per Day"> <series:facet name="test1" type="column" title="First Pass">
</series:facet> <series:facet name="test2" type="column" title="Second Pass">
</series:facet> <chart:tooltip point-format="{series.name}: <b>{point.y}</b>
<br/>"></chart:tooltip> </chart:display> </tk-example> <tk-code-wrapper> <code
hljs> <chart:display response="response" title="Files Indexed" subtitle="Per
Day"> <series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
<chart:tooltip point-format="{series.name}: <b>{point.y}</b><br/>">
</chart:tooltip> </chart:display> </code> </tk-code-wrapper> <hr/> <h3
id="example_shared-tooltip">Shared Tooltip</h3> <p>If you are rendering more then
one series you can set the tooltip to be shared, this means it will display
information about all the series for the point you've hovered over.</p> <tk-
example> <chart:display response="response" title="Files Indexed" subtitle="Per
Day"> <series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
<chart:tooltip point-format="{series.name}: <b>{point.y}</b><br/>"
shared="true"></chart:tooltip> </chart:display> </tk-example> <tk-code-wrapper>
<code hljs> <chart:display response="response" title="Files Indexed"
subtitle="Per Day"> <series:facet name="test1" type="column" title="First Pass">
</series:facet> <series:facet name="test2" type="column" title="Second Pass">
</series:facet> <chart:tooltip point-format="{series.name}: <b>{point.y}</b>
<br/>" shared="true"></chart:tooltip> </chart:display> </code> </tk-code-wrapper>
<hr/> <h3 id="example_percentages">Percentages</h3> <p>If you are rendering data
within a pie or donut chart you can user `{point.percentage}%` to show the
percentage of the segment you have selected in the chart.</p> <tk-example>
<chart:display response="response" title="Files Indexed"> <series:facet
name="test1" type="pie" title="First Pass"></series:facet> <chart:tooltip point-
format="{series.name}: <b>{point.percentage:.1f}%</b><br/>"></chart:tooltip>
</chart:display> </tk-example> <tk-code-wrapper> <code hljs> <chart:display
response="response" title="Files Indexed"> <series:facet name="test1" type="pie"
title="First Pass"></series:facet> <chart:tooltip point-format="{series.name}:
<b>{point.percentage:.1f}%</b><br/>"></chart:tooltip> </chart:display> </code>
</tk-code-wrapper> <hr/> <h3 id="example_adding-colour">Adding Colour</h3> <p>You
can add the colour of the series to your highcharts tooltip by adding the
following snippet:- <code>&lt;span class='highcharts-color-
{point.colorIndex}'&gt;●&lt;/span&gt;</code> To do this you will need to use
single quotes for your point-format eg. point-format='<span class="my-
class">Hello World</span>'. As the highcharts point format will not render
classes within double quotes. </p> <tk-example> <chart:display
response="response" title="Files Indexed"> <series:facet name="test1" type="pie"
title="First Pass"></series:facet> <chart:tooltip point-format='<span
class="highcharts-color-{point.colorIndex}">●</span> {series.name}: <b>
{point.percentage:.1f}%</b><br/>'></chart:tooltip> </chart:display> </tk-
example> <tk-code-wrapper> <code hljs hljs-no-escape> &lt;chart:display
response="response" title="Files Indexed"&gt; &lt;series:facet name="test1"
type="pie" title="First Pass"&gt;&lt;/series:facet&gt; &lt;chart:tooltip point-
format='&lt;span class="highcharts-color-
{point.colorIndex}&quot;&gt;●&lt;/span&gt; {series.name}: &lt;b&gt;
{point.percentage:.1f}%&lt;/b&gt;&lt;br&gt;&gt;'&gt;&lt;/chart:tooltip&gt;
&lt;/chart:display&gt; </code> </tk-code-wrapper> <hr/> <h3
id="example_different-tooltips">Different Tooltips</h3> <p>If for you need the
tooltips to be different for each series you can add the chart:tooltip tag inside
your series to apply the tooltip to just that series.</p> <tk-example>
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
```



```

<series:facet name="test1" type="column" title="First Pass"> <chart:tooltip
point-format='{series.name}: <b>{point.y} Files Indexed By Steve </b><br/>'>
</chart:tooltip> </series:facet> <series:facet name="test2" type="column"
title="Second Pass"> <chart:tooltip point-format='{series.name}: <b>{point.y}
Files Indexed By Mike </b><br/>'></chart:tooltip> </series:facet>
</chart:display> </tk-example> <tk-code-wrapper> <code hljs> <chart:display
response="response" title="Files Indexed" subtitle="Per Day"> <series:facet
name="test1" type="column" title="First Pass"> <chart:tooltip point-
format='{series.name}: <b>{point.y} Files Indexed By Steve </b><br/>'>
</chart:tooltip> </series:facet> <series:facet name="test2" type="column"
title="Second Pass"> <chart:tooltip point-format='{series.name}: <b>{point.y}
Files Indexed By Mike </b><br/>'></chart:tooltip> </series:facet>
</chart:display> </code> </tk-code-wrapper> </div>

```

```

angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { test1: { filters: [
{ val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val: { dsp: 'Day 2',
act: 'Day 2', }, count: 200, } ] }, test2: { filters: [ { val: { dsp: 'Day 1',
act: 'Day 1', }, count: 10, }, { val: { dsp: 'Day 2', act: 'Day 2', }, count: 20,
} ] }, }, results: [ { result: { fields: { label: 'label1', value: 1 } }, }, {
result: { fields: { label: 'label2', value: 2 } }, hits: { act: ['Hits', 264] }
} ] } ResponseService.setResponse('response',$scope.response); }]);

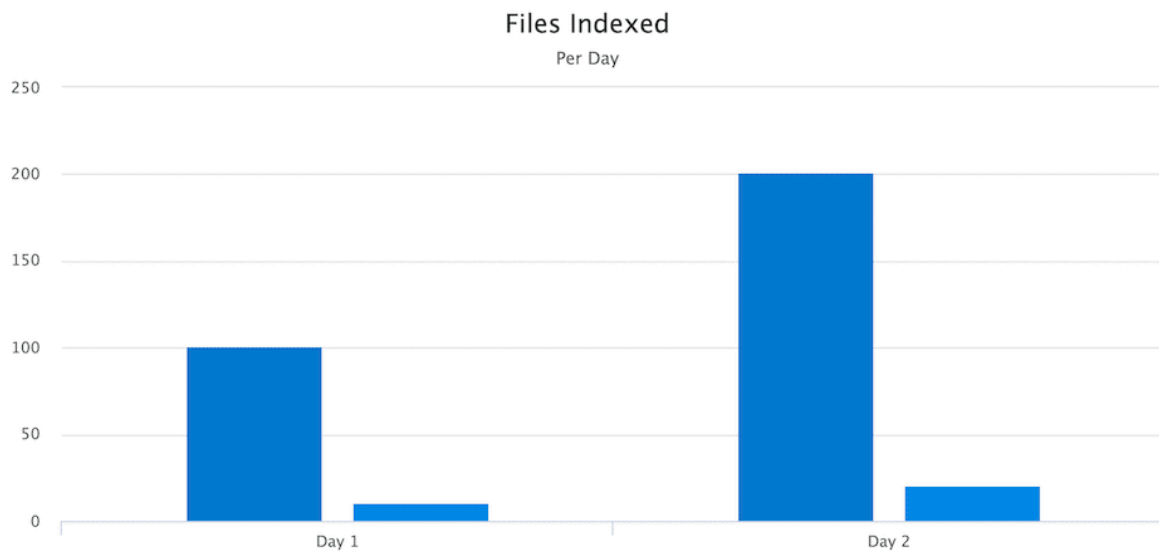
```

Result

Tooltip Tag

When setting a tooltip you have access to two objects called [Point](#) and [Series](#). These can be used by wrapping them in '{}' eg. `{series.name} - {point.y}`.

EXAMPLE

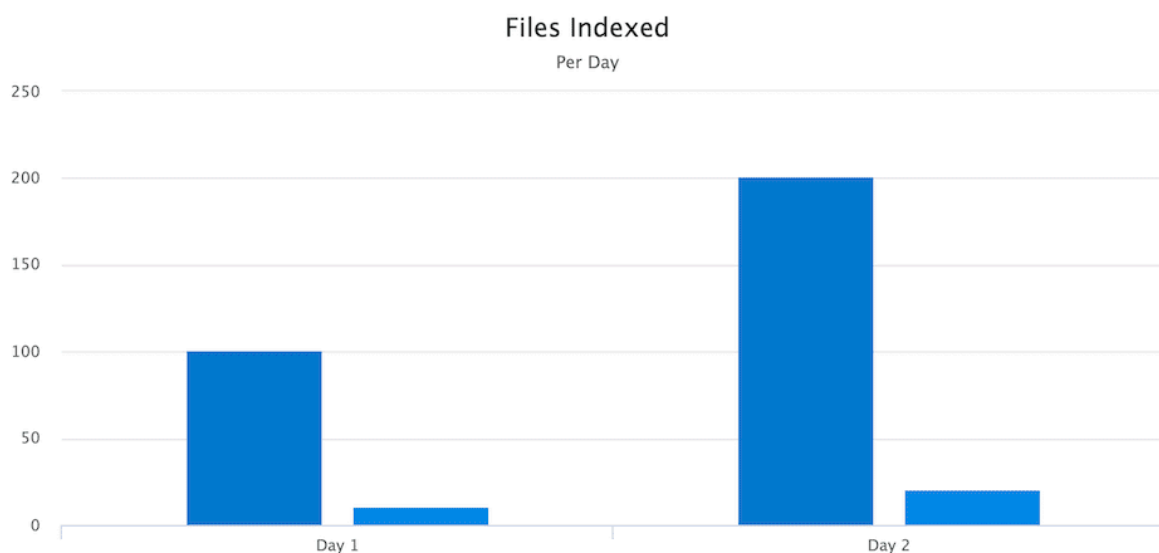


```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
  <series:facet name="test1" type="column" title="First Pass"></series:facet>
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>
  <chart:tooltip point-format="{series.name}: <b>{point.y}</b><br/>"></chart:tooltip>
</chart:display>
```

Shared Tooltip

If you are rendering more than one series you can set the tooltip to be shared, this means it will display information about all the series for the point you've hovered over.

EXAMPLE

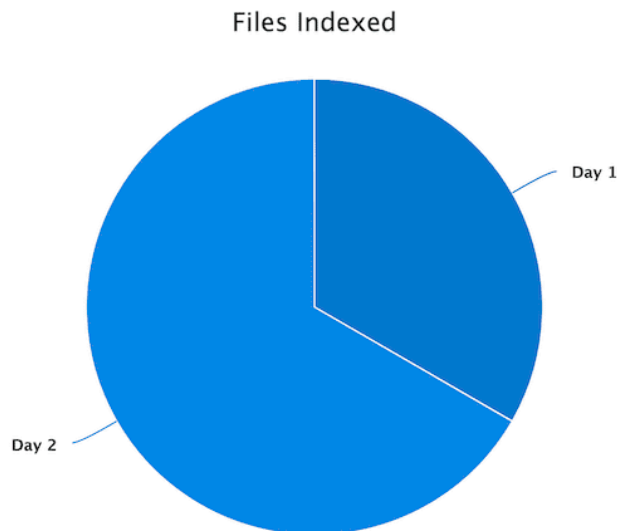


```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
  <series:facet name="test1" type="column" title="First Pass"></series:facet>
  <series:facet name="test2" type="column" title="Second Pass"></series:facet>
  <chart:tooltip point-format="{series.name}: <b>{point.y}</b><br/>" shared="true">
</chart:tooltip>
</chart:display>
```

Percentages

If you are rendering data within a pie or donut chart you can use `{point.percentage}%` to show the percentage of the segment you have selected in the chart.

EXAMPLE



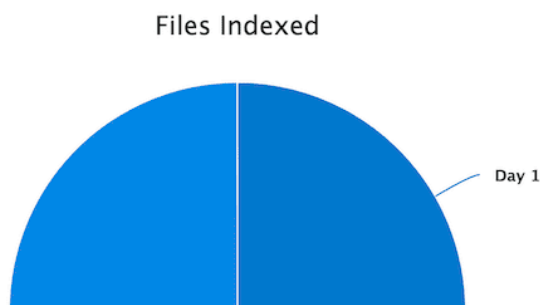
```
<chart:display response="response" title="Files Indexed">
  <series:facet name="test1" type="pie" title="First Pass"></series:facet>
  <chart:tooltip point-format="{series.name}: <b>{point.percentage:.1f}</b> %</b><br/>">
</chart:tooltip>
</chart:display>
```

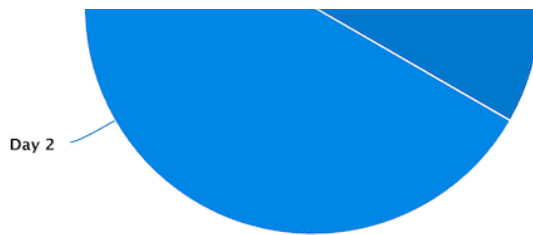
Adding Colour

You can add the colour of the series to your highcharts tooltip by adding the following snippet:-

`<span class='highcharts-color-{point.colorIndex}'>•</span>` To do this you will need to use single quotes for your point-format eg. point-format='Hello World'. As the highcharts point format will not render classes within double quotes.

EXAMPLE



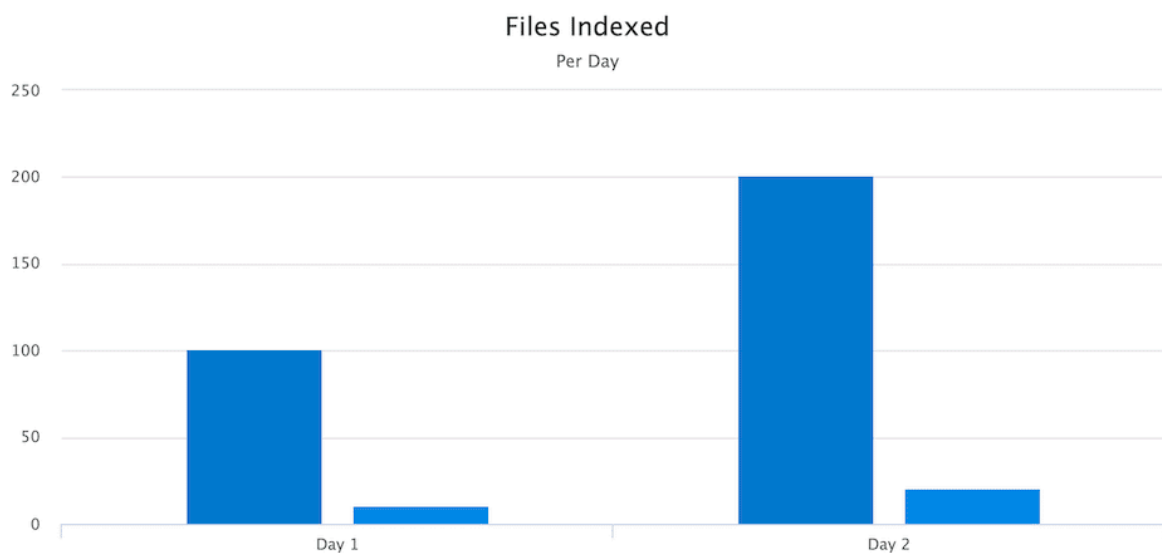


```
<chart:display response="response" title="Files Indexed">
  <series:facet name="test1" type="pie" title="First Pass"></series:facet>
  <chart:tooltip point-format='<span class="highcharts-color-{point.colorIndex}">●</span>
{series.name}: <b>{point.percentage:.1f} %</b><br/>'></chart:tooltip>
</chart:display>
```

Different Tooltips

If for you need the tooltips to be different for each series you can add the chart:tooltip tag inside your series to apply the tooltip to just that series.

EXAMPLE



```
<chart:display response="response" title="Files Indexed" subtitle="Per Day">
  <series:facet name="test1" type="column" title="First Pass">
    <chart:tooltip point-format="{series.name}: <b>{point.y} Files Indexed By Steve </b>
<br/>"></chart:tooltip>
  </series:facet>
  <series:facet name="test2" type="column" title="Second Pass">
    <chart:tooltip point-format="{series.name}: <b>{point.y} Files Indexed By Mike </b>
<br/>"></chart:tooltip>
  </series:facet>
</chart:display>
```

class:toggle

Description

Applies a class to the target when toggled on.

Usage

as element:

```
<class:toggle target="{string}" toggle-class="{string}" [name="{string}"]>
</class:toggle>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
target	string	jQuery selector for element
toggle-class	string	Name of class to toggle
name (optional)	string	Name of toggle. If set the user's preference for the toggle will be saved

Example

```
<search:toggle target="#chart" class="hide" name="chart">Toggle</search:toggle>
```

click:outside

Description

Calls function when a click event occurs outside of the element. Good for closing side panels, menus, etc.

Usage

as attribute

```
<ANY click-outside="{expression}" [outside-if-not="{string}"]> ... </ANY>
```

Parameters

Param	Type	Details
click-outside	expression	Function to call when element is clicked outside of.
outside-if-not (optional)	string	Selector to match any elements that should not trigger the click-outside function

collaborate:bookmark

Description

Creates a new bookmark.

- If used within a result you will need to add the following workflow processor to your platform in order to get the status of the bookmark and the number of people who have bookmarked the result

```
workflow:
  twigkit.social.processor.AddBookmarkFactsProcessor
```

Usage

as element:

```
<collaborate:bookmark target="{string}" title="{string}" [topic="{number}"]
[collection="{string}"] [url="{string}"] [enable-tooltip="{boolean}"]>
</collaborate:bookmark>
```

Parameters

Param	Type	Details
target	string	A unique identifier for a document that is being bookmarked. Example values: document123 Maximum length: 2083 characters
title	string	A title for the new bookmark. Example values: Interesting document Maximum length: 4000 characters
topic (optional)	number	A topic to which the new bookmark should belong.
collection (optional)	string	The collection to which the target document belongs.
url (optional)	string	A URL which the new bookmark should link to. Default: the bookmark target, if it is a URL
enable-tooltip (optional)	boolean	Whether to enable the tooltip (Default:true).

Example

Source

```
<h3 id="example_custom-html">Custom HTML</h3> <collaborate:bookmark><a  
styling="button">Bookmark</a></collaborate:bookmark> <hr/> <h3  
id="example_traditional">Traditional</h3> <collaborate:bookmark>  
</collaborate:bookmark> <hr/> <h3 id="example_without-tooltip">Without  
Tooltip</h3> <collaborate:bookmark enable-tooltip="false"></collaborate:bookmark>
```

Result

Custom HTML

Bookmark

Traditional



Without Tooltip



collaborate:bookmark-list

Description

The bookmark list directive to render a list of bookmarks, you can put your own html inside the bookmark list to customise it, in order to do this you will need to do the following :-

```
<collaborate:bookmark-list>
  <li ng-repeat="bookmark in $parent.bookmarks">YOUR CUSTOM CODE HERE</li>
</collaborate:bookmark-list>
```

Usage

as element:

```
<collaborate:bookmark-list [title="{string}"] [delete-label="{string}"] [target="{string}"] [topic="{number}"] [collection="{string}"] [date-format="{string}"] [url-target="{string}"]> </collaborate:bookmark-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
title (optional)	string	If set, title will be used displayed above bookmarks
delete-label (optional)	string	Label to display for the delete action. Default: "x"
target (optional)	string	If set, only display bookmarks related to this target document.
topic (optional)	number	Optionally, bookmarks can be scoped to a particular topic. If this attribute is set, then the tag will list only those bookmarks that are linked to that topic. Passing topic 0 will check that a topic has not be associated with the bookmark
collection (optional)	string	Optionally, bookmarks can be scoped to a particular collection, identified by a string ID. If this attribute is set, then the tag will list only those bookmarks that belong to the given collection.
date-format (optional)	string	Format to use for the bookmark created date. Default: <code>dd/MM/yyyy h:mm</code> . See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.
url-target (optional)	string	Set an HTML target attribute for the anchor tag. For example use <code>_blank</code> to have the bookmark link open in a new window.

collaborate:like

Description

Creates a like button.

If used within a result you will need to add the following workflow processor to your platform configuration file in order to get the status of the like and the number of people who have liked the result. The default file is `platforms/fusion/data.conf`.

```
workflow:
  twigkit.social.processor.AddLikeFactsProcessor
```

Usage

as element:

```
<collaborate:like target="{string}" [topic="{number}"] platform="{string}"
[collection="{string}"] [enable-tooltip="{boolean}"]> </collaborate:like>
```

Parameters

Param	Type	Details
target	string	A unique identifier for a document that is being liked. Example values: document123 Maximum length: 2083 characters
topic (optional)	number	A topic to which the new like should belong.
platform	string	Name of the platform to send the data to.
collection (optional)	string	The collection to which the target document belongs.
enable-tooltip (optional)	boolean	Whether to enable the tooltip (Default:true).

Example

Source

```
<h3 id="example_traditional">Traditional</h3> <collaborate:like>  
</collaborate:like> <hr/> <h3 id="example_without-tooltip">Without Tooltip</h3>  
<collaborate:like enable-tooltip="false"></collaborate:like>
```

Result

Traditional



Without Tooltip



collaborate:saved-query-list

Description

The saved query list directive to render a list of saved queries

Usage

as element:

```
<collaborate:saved-query-list user="{object}" [title="{string}"] [allow-saving="{string}"] [date-format="{string}"]> </collaborate:saved-query-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
user	object	Display saved queries by the given user (type: <code>twigkit.social.model.Profile</code>).
title (optional)	string	If set, title will be used displayed above bookmarks
allow-saving (optional)	string	If set to false the save search input will not be shown
date-format (optional)	string	Format to use for the saved query created date. Default: <code>dd/MM/yyyy h:mm</code> . See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.

Events

savedQueryAdded

When a query is saved this event broadcast with the name of the query being saved, the URL being used and the topic if used.

```
angular.module('twigkitLightApp').controller('MainCtrl', ['$scope', function ($scope) { $scope.$on('savedQueryAdded',function(query){ alert('Query - ' + query.name + ' have been saved.');
```

Type:

broadcast

Target:

root scope

savedQueryDeleted

When a query is deleted this event is broadcast so that any custom alerts or functions can be run afterwards like the example below.

```
angular.module('twigkitLightApp') .controller('MainCtrl', ['$scope', function ($scope) { $scope.$on('savedQueryDeleted',function(){ alert('Query has been deleted.');
```

Type:

broadcast

Target:

root scope

collaborate:topic-form

Description

Generates a form you can use to create a topic.

Usage

as element:

```
<collaborate:topic-form [name="{string}"] [button-label="{string}"] [namespace="{string}"] [placeholder="{string}"] [show-description="{boolean}"] [description-placeholder="{string}"]> </collaborate:topic-form>
```

Directive info

This directive creates new scope.

Parameters

Param	Type	Details
name (optional)	string	Default title for a topic.
button-label (optional)	string	Label of form submit button. (default: Save)
namespace (optional)	string	Determines whether the topics created using the form will be "local" (restricted to the user) or "global". (default: local)
placeholder (optional)	string	Placeholder value to put in the topic name text box. (default: Type a name for the new topic...)
show-description (optional)	string	Whether to include a text box for adding a description for the topic.
description-placeholder (optional)	string	Placeholder value to put in the description text box (default: Description)

collaborate:topic-list

Description

The topic list directive is used to render a list of topics specific to your organization. To customize the topic list, enter your custom html code with the following commands:

```
<collaborate:topic-list> <li ng-repeat="topic in $parent.topics">YOUR CUSTOM CODE  
HERE</li> </collaborate:topic-list>
```

Usage

as element:

```
<collaborate:topic-list [title="{string}"] [delete-label="{string}"] [show="{  
{number}"]> </collaborate:topic-list>
```

Directive information

This directive creates new scope.

content:boolean

Description

Creates a checkbox boolean

Usage

as element:

```
<content:boolean label="{string}" field="{string}" [true-value="{string}"]  
[false-value="{string}"]> </content:boolean>
```

Parameters

Param	Type	Details
label	string	The label for the textarea
field	string	The name of the field.
true-value (optional)	string	The value to use when true, (default: true)
false-value (optional)	string	The value use when false, (default: false)

Example

Source

```
<h3 id="example_stacked">Stacked</h3> <content:form styling="form-stacked">  
<content:boolean label="Test" field="test"></content:boolean> </content:form>
```

Result

Stacked

☐ Test

content:form

Description

Creates a form, all inputs into the form are attached to a variable called model.

Adding a platform to the form will send the request to the Platform Store API (<http://dev.twigkit.net/docs/modules/Webservices/Store/>) and create the model as a new result.

Usage

as element:

```
<content:form [platform="{string}"] [legend="{string}"] [action="{string}"]  
[content="{Object}"]> </content:form>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
platform (optional)	string	The platform to post the data to.
legend (optional)	string	Text to display in the legend of the form.
action (optional)	string	Whether the form should be creating a new record (store) or updating an existing record (update). Default: `store`, options store or update
content (optional)	Object	Key value pair object or result to map the form data to.

Events

contentFormPostError

When the content form is posted to the API unsuccessfully this event will be broadcasted with the error received from the API.

Type:

broadcast

Target:

root scope

contentFormPostSuccess

When the content form is posted to the API successfully this event will be broadcasted with the model sent to the API.

Type:

broadcast

Target:

root scope

Example

Source

```
<div ng-controller="ExampleController"> <search:platform var="platform"
conf="platform.test"></search:platform> <layout:grid> <layout:block md="1-1"> <h3
id="example_styling">Styling</h3> </layout:block> <layout:block md="1-2"> <h4
id="example_styling">(form-aligned)</h4> <content:form styling="form-aligned"
platform="platform"> <content:input label="id" field="id" value="123456"
hidden="true"></content:input> <content:input label="Name" field="name"
required="true"></content:input> <content:textarea label="Address"
field="Address" required="true"></content:textarea> <content:submit>
</content:submit> </content:form> </layout:block> <layout:block md="1-2"> <h4
id="example_styling">(form-stacked)</h4> <content:form styling="form-stacked"
content="model1" platform="platform"> <content:input label="id" field="id"
value="12345" hidden="true"></content:input> <content:input label="Name"
field="name" required="true"></content:input> <content:textarea label="Address"
field="Address"></content:textarea> <content:submit></content:submit>
</content:form> </layout:block> </layout:grid> <hr/> <h3
id="example_attributes">Attributes</h3> <h4
id="example_attributes_legend">Legend</h4> <content:form legend="Please complete
the form." content="result"> <content:input label="id" field="id" value="1234"
hidden="true"></content:input> <content:input label="Name" field="name"
required="true"></content:input> <content:textarea label="Address"
field="Address"></content:textarea> <content:submit></content:submit>
</content:form> <hr/> <h4 id="example_attributes_content">Content</h4> <button
styling="button" ng-click="changeModel()">Update Content Model</button>
<layout:grid> <layout:block md="1-1"> <p>Click update content model button to
update the values in the inputs.</p> <content:form styling="form-stacked"
content="model1" platform="platform"> <strong>Form Model</strong> <pre>
{{$parent.model}}</pre> <content:input label="id" field="id" value="12345"
hidden="true"></content:input> <content:input label="Name" field="name"
required="true"></content:input> <content:textarea label="Address"
field="Address"></content:textarea> <content:submit></content:submit>
</content:form> </layout:block> <layout:block md="1-1"> <p>To only initiated the
form with the values from a result or scope object use
<code>content="::myModelObject"</code>. If you click update content model the
values of this form will not change. (see <a
href="https://docs.angularjs.org/guide/expression#one-time-binding"
target="_blank">one time bindings</a>)</p> <content:form styling="form-stacked"
content="::model1" platform="platform"> <strong>Form Model</strong> <pre>
{{$parent.model}}</pre> <content:input label="id" field="id" value="12345"
hidden="true"></content:input> <content:input label="Name" field="name"
required="true"></content:input> <content:textarea label="Address"
field="Address"></content:textarea> <content:submit></content:submit>
</content:form> </layout:block> </layout:grid> </div>
```

```
angular.module('lightning').controller('ExampleController', ['$scope',
function($scope) { $scope.model1 = { name:'foo', Address:'bar' } function
generateRandomString(size){ var text = ""; var possible =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"; for( var i=0; i
< size; i++ ) text += possible.charAt(Math.floor(Math.random() *
possible.length)); return text; } $scope.changeModel = function(){ $scope.model1
= { name:generateRandomString(4), Address:generateRandomString(3) } };
$scope.result ={ result: { fields: { name:{ val:'George', } , Address:{ val:'123
George Way', }, } }, } }]);
```

Result

Styling

(form-aligned)

Name *

Address *

(form-stacked)

Name *

Address

Attributes

Legend

Please complete the form.

Name *

Address

Content

Click update content model button to update the values in the inputs.

Form Model

```
{"id":"12345","name":"foo","Address":"bar"}
```

Name *

Address

To only initiated the form with the values from a result or scope object use `content=":myModelObject"`. If you click update content model the values of this form will not change. (see [one time bindings](#))

Form Model

```
{"id":"12345","name":"foo","Address":"bar"}
```

Name *

foo

Address

bar

Submit

content:input

Description

Creates a textbox

Usage

as element:

```
<content:input label="{string}" field="{string}" [required] [value="{string}"]  
[hidden="{boolean}"] [placeholder="{string}"]> </content:input>
```

Parameters

Param	Type	Details
label	string	The label for the textbox
field	string	The name of the field.
required (optional)	boolean	Whether the field is required not, if it is a warning will be displayed until the field is value. Default:false.
value (optional)	string	Value to display in the textbox.
hidden (optional)	boolean	Whether the textbox should be displayed.
placeholder (optional)	string	The placeholder for the textbox, if no placeholder is defined the label will be used as the placeholder.

Example

Source

```
<h3 id="example_stacked">Stacked</h3> <content:form styling="form-stacked">
<content:input label="Test" field="test"></content:input> <content:input
label="Foo" field="bar"></content:input> <content:input label="" field="barfoo"
placeholder="No Label"></content:input> </content:form> <hr/> <h3
id="example_aligned">Aligned</h3> <content:form styling="form-aligned">
<content:input label="Test" field="test"></content:input> <content:input
label="Foo" field="bar"></content:input> <content:input label="" field="barfoo"
placeholder="No Label"></content:input> </content:form>
```

Result

Stacked

Test

Foo

Aligned

Test

Foo

content:radio

Description

Creates a radio button

Usage

as element:

```
<content:radio label="{string}" field="{string}" value="{string}">
</content:radio>
```

Parameters

Param	Type	Details
label	string	The label for the textarea
field	string	The name of the field.
value	string	The value of the field when selected

Example

Source

```
<h3 id="example_stacked">Stacked</h3> <content:form styling="form-stacked">
<content:radio label="Test" field="test" value="tested"></content:radio>
<content:radio label="Do not test" field="test" value="untested"></content:radio>
</content:form>
```

Result

Stacked

☐ Test

☐ Do not test

content:submit

Description

Creates a submit button

Usage

as element:

```
<content:submit> </content:submit>
```

content:textarea

Description

Creates a textbox

Usage

as element:

```
<content:textarea label="{string}" field="{string}" [required] [value="{string}"]  
[placeholder="{string}"]> </content:textarea>
```

Parameters

Param	Type	Details
label	string	The label for the textarea
field	string	The name of the field.
required (optional)	boolean	Whether the field is required not, if it is a warning will be displayed until the field is value. Default:false.
value (optional)	string	Value to display in the textarea.
placeholder (optional)	string	The placeholder for the textarea, if no placeholder is defined the label will be used as the placeholder.

Example

Source

```
<h3 id="example_stacked">Stacked</h3> <content:form styling="form-stacked">
<content:input label="Test" field="test"></content:input> <content:input
label="Foo" field="bar"></content:input> <content:textarea label="Bar"
field="foo"></content:textarea> <content:textarea label="" field="foobar"
placeholder="No Label"></content:textarea> </content:form> <hr/> <h3
id="example_aligned">Aligned</h3> <content:form styling="form-aligned">
<content:input label="Test" field="test"></content:input> <content:input
label="Foo" field="bar"></content:input> <content:textarea label="Bar"
field="foo"></content:textarea> <content:textarea label="" field="foobar"
placeholder="No Label"></content:textarea> </content:form>
```

Result

Stacked

Test

Foo

Bar

Aligned

Test

Foo

Bar

controller

Description

Wraps the html in a controller.

Usage

as attribute

```
<ANY controller="{string}"> ... </ANY>
```

Parameters

Param	Type	Details
controller	string	The name of the controller.

Example

Source

```
<div controller="ExampleController"> <p>{{foo}} <a href="javascript:void(0);" ng-  
click="alert()" styling="button">Alert!</a> </p> </div> <hr/> <div  
controller="Example2Controller"> <p>{{bar}}</p> </div>
```

```
angular.module('lightning') .controller('ExampleController',  
['$scope','$timeout', function($scope,$timeout) { $scope.foo = 'bar';  
$scope.alert = function(){ alert('Hello World!') }; }])  
.controller('Example2Controller', ['$scope','$timeout', function($scope,$timeout)  
{ $scope.bar = 'foo'; }]);
```

Result

bar Alert!

foo

default:src

Description

When an `img` tag gets a 404 the `src` attribute will be replaced with the `default-src` attribute

Usage

as attribute

```
<ANY default-src> ... </ANY>
```

dev:track-digests

Description

Prints out the number of digests that have taken place. <https://toddmotto.com/super-fast-angular-ng-model-options-limit-digest-cycles/> (<https://toddmotto.com/super-fast-angular-ng-model-options-limit-digest-cycles/>)

Usage

as element:

```
<dev:track-digests> </dev:track-digests>
```

Example

```
<dev:track-digests></dev:track-digests>
```


drawer

Description

Adds a class for the drawer

Usage

as attribute

```
<ANY drawer="{number}"> ... </ANY>
```

Parameters

Param	Type	Details
drawer	string	The direction the drawer should come from (left or right);

Example

Source

```
<layout:grid> <layout:block tiny="1-4"> <layout:box> <search:offcanvas-toggle  
target="#left-sidebar"></search:offcanvas-toggle> </layout:box> </layout:block>  
<layout:block tiny="1-2"> </layout:block> <layout:block tiny="1-4"> <layout:box>  
<search:offcanvas-toggle target="#right-sidebar" style="float:right;">  
</search:offcanvas-toggle> </layout:box> </layout:block> </layout:grid>  
<layout:grid style="row-offcanvas"> <layout:block medium="1-4" drawer="left"  
id="left-sidebar" > <layout:box class="tk-stl-bg-red" > <layout:sidebar> <h4  
id="example_left-sidebar-content">Left Sidebar Content</h4> </layout:sidebar>  
</layout:box> </layout:block> <layout:block medium="1-2"> <layout:box> <p  
style="text-align:center;">Shrink your browser window to toggle sidebars</p>  
</layout:box> </layout:block> <layout:block medium="1-4" drawer="right"  
id="right-sidebar" > <layout:box class="tk-stl-bg-blue" > <layout:sidebar> <h4  
id="example_right-sidebar-content">Right Sidebar Content</h4> </layout:sidebar>  
</layout:box> </layout:block> </layout:grid>
```

Result

Left Sidebar Content

Shrink your browser window to toggle sidebars

Right Sidebar Content

dynamic:styling

Description

Works the same as the styling directive but watches for updates. This should only be used if it is required that the styling classes are updated after their initial value to improve performance.

Usage

as attribute

```
<ANY dynamic-styling="{string}"> ... </ANY>
```

Parameters

Param	Type	Details
dynamicStyling	string	separated string of styles to apply to the element

Example

Source

```
<layout:box dynamic-styling="{{styles}}"> <button styling="button" ng-  
click="toggleStyles()">Toggle box styling</button> </layout:box>
```

```
angular.module('lightning') .controller('ExampleController', ['$scope',  
function($scope) { $scope.styles = 'bg-raspberry text-raspberry-dark';  
$scope.toggleStyles = function () { if ($scope.styles == 'bg-raspberry text-  
raspberry-dark') { $scope.styles = 'bg-sunflower text-sunflower-dark'; } else {  
$scope.styles = 'bg-raspberry text-raspberry-dark'; } }; });
```

Result

Toggle box styling

esri:heatmap

Description

The heatmap directive

Usage

as element:

```
<esri:heatmap response="{string}" facet-name="{string}" [height="{string}"]  
[width="{string}"] [basemap="{string}"]> </esri:heatmap>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	Name of the response.
facet-name	string	Facet containing the data.
height (optional)	string	The height of the map container. Default is 400px.
width (optional)	string	The width of the map container. Default is 100%.
basemap (optional)	string	The map to use to display the points on. Default: topo – Esri options (https://developers.arcgis.com/javascript/4.2/api-reference/esri-Map.html#basemap)

Example

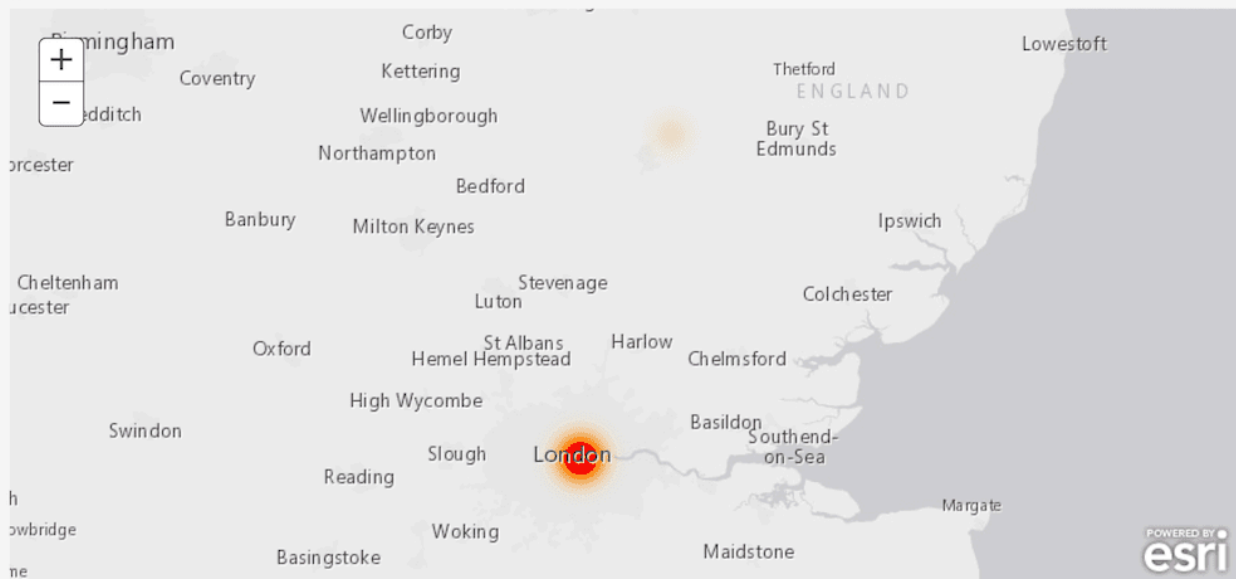
Source

```
<h3 id="example_geo_loc_coordraw-facet">geo_loc_coord.raw Facet</h3>
<esri:heatmap response="response" facet-name="geo_loc_coord.raw" basemap="gray">
</esri:heatmap>
```

```
angular.module('lightning').controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { $scope.response = { page: 2,
query: { rpp: 50 }, facets: { heat: { filters: [ { val: '37.770090, -122.408904',
count: 1, }, { val: '37.782745, -122.444586', count: 1, }, { val: '37.782842,
-122.443688', count: 1, }, { val: '37.782919, -122.442815', count: 1, }, { val:
'37.782992, -122.442112', count: 1, }, { val: '37.783100, -122.441461', count: 1,
}, { val: '37.783206, -122.440829', count: 1, }, { val: '37.783273, -122.440324',
count: 1, }, { val: '37.783316, -122.440023', count: 1, }, { val: '37.783357,
-122.439794', count: 1, }, { val: '37.783371, -122.439687', count: 1, }, { val:
'37.783368, -122.439666', count: 1, }, ], "geo_loc_coord.raw": { "id":
"geo_loc_coord.raw", "field": "geo_loc_coord.raw", "filters": [{ "field":
"geo_loc_coord.raw", "count": 39, "facts": {}, "val": "51.4964, -0.1224", "rel":
[] }, { "field": "geo_loc_coord.raw", "count": 3, "facts": {}, "val": "52.2599,
0.2288", "rel": [] } ], "params": {}, "display": "geo_loc_coord.raw", "max-
count": 39.0, "min-count": 0.0, "sum-of-counts": 42.0 }, }, results: [] } }]);
```

Result

geo_loc_coord.raw Facet



esri:webmap

Description

Adds webmap layers from an esri webmap. The webmap id must be applied to the geo-map element.

Usage

as element:

```
<esri:webmap [url="{string}"] [toggle-layers="{boolean}"]> </esri:webmap>
```

Parameters

Param	Type	Details
url (optional)	string	The URL to the arcgis portal, if not supplied the default will be used by esri.
toggle-layers (optional)	boolean	Whether layers can be turned on and off.

Example

Source

```
<h3 id="example_esri-map">Esri Map</h3> <geo:map response="response2"
provider="esri" address-field="address" label-field="name" webmap-
id="ef9c7fbda731474d98647bebb4b33c20" zoom-level="13"> <esri:webmap toggle-
layers="true"></esri:webmap> </geo:map>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { $scope.response2 = { page: 2,
query: { rpp: 50 }, facets: {}, results: [ { result: { fields: { name:{
val:'Twigkit CA', }, address: { val:'1551 McCarthy Boulevard, Suite 209,
Milpitas, CA 95035', }, lat:{ val:'37.442172' }, long:{ val:'-121.9151737' } } },
]], } }]);
```

Result



facet:dropdown

Description

Display a facet as a select dropdown, allowing the user to apply one or more filters by selection options from the dropdown. To specify the facet that should be rendered, either pass in a facet object directly or specify a facet-name and a response from where to retrieve that named facet.

Usage

as element:

```
<facet:dropdown facet="{string}" response="{string}" facet-name="{string}"
[select="{string}"] display-label="{string}" [search-enabled="{boolean}"]>
</facet:dropdown>
```

Parameters

Param	Type	Details
facet	string	A facet object to display
response	string	The name of the response to use.
facet-name	string	Facet name to use from the response object
select (optional)	string	How to render each facet filter + Valid values:- multiselect Supports selecting multiple filters that get combined with an 'OR'. * toggle Only allows one filter to be selected at a time but shows unselected filters and counts * multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.
display-label	string	Initial display value for dropdown
search-enabled (optional)	boolean	Whether to enable search on the facet. If no query or platform is defined on the facet or facet-list it will enable the user to search through the filters that have been returned in the response. If a platform and query is defined on the facet or facet-list then it will search for filters in the search index. (Requires a tokenized field, if not using a tokenized field, if not using a tokenized field please see facet-field).

facet:hierarchical

Description

Hierarchical Facets for Solr.

In order to correctly display a taxonomy as a hierarchical facet you need to configure the requisite Twigkit service to retrieve the data structure [configured in your Solr index][1].

First you need to create a service configuration:

```
/conf
  /services
    /hierarchical
      /hierarchical.conf
```

The file 'hierarchical.conf' should be configured similar to the following example:

```
name: twigkit.search.solr.service.facet.GreedySolrHierarchicalFacetService
platform: platforms.solr
parents-suffix: _parent
```

platform This specifies the configured platform where the taxonomy has been indexed.

parents-suffix The suffix, that if appended to the facet's fieldname will designate the field containing parent node data. Will depend on how the taxonomy data has been indexed.

Example:

```
<search:platform var="platformHierarchical" conf="services.hierarchical">
</search:platform>
<search:facet facet-name="location" show="6" show-more="6">
  <facet:hierarchical facet-name="location" platform="platformHierarchical"
query="query"></facet:hierarchical>
</search:facet>
```

Usage

as element:

```
<facet:hierarchical query="{string}" [facet-name="{string}"] [platform="{string}"] [count-number="{string}"] [select="{string}"] [max-characters="{Number}"] [expand-root="{boolean}"]> </facet:hierarchical>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
query	string	The name of the query object.
facet-name (optional)	string	Facet name to use from the response object
platform (optional)	string	The name of the platform to use.
count-number (optional)	string	Specify how the number representing the count should be formatted. Optional values are 'plain', 'formatted' and 'rounded'. Rounded means the number is represented in a short format (2k for ~2000). Formatted means that the number is comma formatted (2,000,000 for 2000000).Default: 'formatted'
select (optional)	string	How to render each FacetFilter. 'click': User selects a filter by clicking the value; and 'mutliselect': Supports selecting multiple filters that get combined with an 'OR'. Default: 'click'
max-characters (optional)	Number	Limit display value to a certain number of characters, adding '...' if maxCharacters is exceeded.
expand-root (optional)	boolean	If only one root filter exists, if this is set to true it will expand this filter showing the children. (Default:false).

Example

Source

```
<search:platform var="platform" conf="platforms.workflow.data.collection.google-  
directory"></search:platform> <search:query var="query" parameters='q' results-  
per-page="1"></search:query> <search:facet-list response="response"  
styling="facet-list facet-list-step facet-list-wrappedheader facet-list-  
wrappedheader-noborder"> <search:facet facet-name="test1" title="First Pass  
Files" collapsible="true"> <facet:hierarchical facet-name="test1"  
platform="platform" query="query"></facet:hierarchical> </search:facet>  
</search:facet-list>
```

```
angular.module('lightning').controller('ExampleController',  
 ['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)  
 { $scope.response = { page: 2, query: { rpp: 50 }, facets: { test: { filters: [ {  
 val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val: { dsp: 'Day 2', act:  
 'Day 2', }, count: 200, } ] }, }, results: [] }  
 ResponseService.setResponse('response', $scope.response); }]);
```

geo:heatmap

Description

The heatmap directive

Usage

as element:

```
<geo:heatmap response="{string}" facet-name="{string}" [height="{string}"]  
[width="{string}"] [scroll-to-zoom="{boolean}"] [options="{object}"]>  
</geo:heatmap>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	Name of the response.
facet-name	string	Facet containing the data.
height (optional)	string	The height of the map container. Default is 400px.
width (optional)	string	The width of the map container. Default is 100%.
scroll-to-zoom (optional)	boolean	Whether the scroll wheel should control the map zoom. Disabled by default.
options (optional)	object	Extend the current options for the Google Map (Options Here) (https://developers.google.com/maps/documentation/javascript/reference#MapOptions)

geo:highmaps

Description

Highmaps directive for mapping data onto a map.

In order to use this directive you need to include the map you wish to use, they can be found here (<http://code.highcharts.com/mapdata/>)

For example if you want to use this map of the UK (<http://jsfiddle.net/gh/get/jquery/1.11.0/highslide-software/highcharts.com/tree/master/samples/mapdata/countries/gb/gb-all>) you need to include the following script tag `<script src="https://code.highcharts.com/mapdata/countries/gb/gb-all.js"></script>` and put `countries/gb/gb-all` on the chart `highmaps` map attribute. eg. `<chart:highmaps response="response" facet-name="facet" map="countries/gb/gb-all"></chart:highmaps>`

Usage

as element:

```
<geo:highmaps response="{string}" facet-name="{string}" map="{string}" [key="{string}"] [title="{string}"] [subtitle="{string}"] [series-name="{string}"] [query="{string}"] [options="{object}"] [select="{string}"]> </geo:highmaps>
```

Parameters

Param	Type	Details
response	string	Name of the response object to get the facet from.
facet-name	string	Name of the facet to retrieve the data from.
map	string	The map to use to render the data on.
key (optional)	string	The property for joining the data. (Default: 'hc-key') More details here (http://www.highcharts.com/docs/maps/map-collection)
title (optional)	string	Title for the chart.
subtitle (optional)	string	Subtitle for the chart.
series-name (optional)	string	Name of the series. (Default: Name from the facet)
query (optional)	string	Name of the query to apply filters to.
options (optional)	object	Extend the current options. Options can be found here. (http://api.highcharts.com/highmaps/)

Param	Type	Details
select (optional)	string	How the data should be filtered when clicking on a part of the chart, a valid query must be present. + Valid values:- • click User selects a filter by clicking the value; (default) * multiselect Supports selecting multiple filters that get combined with an 'OR'. * multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.

geo:layer

Description

The `geo:layer` directive allows you to add multiple results from multiple responses to your *Geo Map*.

Usage

as element:

```
<geo:layer response="{string}" [address-field="{string}"] [latitude-field="{string}"] [longitude-field="{string}"] [target="{string}"] [url-field="{string}"] [url-prefix="{string}"] [label-field="{string}"] [icon="{string}"] [icon-field="{string}"]> </geo:layer>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	Name of the response.
address-field (optional)	string	The field containing a full address to geocode to obtain latitude and longitude values. Use this or latitudeField/longitudeField, not both.
latitude-field (optional)	string	The field containing the latitude.
longitude-field (optional)	string	The field containing the longitude.
target (optional)	string	Specify the link target to use for the map pins. Can be one of '_blank', '_parent', '_self', '_top' or the name of a window. Default value is '_blank'."
url-field (optional)	string	The field containing the URL to make clickable from the map marker.
url-prefix (optional)	string	Use this if the URL retrieved from the URLField needs to be prefixed with a value.
label-field (optional)	string	The field containing the label to use as tooltip for the marker.
icon (optional)	string	The location of the marker image to use for this set of coordinates.
icon-field (optional)	string	The field from the response to get the icon from.

geo:map

Description

The geoMap directive provides an abstraction layer to other map directives.

Google Maps

API Key

To add the API for Google Maps you need to include a constant on your index page like so with your API Key. `<helper:constant name="googleMapApi" value="YOUR_API_KEY_HERE">`
`</helper:constant>` Doing this means you do not have to add the Google Map scripts to your index page

Esri Maps

For ESRI you will need to include the CSS and JavaScript from ESRI on your index page.

```
<script type="text/javascript" src="file:///Users/dustingericke/repositories/fusion-docs-site-legacy/public/js.arcgis.com/3.17/"></script>
<link rel="stylesheet" href="//js.arcgis.com/3.17/esri/css/esri.css">
```

Usage

as element:

```
<geo:map provider="{string}" [response="{string}"] [address-field="{string}"]
[latitude-field="{string}"] [longitude-field="{string}"] [api-key="{string}"]
[zoom-level="{string}"] [height="{string}"] [width="{string}"] [scroll-to-zoom="{
boolean}"] [target="{string}"] [url-field="{string}"] [url-prefix="{string}"]
[label-field="{string}"] [loading-overlay="{boolean}"] [icon="{string}"] [icon-
field="{string}"] [weight-field="{string}"] [geocode-server-url="{string}"]
[javascript-api-url="{string}"] [basemap="{string}"] [webmap-id="{string}"]
[options="{object}"] [events="{string}"]> </geo:map>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
provider	string	Map provider to use (Supported values: google, esri)
response (optional)	string	Name of the response
address-field (optional)	string	The field containing a full address to geocode to obtain latitude and longitude values. Use this or latitudeField/longitudeField, not both.
latitude-field (optional)	string	The field containing the latitude.
longitude-field (optional)	string	The field containing the longitude.
apiKey (optional)	string	An API key to use with the map platform (if applicable)
zoom-level (optional)	string	The default zoom level to use for the map. Default is 13 for single results, with a group of results the map will scale automatically to fit the pins into the bounds, zoom level will be applied if the pins still fit within the new bounds.
height (optional)	string	The height of the map container. Default is 400px.
width (optional)	string	The width of the map container. Default is 100%.

Param	Type	Details
scroll-to-zoom (optional)	boolean	Whether the scroll wheel should control the map zoom. Disabled by default.
target (optional)	string	Specify the link target to use for the map pins. Can be one of '_blank', '_parent', '_self', '_top' or the name of a window. Default value is '_blank'."
url-field (optional)	string	The field containing the URL to make clickable from the map marker.
url-prefix (optional)	string	Use this if the URL retrieved from the URLField needs to be prefixed with a value.
label-field (optional)	string	The field containing the label to use as tooltip for the marker.
loading-overlay (optional)	boolean	Whether to show an overlay when the map is loading new points.
icon (optional)	string	The location of the marker image to use for this set of coordinates.
icon-field (optional)	string	The field from the response to get the icon from.
weight-field (optional)	string	The field containing information about the weight of the result. (Google Only)

Param	Type	Details
geocode-server-url (optional)	string	The URL of the geocode server to turn addresses into coordinates (Esri only)
javascript-api-url (optional)	string	The URL with the location of the JavaScript API (Esri only must be 3.15 or greater)
basemap (optional)	string	The map to use to display the points on (Esri and Google only) Google defaults to: ROADMAP Google Options – HYBRID, ROADMAP, SATELLITE, TERRAIN. Esri defaults to: topo – Esri options (https://developers.arcgis.com/javascript/latest/api-reference/)
webmap-id (optional)	string	The id of the web map to add to the esri map. (Only for esri, you must include the <i>esri webmap tag</i> within the geo:map directive).
options (optional)	object	Extend the current options for the Map (Google Only) (Options Here) (https://developers.google.com/maps/documentation/javascript/reference#MapOptions)
events (optional)	string	Adds support for filtering results based on the current map position (apply), on zoom (zoom), on drag (drag) and also when drawing a rectangle (draw). (You must define latitude and longitude fields for this to work). Valid values: draw Allows the user to draw a box on the map which will then return results within the box. apply Allows the user to search the currently visible area. (Google only) zoom When the user zooms in or out of an area the event "zoom_changed_gmaps" will fire with an object containing the coordinates of the box. (Google only) drag When the user drags the map and the selected area changes the event "dragend_gmaps" will fire with an object containing the coordinates of the box. (Google only)

graph:flow

Description

The Graph Flow Tag displays network data rendered as a Sankey diagram. The body of the Graph Flow tag can contain a *Chart Credit Tag* for adding a credit to the chart and the *Chart Export Tag* for exporting the chart.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see [here](http://www.highcharts.com/docs/chart-design-and-style/style-by-css).

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

The following tags can be nested within the chart sankey tag.

- *Chart Credit Tag*
- *Chart Export Tag*

Usage

as element:

```
<graph:flow platform="{string}" query="{string}" source-field="{string}" target-field="{string}" [measure="{string}"] [measure-field="{string}"] data="{object}" [title="{string}"] [subtitle="{string}"] [series-name="{string}"] [inverted="{boolean}"] [options="{object}"]> </graph:flow>
```

Directive info

- This directive creates new scope.

Parameters

Parameter	Type	Details
platform	string	Name of the platform (Required if not using <code>data</code> attribute).
query	string	Name of the query (Required if not using <code>data</code> attribute).
source-field	string	Name of a primary field to facet on (for example "Country"). The values of this facet will define the set of graph nodes with outgoing edges. (Required if not using <code>data</code> attribute).
target-field	string	Name of a secondary field to facet on (for example "City"), for every facet value of the primary field. The values of this facet will define the set of graph nodes with incoming edges. For example, say the primary facet (source-field) has values for Country=Germany and Country=France and the secondary facet (target-field) has values for City=Berlin, City=Munich and City=Paris. Then we add graph edges Germany→Berlin, Germany→Munich and France→Paris. (Required if not using <code>data</code> attribute).
measure (optional)	string	Name of a numerical measure that defines the weight of each graph edge. When omitted, this defaults to the counts of the facets defined by the target-field. Some possible measures include "sum", "mean" and "min". (Only applicable if not using <code>data</code> attribute).
measure-field (optional)	string	Field to measure, when the "measure" option is specified.

Parameter	Type	Details
data	object	Array of arrays containing the data to render the sankey chart. In the format of <code>['from', 'to', 'weight']</code> <pre>[['Spain', 'Portugal', 100] ['Spain', 'Brazil', 300]]</pre> (Required if not using the <code>platform</code> attribute)
title (optional)	string	Chart title
subtitle (optional)	string	Chart sub title
series-name (optional)	string	The name of the series. (default: '')
inverted (optional)	boolean	Whether the chart should be inverted. (default: true)

Parameter	Type	Details
options (optional)	object	Extend the current options. Options can be found here. (http://api.highcharts.com/highcharts/)

helper:constant

Description

The constant tagf adds a constant to the application.

Usage

as element:

```
<helper:constant name="{string}" value="{string}"> </helper:constant>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
name	string	The name of the constant
value	string	The value of the constant

Example

Source

```
<helper:constant name="test" value="banana"></helper:constant> <span>My Constant: {{test}}</span>
```

```
angular.module('lightning') .controller('ExampleController',  
  ['$scope','$twigkit', function($scope,$twigkit) { $scope.test =  
    $twigkit.getConstant('test','noValue'); }]);
```

Result

My Constant: banana

helper:set

Description

The Set directive sets a variable in the view to a new variable in the view controller.

Usage

as element:

```
<helper:set var="{string}" value="{string}"> </helper:set>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var	string	The name of the new variable
value	string	The variable to copy

Example

Source

```
<helper:set var="myText" value="'My Text Result'"></helper:set> <p>  
<strong>myText:</strong> {{myText}}</p>
```

Result

myText: My Text Result

helper:title

Description

The title directive allows you to set the title for the page.

Usage

as element:

```
<helper:title title="{string}"> </helper:title>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
title	string	The title of the page.

Example

```
<helper:title title="Twigkit Application" </search:title>
```

hide:when

Description

Show or hideWhen element based on expression passed to it.

Usage

as attribute

```
<ANY hide-when="{expression}"> ... </ANY>
```

Parameters

Param	Type	Details
hide-when	expression	Expression to evaluate.

Example

Source

```
<p><a href="javascript:void(0);" ng-click="alert()" styling="button">{{foo ?  
'Show' : 'Hide'}}</a></p> <h3 hide-when="foo" id="example_hello-world">Hello  
World</h3>
```

```
angular.module('lightning') .controller('ExampleController',  
 ['$scope', '$timeout', function($scope,$timeout) { $scope.foo = false;  
 $scope.alert = function(){ $scope.foo = !$scope.foo }; }]);
```

Result

Hide!

Hello World

instant:result-list

Description

The Instant Result List directive allows you to show results matching the users typed query under the search box.

Caveats

- You cannot use an `ng-if` on a direct child of the `<instant:result-list>` tag, instead wrap your `ng-if` with a `<div>` tag. Example:-

Do Not!

```
<search:box>
  <instant:result-list>
    <search:result ng-if="type == 'foo'"></search:result>
    <search:result ng-if="type == 'bar'"></search:result>
  </instant:result-list>
</search:box>
```

Do

```
<search:box>
  <instant:result-list>
    <div>
      <search:result ng-if="type == 'foo'"></search:result>
      <search:result ng-if="type == 'bar'"></search:result>
    </div>
  </instant:result-list>
</search:box>
```

Usage

as element:

```
<instant:result-list [platform="{string}"] [query="{string}"] [children-
selector="{string}"] [wildcard="{string}"]> </instant:result-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
platform (optional)	string	The platform to get the results from, if no platform is defined it will try and use a platform called 'platform'.
query (optional)	string	The query to use against the platform. If the search-within checkbox is checked the filters of this query will be reset. Also if rpp is set on the query then this will be used when getting back the results, otherwise it will be set to 5.
children-selector (optional)	string	A selector for selecting the children elements that should be used for the arrow key selection mechanism. Should be used when the children to be selected are not the immediate children of resultList.
wildcard (optional)	string	Whether to add a wildcard to the search query. Valid values: • left - *searchQuery • right (default) - searchQuery* • both - *searchQuery* • none - searchQuery

layout:animate

Description

Creates an animation when the value (on) is available.

Usage

as element:

```
<layout:animate animation="{string}" on="{object}"> </layout:animate>
```

as attribute

```
<ANY layout-animate animation="{string}" on="{object}"> ... </ANY>
```

Parameters

Param	Type	Details
animation	string	The animation to use when the value is available
on	object	The object to watch for a value

Example

Source

```
<button class="tk-stl-button" ng-click="fadeInUp()">Show Fade In Up Animation</button> <layout:animate animation="animate-fadeInUp" on="fadeInUpValue"> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p></layout:animate> <br /><br /> <button class="tk-stl-button" ng-click="fadeInDown()">Show Fade In Down Animation</button> <layout:animate animation="animate-fadeInDown" on="fadeInDownValue"> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p> </layout:animate>
```



```
angular.module('lightning') .controller('ExampleController', ['$scope',  
function($scope) { $scope.fadeInUp = undefined; $scope.fadeInUp = function(){  
if($scope.fadeInUpValue == undefined){ $scope.fadeInUpValue = 'A Value'; } else {  
$scope.fadeInUpValue = undefined; } } $scope.fadeInDown = function(){  
if($scope.fadeInDownValue == undefined){ $scope.fadeInDownValue = 'A Value'; }  
else { $scope.fadeInDownValue = undefined; } } }]);
```

Result

Show Fade In Up Animation

Show Fade In Down Animation

layout: block

Description

Adds grid block classes to element

Usage

as element:

```
<layout: block [xs="{size}"] [sm="{size}"] [md="{size}"] [lg="{size}"] [xl="{size}"] [xs-offset="{size}"] [sm-offset="{size}"] [md-offset="{size}"] [lg-offset="{size}"] [xl-offset="{size}"] [dynamic="{string}"]> </layout: block>
```

as attribute

```
<ANY layout-block [xs="{size}"] [sm="{size}"] [md="{size}"] [lg="{size}"] [xl="{size}"] [xs-offset="{size}"] [sm-offset="{size}"] [md-offset="{size}"] [lg-offset="{size}"] [xl-offset="{size}"] [dynamic="{string}"]> ... </ANY>
```

Parameters

Param	Type	Details
xs (optional)	size	The grid ratio to use for the smallest break point
sm (optional)	size	The grid ratio to use for the small break point
md (optional)	size	The grid ratio to use for the medium break point
lg (optional)	size	The grid ratio to use for the large break point
xl (optional)	size	The grid ratio to use for above the large break point
xs-offset (optional)	size	The offset ratio to use for the smallest break point
sm-offset (optional)	size	The offset ratio to use for the small break point
md-offset (optional)	size	The offset ratio to use for the medium break point
lg-offset (optional)	size	The offset ratio to use for the large break point
xl-offset (optional)	size	The offset ratio to use for above the large break point
dynamic (optional)	string	Watch for updates to the attributes

Example

Source

```
<layout:grid> <layout:block sm="23-24" md="3-4" lg="1-2" xl="1-4" lg-offset="1-4"
xl-offset="1-2">Content</layout:block> </layout:grid>
```

layout:box

Description

Creates a box

Usage

as element:

```
<layout:box> </layout:box>
```

layout:center-content

Description

Creates a centered content area with some padding

Usage

as element:

```
<layout:center-content> </layout:center-content>
```

layout:content

Description

Creates a content area

Usage

as element:

```
<layout:content> </layout:content>
```

layout:controls

Description

The `<layout:controls>` tag wraps control elements into a group making all the controls on one line or individually for a control to be on a separate line.

Layout controls is required for the following directives:-

- *Sort Tag*
- *Results Per Page Tag*
- *Fields Tag*

Usage

as element:

```
<layout:controls> </layout:controls>
```

layout:grid

Description

Adds grid classes to element

Usage

as element:

```
<layout:grid [fixed="{string}"]> </layout:grid>
```

as attribute

```
<ANY layout-grid [fixed="{string}"]> ... </ANY>
```

Parameters

Param	Type	Details
fixed (optional)	string	If an attribute of 'fixed' is on the element a fixed grid will be used instead

layout:include

Description

The include directives allows you to include a template on the page.

Usage

as element:

```
<layout:include file="{string}"> </layout:include>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
file	string	The location of the file of the template

Example

Source

```
<h3 id="example_static-file-path">Static File Path</h3> <layout:include  
file="css/user-defined-template.tmp.html" user="{{user}}" file-path="  
{{filePath}}"></layout:include> <hr /> <h3 id="example_dynamic-file-path">Dynamic  
File Path</h3> <layout:include file="{{filePath}}" user="{{user}}" file-path="  
{{filePath}}"></layout:include> <hr /> <h3 id="example_dynamic-2-part-file-  
path">Dynamic 2 Part File Path</h3> <layout:include file="{{partFilePath1 +  
partFilePath2}}" user="foo" file-path="{{filePath}}"></layout:include> <hr /> <h3  
id="example_dynamic-file-path+-static-file-path">Dynamic File Path + Static File  
Path</h3> <layout:include file="{{partFilePath1 + 'tmp.html'}}" user="foo" file-  
path="{{filePath}}"></layout:include> <hr /> <h3 id="example_multi-level">Multi-  
level</h3> <layout:include file="css/level1.tpl.html" leveltype="Parent"  
child="css/level2.tpl.html"></layout:include>
```

```
angular.module('lightning').controller('ExampleController', ['$scope',  
function($scope) { $scope.user = 'Developer'; $scope.filePath = 'css/user-  
defined-template.tmp.html'; $scope.partFilePath1 = 'css/user-defined-template.';  
$scope.partFilePath2 = 'tmp.html'; }]);
```

Result

Static File Path

Hey! I'm a custom template that can be included anywhere!

File path is css/user-defined-template.tmp.html

Logged in as Developer.

Dynamic File Path

Hey! I'm a custom template that can be included anywhere!

File path is css/user-defined-template.tmp.html

Logged in as Developer.

Dynamic 2 Part File Path

Hey! I'm a custom template that can be included anywhere!

File path is css/user-defined-template.tmp.html

foo you are not suppose to be here!

Dynamic File Path + Static File Path

Hey! I'm a custom template that can be included anywhere!

File path is css/user-defined-template.tmp.html

foo you are not suppose to be here!

Multi-level

I'm level 1! Parent - css/level2.tpl.html

I'm level 2! css/level3.tpl.html

I'm level 3! Finished = true

Including User-Defined-Template

Hey! I'm a custom template that can be included anywhere!

File path is css/user-defined-template.tmp.html

Foo you are not suppose to be here!

layout:logo

Description

Creates a content area

Usage

as element:

```
<layout:logo [image="{string}"] [image-link="{string}"] [height="{string}"]  
[width="{string}"] [alt="{string}"]> </layout:logo>
```

Parameters

Param	Type	Details
image (optional)	string	The URL of the image to display.
image-link (optional)	string	The URL to redirect to when the image is click. Default: '/'.
height (optional)	string	The height of the logo in px.
width (optional)	string	The width of the logo in px.
alt (optional)	string	The alternative text if the image cannot be displayed.

Example

Source

```
<layout:logo image-link="https://twigkit.com"  
image="//static1.squarespace.com/static/53da228ce4b052fe60de3a3c/t/548af226e4b0dd  
3cb0252f2b/1466005538473/?format=1500w" height="34px" alt="Logo"></layout:logo>
```

Result



layout:sidebar

Description

Creates a sidebar area

Usage

as element:

```
<layout:sidebar> </layout:sidebar>
```

lifecycle:delay

Description

Delays rendering of an object

Usage

as element:

```
<lifecycle:delay delay="{string}"> </lifecycle:delay>
```

Parameters

Param	Type	Details
delay	string	The delay in ms

Example

Source

```
<p>A message will appear after 5 seconds has passed.</p> <lifecycle:delay  
delay="5000"> <search:alert style="alertbox-positive">Alert - 5 Seconds has  
passed.</search:alert> </lifecycle:delay> <p>A message will appear after 10  
seconds has passed.</p> <lifecycle:delay delay="10000"> <search:alert  
style="alertbox-warning">Alert - 10 Seconds has passed.</search:alert>  
</lifecycle:delay> <p>A message will appear after 20 seconds has passed.</p>  
<lifecycle:delay delay="20000"> <search:alert style="alertbox-negative">Alert -  
20 Seconds has passed.</search:alert> </lifecycle:delay>
```

Result

A message will appear after 5 seconds has passed.

A message will appear after 10 seconds has passed.

A message will appear after 20 seconds has passed.

lifecycle:display

Description

Display an element for a period of time

Usage

as element:

```
<lifecycle:display time="{string}"> </lifecycle:display>
```

Parameters

Param	Type	Details
time	string	The time in ms to display the elements

Example

Source

```
<p>A message will disappear after 5 seconds.</p> <lifecycle:display time="5000">
<search:alert style="alertbox-negative">Alert - I disappear after 5 seconds.
</search:alert> </lifecycle:display> <p>A message will disappear after 10 seconds
has passed.</p> <lifecycle:display time="10000"> <search:alert style="alertbox-
warning">Alert - I disappear after 10 seconds.</search:alert>
</lifecycle:display> <p>A message will disappear after 20 seconds has passed.</p>
<lifecycle:display time="20000"> <search:alert style="alertbox-positive">Alert -
I disappear after 20 seconds.</search:alert> </lifecycle:display>
```

Result

A message will disappear after 5 seconds.

Alert - I disappear after 5 seconds.

A message will disappear after 10 seconds has passed.

Alert - I disappear after 10 seconds.

A message will disappear after 20 seconds has passed.

Alert - I disappear after 20 seconds.

lifecycle:event

Description

Broadcasts an event

Usage

as element:

```
<lifecycle:event event="{string}" global="{boolean}"> </lifecycle:event>
```

Parameters

Par am	Type	Details
event	string	The name of the event to broadcast
global	boolean	If true the event is broadcasted to the entire application, otherwise it will just broadcast to it's children. (Default: true)

Example

Source

```
<p>An event called foo will broadcast in 5 seconds.</p> <lifecycle:delay  
delay="5000"> <lifecycle:event event="foo"> <search:alert style="alertbox-  
positive">Event foo has been broadcast</search:alert> </lifecycle:event>  
</lifecycle:delay> <lifecycle:on event="foo"> <lifecycle:delay delay="1000">  
<search:alert style="alertbox-positive">Event foo has been heard so I have  
appeared!</search:alert> </lifecycle:delay> </lifecycle:on> <lifecycle:on  
event="foo" hide="true"> <search:alert style="alertbox-warning">When I hear the  
event foo I will hide!</search:alert> </lifecycle:on>
```

Result

An event called foo will broadcast in 5 seconds.

When I hear the event foo I will hide!

lifecycle:on

Description

Shows or hides elements when an event is broadcast

Usage

as element:

```
<lifecycle:on event="{string}" hide="{boolean}" show="{boolean}"> </lifecycle:on>
```

Parameters

Param	Type	Details
event	string	The name of the event to be listened to
hide	boolean	If true when the event is heard the elements will be hidden.
show	boolean	If true when the event is heard the elements will be shown.

Example

Source

```
<p>An event called foo will broadcast in 5 seconds.</p> <lifecycle:delay  
delay="5000"> <lifecycle:event event="foo"> <search:alert style="alertbox-  
positive">Event foo has been broadcast</search:alert> </lifecycle:event>  
</lifecycle:delay> <lifecycle:on event="foo"> <lifecycle:delay delay="1000">  
<search:alert style="alertbox-positive">Event foo has been heard so I have  
appeared!</search:alert> </lifecycle:delay> </lifecycle:on> <lifecycle:on  
event="foo" hide="true"> <search:alert style="alertbox-warning">When I hear the  
event foo I will hide!</search:alert> </lifecycle:on>
```

Result

An event called foo will broadcast in 5 seconds.

When I hear the event foo I will hide!

lifecycle:required

Description

Renders objects when other object is available;

Usage

as element:

```
<lifecycle:required object="{string}"> </lifecycle:required>
```

Parameters

Param	Type	Details
object	string	The object in question

Example

Source

```
<button class="tk-stl-button" ng-click="toggle()">Toggle</button>  
<lifecycle:required object="value"> <p>I only appear when "value" has a value.  
</p> </lifecycle:required>
```

```
angular.module('lightning') .controller('ExampleController', ['$scope',  
function($scope) { $scope.toggle = function(){ if($scope.value == undefined){  
$scope.value = 'A Value'; } else { $scope.value = undefined; } } }]);
```

Result

Toggle

media:csv

Description

The media:csv tag allows you to export results into csv format.

Usage

as element:

```
<media:csv conf="{string}" query="{object}" [fetch="{number|expression}"] [batch-size="{number}"] [delimiter="{string}"] [fields="{string}"]> </media:csv>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
conf	string	The platform configuration to use e.g. platforms.gsa.foo
query	object	The query to submit to the CSV service
fetch (optional)	number link: [expression]	The number of results or rows to fetch.
batch-size (optional)	number	The number of results or rows to fetch in each batch.
delimiter (optional)	string	Delimiter to use in the CSV
fields (optional)	string	Fields to output, comma-separated or * for all.

media:excel

Description

The media:excel tag allows you to export results into excel format.

Usage

as element:

```
<media:excel conf="{string}" query="{object}" [fetch="{number|expression}"]  
[fields="{string}"] [header="{string}"]> </media:excel>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
conf	string	The platform configuration to use e.g. platforms.gsa.foo
query	object	The query to submit to the CSV service
fetch (optional)	number]link: [expression	The number of results or rows to fetch.
fields (optional)	string	Fields to output, comma-separated or * for all.
header (optional)	string	Replace field names in the header with these header values. Must be a comma separated list of equal length to the fields parameter

media:image

Description

The Image tag displays images from a given field in a result or a `src`. It will automatically resize and cache the image for performance.

The following module will need to be added to your pom.xml in order to use the twigkit image-service.

```
<dependency>
  <groupId>twigkit</groupId>
  <artifactId>twigkit.media</artifactId>
  <version>${project.parent.version}</version>
</dependency>
```

If your image is within an expression you will need to use the trusted filter eg (`<media:image src="{{imageUrl | trusted}}"></media:image>`)

If you are combining plain text with a variable it is recommend that you do this within a function in a controller. You can see this in the example below. This will remove the chance of getting a 404 in the JavaScript console caused by a request for the image being sent before the values in the src attribute were evaluated.

Example

Controller

```
$scope.myImageUrl = function(result){
  return 'http://placeholder.it/' + result.fields.x.val + 'x' + result.fields.y.val;
}
```

View

```
<media:image src="{{myImageUrl(result) | trusted}}"></media:image>
```

Usage

as element:

```
<media:image [result="{object}"] [field-name="{string}"] [field-prefix="{string}"] [field-suffix="{string}"] [src="{string}"] [lazy-load="{boolean}"] [fallback-image-url="{string}"] [width="{number}"] [height="{number}"] [alt="{string}"] [url="{string}"] [cache-original="{boolean}"] [image-service-url="{string}"] [use-headers="{boolean}"]> </media:image>
```

Parameters

Param	Type	Details
result (optional)	object	The result object the field with the image is coming from.
field-name (optional)	string	The field in the result to obtain the image URL.
field-prefix (optional)	string	If the field contains a relative URL, then prefix it with this value.
field-suffix (optional)	string	If the field contains a relative URL, then append this to that value.
src (optional)	string	Rather than specify a field-name, provide a URL to the desired image. If you are using an angular expression you will need to add the <code>#/api/lightning.filter:trusted[trusted filter]</code>. For example, (<code><media:image src="{{imageUrl trusted}}"></media:image></code>)
lazy-load (optional)	boolean	Whether the images should be lazy loaded. Lazy loading works best if image size is defined. Default:false.
fallback-image-url (optional)	string	If no image is found, use the fallback URL provided. If no fallback image URL is provided and the image is not found no image will be displayed.
width (optional)	number	Width of the image to resize to. You can specify either width, height or both.
height (optional)	number	Height of the image to resize to. You can specify either width, height or both.

Param	Type	Details
alt (optional)	string	Provide alternative text for the image.
url (optional)	string	Provide a URL to make the image clickable.
cache-original (optional)	boolean	Whether the write the original image (prior to resizing) to cache. Defaults to true.
image-service-url (optional)	string	Specify an alternative URL for the Twigkit Media Module image service.
use-headers (optional)	boolean	Whether to pass HTTP headers to the underlying image service (default: false).

ng:compile

Description

Compiles HTML string. Works like ngBindHtml but compiles the string.

Usage

as attribute

```
<ANY ng-compile> ... </ANY>
```

Directive info

- This directive creates new scope.

ng:inject

Description

Transcludes the content and sets the scope of the transcluded content to that of the parent directive. ngTransclude would set the scope of the transcluded content to the scope of the view. More information on the reason for this directive can be found here: <https://github.com/angular/angular.js/issues/7874#issuecomment-47644966> (<https://github.com/angular/angular.js/issues/7874#issuecomment-47644966>)

Usage

as element:

```
<ng:inject> </ng:inject>
```

as attribute

```
<ANY ng-inject> ... </ANY>
```

as class

```
<ANY class="ng-inject"> ... </ANY>
```

Directive info

- This directive creates new scope.

padding-bottom

Description

Adds a class for padding bottom based on attribute value

Usage

as attribute

```
<ANY padding-bottom="{number}"> ... </ANY>
```

Parameters

Param	Type	Details
padding-bottom	number	Amount of padding to include, acceptable values (0, 0.25, 0.5, 1 and 2)

Example

Source

```
<p style="bg-tertiary text-tertiary-contrast" padding-bottom="2"> Check out my padding at the bottom (padding-bottom="2") </p> <p style="bg-tertiary text-tertiary-contrast" padding-bottom="1"> Check out my padding at the bottom (padding-bottom="1") </p> <p style="bg-tertiary text-tertiary-contrast" padding-bottom="0.5"> Check out my padding at the bottom (padding-bottom="0.5") </p> <p style="bg-tertiary text-tertiary-contrast" padding-bottom="0.25"> Check out my padding at the bottom (padding-bottom="0.25") </p> <p style="bg-tertiary text-tertiary-contrast" padding-bottom="0"> Check out my padding at the bottom (padding-bottom="0") </p>
```

Result

Check out my padding at the bottom (padding-bottom="2")

Check out my padding at the bottom (padding-bottom="1")

Check out my padding at the bottom (padding-bottom="0.5")

Check out my padding at the bottom (padding-bottom="0.25")

Check out my padding at the bottom (padding-bottom="0")

padding:left

Description

Adds a class for padding left based on attribute value

Usage

as attribute

```
<ANY padding-left="{number}"> ... </ANY>
```

Parameters

Param	Type	Details
padding-left	number	Amount of padding to include, acceptable values (0, 0.25, 0.5, 1 and 2)

Example

Source

```
<p style="bg-tertiary text-tertiary-contrast" padding-left="2"> Check out my padding on the left (padding-left="2") </p> <p style="bg-tertiary text-tertiary-contrast" padding-left="1"> Check out my padding on the left (padding-left="1") </p> <p style="bg-tertiary text-tertiary-contrast" padding-left="0.5"> Check out my padding on the left (padding-left="0.5") </p> <p style="bg-tertiary text-tertiary-contrast" padding-left="0.25"> Check out my padding on the left (padding-left="0.25") </p> <p style="bg-tertiary text-tertiary-contrast" padding-left="0"> Check out my padding on the left (padding-left="0") </p>
```

Result

Check out my padding on the left (padding-left="2")

Check out my padding on the left (padding-left="1")

Check out my padding on the left (padding-left="0.5")

Check out my padding on the left (padding-left="0.25")

Check out my padding on the left (padding-left="0")

padding:right

Description

Adds a class for padding right based on attribute value

Usage

as attribute

```
<ANY padding-right="{number}"> ... </ANY>
```

Parameters

Param	Type	Details
padding-right	number	Amount of padding to include, acceptable values (0, 0.25, 0.5, 1 and 2)

Example

Source

```
<p style="text-align:right;" class="tk-stl-bg-tertiary tk-stl-text-tertiary-contrast" padding-right="2"> Check out my padding on the right (padding-right="2") </p> <p style="text-align:right;" class="tk-stl-bg-tertiary tk-stl-text-tertiary-contrast" padding-right="1"> Check out my padding on the right (padding-right="1") </p> <p style="text-align:right;" class="tk-stl-bg-tertiary tk-stl-text-tertiary-contrast" padding-right="0.5"> Check out my padding on the right (padding-right="0.5") </p> <p style="text-align:right;" class="tk-stl-bg-tertiary tk-stl-text-tertiary-contrast" padding-right="0.25"> Check out my padding on the right (padding-right="0.25") </p> <p style="text-align:right;" class="tk-stl-bg-tertiary tk-stl-text-tertiary-contrast" padding-right="0"> Check out my padding on the right (padding-right="0") </p>
```

Result

Check out my padding on the right (padding-right="2")

Check out my padding on the right (padding-right="1")

Check out my padding on the right (padding-right="0.5")

Check out my padding on the right (padding-right="0.25")

Check out my padding on the right (padding-right="0")

padding-top

Description

Adds a class for padding top based on attribute value

Usage

as attribute

```
<ANY padding-top="{number}"> ... </ANY>
```

Parameters

Param	Type	Details
padding-top	number	Amount of padding to include, acceptable values (0, 0.25, 0.5, 1 and 2)

Example

Source

```
<p style="bg-tertiary text-tertiary-contrast" padding-top="2"> Check out my padding at the top (padding-top="2") </p> <p style="bg-tertiary text-tertiary-contrast" padding-top="1"> Check out my padding at the top (padding-top="1") </p> <p style="bg-tertiary text-tertiary-contrast" padding-top="0.5"> Check out my padding at the top (padding-top="0.5") </p> <p style="bg-tertiary text-tertiary-contrast" padding-top="0.25"> Check out my padding at the top (padding-top="0.25") </p> <p style="bg-tertiary text-tertiary-contrast" padding-top="0"> Check out my padding at the top (padding-top="0") </p>
```

Result

Check out my padding at the top (padding-top="2")

Check out my padding at the top (padding-top="1")

Check out my padding at the top (padding-top="0.5")

Check out my padding at the top (padding-top="0.25")

Check out my padding at the top (padding-top="0")

pivot:table

Description

Display a facet with related filters as a table

Usage

as element:

```
<pivot:table platform="{string}" query="{string}" facet="{string}" expand-query="{string}" expand-facet="{string}" [expand="{boolean}"] [field="{string}"]>
</pivot:table>
```

Parameters

Param	Type	Details
platform	string	The platform to query against
query	string	The query to use
facet	string	The facet to display
expand-query	string	The query to use for expanding
expand-facet	string	The facet to pivot on when expanding
expand (optional)	boolean	Allow expanding. Default: true
field (optional)	string	The filter field name to add to the query

progress:bar

Description

Creates a horizontal progress bar.

Usage

as element:

```
<progress:bar percentage="{string}"> </progress:bar>
```

Parameters

Param	Type	Details
percentage	string	The percentage progress

Example

Source

```
<label for="value">Value</label> <input id="value" type="number"
placeholder="Value" ng-model="value"><br /> <button ng-
click="generateNewNumber()" style="button">Generate Random Percentage</button>
<hr /> <h3 id="example_default-progress-bar">Default Progress bar</h3>
<progress:bar percentage="value"></progress:bar>
```

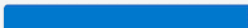
```
angular.module('lightning').controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { $scope.value = 20;
$scope.generateNewNumber = function(){ $scope.value = Math.floor(Math.random() *
(100 - 0 + 1)); }; }]);
```

Result

Value

Generate Random Percentage

Default Progress bar



progress:radial

Description

Creates a radial progress bar

Usage

as element:

```
<progress:radial percentage="{string}" [fill-class="{string}"] [percentage-class="{string}"]> </progress:radial>
```

Parameters

Param	Type	Details
percentage	string	The percentage progress
fill-class (optional)	string	Add a class to the fill elements
percentage-class (optional)	string	Add a class to the percentage element

Example

Source

```
<label for="value">Value</label> <input id="value" type="number"
placeholder="Value" ng-model="value"><br /> <button ng-
click="generateNewNumber()" style="button">Generate Random Percentage</button>
<hr /> <h3 id="example_default-progress-radial">Default Progress Radial</h3>
<progress:radial percentage="value"></progress:radial> <hr /> <h3
id="example_raspberry-progress-radial">Raspberry Progress Radial</h3>
<progress:radial percentage="value" fill-class="tk-stl-bg-raspberry" percentage-
class="tk-stl-text-raspberry"></progress:radial> <hr /> <h3 id="example_emerald-
progress-radial">Emerald Progress Radial</h3> <progress:radial percentage="value"
fill-class="tk-stl-bg-emerald" percentage-class="tk-stl-text-emerald">
</progress:radial>
```

```
angular.module('lightning').controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { $scope.value = 20;
$scope.generateNewNumber = function(){ $scope.value = Math.floor(Math.random() *
(100 - 0 + 1)); }; }]);
```

Result

Value

Generate Random Percentage

Default Progress Radial



Raspberry Progress Radial



Emerald Progress Radial



query:custom

Description

The custom parameter directive is to be used within the *Query tag* to add custom query parameters.

Adding custom parameters

```
<search:query parameters="*" var="query">
  <query:custom name="facet.range" value="year"></query:custom>
  <query:custom name="facet.mincount" value="1"></query:custom>
</search:query>
```

Usage

as element:

```
<query:custom name="{string}" [value="{string}"]> </query:custom>
```

Parameters

Param	Type	Details
name	string	Name of the custom parameter.
value (optional)	string	Value of the custom parameter.

query:filter

Description

The filter directive is to be used within the *Query* tag.

Filter

Remove All Filters with Field Name

```
<search:query parameters="*" var="query">
  <query:filter field="company" action="remove"></query:filter>
</search:query>
```

Add Filter

```
<search:query parameters="*" var="query">
  <query:filter field="company" value="twigkit"></query:filter>
</search:query>
```

Remove Filter

```
<search:query parameters="*" var="query">
  <query:filter field="company" value="twigkit" action="remove"></query:filter>
</search:query>
```

Range Filter

Add Filter

```
<search:query parameters="*" var="query">
  <query:filter field="age" from="25" to="50" display="25 to 50"></query:filter>
</search:query>
```

Remove Filter

```
<search:query parameters="*" var="query">
  <query:filter field="age" from="25" to="50" display="25 to 50" action="remove">
</query:filter>
</search:query>
```

Usage

as element:

```
<query:filter field="{string}" [value="{string}"] [from="{Number|Date}"] [to="{Number|Date}"] [display="{string}"] [optional="{string}"] [exclude="{boolean}"] [action="{string}"] [hidden="{boolean}"] [match="{string}"] [type="{string}"]>
</query:filter>
```

Parameters

Param	Type	Details
field	string	The field the filter applies to.
value (optional)	string	The value to apply to the field
from (optional)	Number] link: [Date	The from value for the range filter.
to (optional)	Number] link: [Date	The to value for the range filter.
display (optional)	string	The display value for the range filter.
optional (optional)	string	If the filter is optional (or) (field or value)
exclude (optional)	boolean	If the filter should exclude values matching this filter.
action (optional)	string	Whether to remove the filter or add the filter. (Default:'add')
hidden (optional)	boolean	Whether the filter should be hidden (visible in breadcrumbs and URLs) or not.

Param	Type	Details
match (optional)	string	Determines whether to match the value of the filter exactly, or partially, with either left or right wildcards. Accepted values are 'left_partial' or 'partial' or 'right_partial' or 'exact'.
type (optional)	string	Explicitly set the type of the filter, 'string' or 'number'. A string filter value will be wrapped in double quotes.

query:suggestions

Description

Query Suggestions

Usage

as element:

```
<query:suggestions completion-service="{string}" [action="{string}"] [service-url="{boolean}"] [show="{number}"] [title="{string}"]> </query:suggestions>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
completion-service	string	Which service or services (as a comma separated list) to use for as-you-type suggestions, as defined in the configuration files for services.
action (optional)	string	Search action path information, e.g. 'search/'.
service-url (optional)	boolean	URL of the web service. Default is '{{context}}/twigkit/services/query-suggestion-service/.
show (optional)	number	The number of suggestions to show.
title (optional)	string	An optional title to display above the list

Example

Source

```
<search:box> <query:suggestions completion-service="services.suggestions.static">
</query:suggestions> </search:box>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ }]);
```

Result

search:advanced

Description

Creates an advanced search form to allow the user to add multiple filters to a search query.

The following tags are commonly nested within a search advanced tag.

- *Search Box Tag*
- *Advanced Text Tag*
- *Advanced Date Range Tag*
- *Advanced Button Tag*

Usage

as element:

```
<search:advanced> </search:advanced>
```

Example

Source

```
<search:advanced> <search:box></search:box> <advanced:text  
fields="source=Source,Material" label="Source"></advanced:text> <advanced:date-  
range fields="source=Source,Material"></advanced:date-range> </search:advanced>
```

Result

The screenshot shows a web interface for an advanced search. At the top is a search bar with the placeholder text 'Search' and a blue 'Search' button. Below the search bar are two filter sections. The first section is labeled 'Source' and contains a 'Select' dropdown menu, a 'Exact match' dropdown menu, and a plus sign. Below this is a text input field labeled 'Search terms'. The second section is labeled 'From' and 'To'. It contains a 'From Date' text input field, a 'To Date' text input field, a 'Select' dropdown menu, a 'Match' dropdown menu, and a plus sign.

search:alert

Description

Creates an alert box with the transcluded content

Usage

as element:

```
<search:alert> </search:alert>
```

Example

Source

```
<tk-example> <search:alert>Did you mean the shop <a  
href="#">kitchensinks.co.uk</a>?</search:alert> <search:alert styling="alertbox-  
secondary">Something you can probably ignore has happened.</search:alert>  
<search:alert styling="alertbox-negative">Something bad has happened.  
</search:alert> <search:alert styling="alertbox-warning">Something slightly  
worrying has happened.</search:alert> <search:alert styling="alertbox-  
positive">Something good has happened.</search:alert> </tk-example> <tk-code-  
wrapper> <code hljs hljs-no-escape>&lt;search:alert&gt;...&lt;/search:alert&gt;  
</code> <code hljs hljs-no-escape>&lt;search:alert styling="alertbox-  
secondary"&gt;...&lt;/search:alert&gt;</code> <code hljs hljs-no-  
escape>&lt;search:alert styling="alertbox-negative"&gt;...&lt;/search:alert&gt;  
</code> <code hljs hljs-no-escape>&lt;search:alert styling="alertbox-  
warning"&gt;...&lt;/search:alert&gt;</code> <code hljs hljs-no-  
escape>&lt;search:alert styling="alertbox-positive"&gt;...&lt;/search:alert&gt;  
</code> </tk-code-wrapper>
```

Result

EXAMPLE

Did you mean the shop kitchensinks.co.uk? ✕

Something you can probably ignore has happened. ✕

Something bad has happened. ✕

Something slightly worrying has happened. ✕

Something good has happened. ✕

```
<search:alert>...</search:alert>  
<search:alert styling="alertbox-secondary">...</search:alert>  
<search:alert styling="alertbox-negative">...</search:alert>  
<search:alert styling="alertbox-warning">...</search:alert>  
<search:alert styling="alertbox-positive">...</search:alert>
```

search:box

Description

The Header Search directive

Usage

as element:

```
<search:box response="{Object}" [query="{string}"] [action="{string}"]  
[placeholder="{string}"] [button-label="{string}"] [delay="{number}"] [platform="{  
{string}"}] [size="{string}"] [social="{boolean}"] [search-within-checkbox="  
{boolean}"] [search-within-label="{boolean}"] [search-within-default-value="  
{boolean}"] [min-length="{number}"] [pre-search="{function}"]> </search:box>
```

Parameters

Param	Type	Details
response	Object	A response object
query (optional)	string	The name of the query object
action (optional)	string	Page the search should take place on, results will not change automatically when an action is set.
placeholder (optional)	string	Placeholder for the search box
button-label (optional)	string	Label of the button.
delay (optional)	number	The delay before searching, not applicable if an action is set or instant search is being used. If a delay is greater than 1, results will load instantly on the page.
platform (optional)	string	The platform to use for social search
size (optional)	string	The size of the search box (tiny, small, large, huge, massive)
social (optional)	boolean	Whether to enable the social features of the search header. Defaults to false. If you require a customized <i>Bookmark List Tag</i> or <i>Saved Query List Tag</i> you can add your own inside the search box tag and these will be used instead.

Param	Type	Details
search-within-checkbox (optional)	boolean	Show the 'search within' option as a checkbox. This enables the query to retain the filters already applied when checked. If not checked the search is reset when a query is entered and the form submitted. If you want this functionality but don't want a checkbox showing, you can use this attribute set to be false, and set <code>search-within-default-value</code> to be true - this uses a hidden field instead of a checkbox.
search-within-label (optional)	boolean	The label for the 'search within' checkbox.
search-within-default-value (optional)	boolean	Whether to enable 'search within' by default. The default is option is 'false'.
min-length (optional)	number	The number of characters before a search is performed, only applicable for instant-results, query suggestions, advanced search or when a delay without an action. Default: 2
pre-search (optional)	function	Callback function that will be called on a search submit before the request is sent, receives the current URL state as a parameter that can be updated.

Example

Source

```
<h3 id="example_search">Search</h3> <search:box></search:box> <hr /> <h3
id="example_search-with-social">Search with Social</h3> <search:box
social="true"></search:box> <hr /> <h3 id="example_search-with-search-
within">Search with Search Within</h3> <search:box search-within-checkbox="true">
</search:box> <hr /> <h3 id="example_search-with-search-within-with-label">Search
with Search Within with Label</h3> <search:box search-within-checkbox="true"
search-within-label="Keep filters?"></search:box> <hr /> <h3 id="example_search-
with-search-within-with-label-and-default">Search with Search Within with Label
and Default</h3> <search:box search-within-checkbox="true" search-within-
label="Keep filters?" search-within-default-value="true"></search:box> <hr /> <h3
id="example_search-with-search-within-with-no-checkbox">Search with Search Within
with no checkbox</h3> <search:box search-within-checkbox="false" search-within-
default-value="true"></search:box>
```

Result

Search

Search with Social



Search with Search Within

☐

Search with Search Within with Label

☐ Keep filters?

Search with Search Within with Label and Default

☒ Keep filters?

Search with Search Within with no checkbox

search:breadcrumbs

Description

The Breadcrumbs directive.

Usage

as element:

```
<search:breadcrumbs [response="{object}"] [query="{string}"] [group-by-field="{boolean}"] [show-field-name="{boolean}"] [show-query="{boolean}"] [max-characters="{number}"] [action="{string}"] [optional-label="{string}"] [date-format="{string}"] [exclude="{string}"]> </search:breadcrumbs>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response (optional)	object	A response object
query (optional)	string	A query object
group-by-field (optional)	boolean	Group filters by field name (only available if query is not in grouping mode).
show-field-name (optional)	boolean	Whether to show field name as a part of the breadcrumb (only available when group by field is true), Default:true.
show-query (optional)	boolean	Whether to include the user query text as a breadcrumb. Default:true
max-characters (optional)	number	Limit display value to a certain number of characters, adding '...' if maxCharacters is exceeded.
action (optional)	string	Alternative page to submit the query parameters to – defaults to current page.
optional-label (optional)	string	Label to show when the Filter is optional.
date-format (optional)	string	Format to use for date values if no display value is specified. See #/api/lightning.filter:dateFormat[date format filter] for valid formats.
exclude (optional)	string	Comma separated list of fields to exclude, you can also use field=value to exclude only fields with the value provided.

Example

Source

```
<tk-example> <search:breadcrumbs response="response" group-by-field="true" date-format="shortDate" exclude="problem"></search:breadcrumbs> </tk-example> <tk-code-wrapper> <code hljs><search:breadcrumbs response="response" group-by-field="true" date-format="shortDate"></search:breadcrumbs></code> </tk-code-wrapper>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { $scope.response = { page: 2,
query: { rpp: 50, filters:[ { field:'file type', val:'pdf' }, { field:'type',
val:'man' }, { val:{ act: { type: "range", from: "2016-08-05T12:50:30.660", to:
"2016-08-10T12:50:40.509", fromExcluded: false, toExcluded:false }, dsp:'2016-08-
05T12:50:30.660Z - 2016-08-10T12:50:40.509Z' }, field:'date' }, { val: { act: {
type: "range", from: 100, to: "200", fromExcluded: false, toExcluded: false },
dsp: '100 - 200' }, field: 'numbers' }, { val: { act:'foo', dsp:'bar' },
field:'problem' } ] }, facets: {} } }]);
```

Result

EXAMPLE

file type × pdf × type × man × date × 8/5/16 - 8/10/16 × numbers × 100 - 200 × Clear all

```
<search:breadcrumbs response="response" group-by-field="true" date-format="shortDate">
</search:breadcrumbs>
```

search:conditional

Description

The search conditional directive

Usage

as element:

```
<search:conditional [test="{string}"] [response="{object}"]>
</search:conditional>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
test (optional)	string	The expression to test conditional logic for. Only render the contents tag if this evaluates to true.
response (optional)	object	A Response object to check, the tag will render the body if the Response is not null.

Example

Source

```
<h3 id="example_test-attribute">Test Attribute</h3> <search:conditional
test="test1 > 100"> <p>False don't display</p> </search:conditional>
<search:conditional test="test1 >= 100"> <p>True! test1 is greater than or equal
to 100</p> </search:conditional> <search:conditional test="test2"> <p>True! test2
is true</p> </search:conditional> <search:conditional test="test3 == 'Foo'">
<p>False don't display</p> </search:conditional> <search:conditional test="test3
== 'foo'"> <p>True test3 does equal foo</p> </search:conditional> <hr/> <h3
id="example_response-attribute">Response Attribute</h3> <search:conditional
response="response"> <p>True! Response does exist.</p> </search:conditional>
<search:conditional response="response2"> <p>False! Response2 does not exist.</p>
</search:conditional>
```

```
angular.module('lightning') .controller('ExampleController',  
  ['$scope','ResponseService', function($scope,ResponseService) { $scope.test1 =  
  100; $scope.test2 = true; $scope.test3 = 'foo'  
  ResponseService.setResponse('response', {foo:'bar'}); }]);
```

Result

Test Attribute

True! test1 is greater than or equal to 100

True! test2 is true

True test3 does equal foo

Response Attribute

True! Response does exist.

search:facet

Description

The facet directive displays an individual facet. Either a response object and facet name must be passed to it, or a facet object.

Search Within Facet

To enable search within facets the following attributes must be set on the facet `search-enabled="true"`. The following attributes can be set on either the facet-list tag or the facet tag to enable search within `platform` and `query`.

If using a string field such as `facet_field_s` then you should add the attribute `facet-field` to your facet tag with the tokenized version of that field eg. `facet-field="facet_field_t"`, for more information see the `search-enabled` and `facet-field` attributes.

Multi-term facet queries

A field named `location_facet` might have several facets, such as 'AGL Centre', 'Airport City', and 'Airport Road'. When you search for 'AGL c', you want to see 'AGL Centre' in the results.

When conducting partial searches on multi-term facets, tokenize the searched field using `solr.KeywordTokenizerFactory`.

Lower-case transformation is crucial for case-insensitive matching. You can use the built-in `lowercase` field type in Fusion:

```
<fieldType name="lowercase" class="solr.TextField" positionIncrementGap="100">
<analyzer> <tokenizer class="solr.KeywordTokenizerFactory"/> <filter
class="solr.LowerCaseFilterFactory"/> </analyzer> </fieldType>
```

This applies a tokenizer and a lower-case filter to get partial and case-insensitive string matches when searching within facets.

Usage

as element:

```
<search:facet [response="{string}"] [facet-name="{string}"] [facet="{
{expression}"] [title="{string}"] [show="{string}"] [show-more="{string}"] [max-
characters="{string}"] [select="{string}"] [platform="{string}"] [query="
{string}"] [search-enabled="{boolean}"] [wildcard="{string}"] [facet-field="
{string}"] [collapsible="{boolean}"] [show-empty="{boolean}"] [collapsed-by-
default="{boolean}"] [show-count="{boolean}"] [action="{string}"] [count-number="
{string}"] [date-format="{string}"] [measure="{string}"] [when-field="{string}"]
[exclude="{boolean}"]> </search:facet>
```

Directive info

- This directive creates new scope.

Parameters

Parameter	Type	Details
response (optional)	string	The name of the response to use.
facet-name (optional)	string	Facet name to use from the response object
facet (optional)	expression	A facet object to display
title (optional)	string	The title of the facet to display, defaults to the name of the facet
show (optional)	string	The number of filters to show by default
show-more (optional)	string	The number of extra filters to display when the show more link is clicked
max-characters (optional)	string	Limit display value to a certain number of characters, adding '...' if max-characters is exceeded.

Param	Type	Details
select (optional)	string	How to render each FacetFilter. Valid values: • click User selects a filter by clicking the value; (default) • multiselect Supports selecting multiple filters that get combined with an 'OR'. • toggle Only allows one filter to be selected at a time but shows unselected filters and counts • multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.
platform (optional)	string	The name of the platform to use for search within the facet.
query (optional)	string	None of the query to be used.
search-enabled (optional)	boolean	Whether to enable search on the facet. If no query or platform is defined on the facet or facet-list it will enable the user to search through the filters that have been returned in the response. If a platform and query is defined on the facet or facet-list then it will search for filters in the search index. (Requires a tokenized field, if not using a tokenized field, if not using a tokenized field please see facet-field).
wildcard (optional)	string	If search-enabled is true you can set wildcard rules for this search. Valid values are: <i>left</i> : This adds a wildcard prefix to facet search terms, for example "term". <i>right</i> : This adds a wildcard suffix to facet search terms, for example "term". <i>both</i> (default): This adds both a wildcard prefix and suffix to facet search terms, for example "term". <i>none</i> : This adds no wildcards to the facet search term (only exact matches).
facet-field (optional)	string	When this option is provided, the user will have the option to search values within the facet using the named field to search against. The named field should be tokenized. Example usage is if you are using facet_field_s (a string facet/field) as your facet you need to provide the tokenized version of that field for search against, e.g. facet_field_t.

Param	Type	Details
collapsible (optional)	boolean	Set to true if the facet should be collapsible in the list of facets. (default: false)
show-empty (optional)	boolean	If true then the facet will also output empty filters where count is zero. (default: false)
collapsed-by-default (optional)	boolean	Set to true if this facet should be collapsed by default. (default: false)
show-count (optional)	boolean	Whether to add metadata with frequency counts etc. (default: true)
action (optional)	string	Alternative Action to submit the query parameters to - defaults to current page.
count-number (optional)	string	Specify how the number representing the count should be formatted. Valid values: plain Displays the number without any formatting. formatted Formatted means that the number is comma formatted (2,000,000 for 2000000). (default) rounded Rounded means the number is represented in a short format (2k for ~2000).

Param	Type	Details
date-format (optional)	string	How to format the date. See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.
measure (optional)	string	What fact to use as an alternative to count as quantitative value for a filter.
when-field (optional)	string	Comma-separated list of field names. When one matches this facet representation will be used during an iteration.
exclude (optional)	boolean	On filter hover, displays an icon to allow the user to add an exclusion filter for the value. (default: <i>false</i>)

Example

Source

```
<search:facet-list response="response" styling="facet-list facet-list-step facet-list-wrappedheader facet-list-wrappedheader-noborder"> <search:facet facet-name="test1" title="First Pass Files" collapsible="true"></search:facet>
<search:facet facet-name="test2" title="Second Pass Files"></search:facet>
<search:facet facet-name="empty" title="Empty Facet"></search:facet>
<search:facet facet-name="date" title="Date Facet With Switches" date-format="fullDate" select="multiselect" styling="switches"></search:facet>
<search:facet facet-name="date2" title="Date Facet With Checkboxes" date-format="fullDate" select="multiselect"></search:facet> <search:facet facet-name="date3" title="Date Facet With Format" date-format="dd/MM/yyyy">
</search:facet> </search:facet-list> <search:facet-list response="response" styling="facet-list facet-list-step facet-list-wrappedheader facet-list-wrappedheader-noborder" facet-names="*"> <search:facet facet-name="date3" title="Date Facet With Format" date-format="dd/MM/yyyy"></search:facet>
<search:facet></search:facet> </search:facet-list> <h5 id="example_source_show-count---false">Show Count - false</h5> <search:facet-list response="response" styling="facet-list facet-list-step facet-list-wrappedheader facet-list-wrappedheader-noborder" facet-names="*"> <search:facet facet-name="date3" title="Date Facet With Format" date-format="dd/MM/yyyy" show-count="false">
</search:facet> <search:facet show-count="false"></search:facet> </search:facet-list>
```

```
angular.module('lightning').controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { test1: { filters: [
{ val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val: { dsp: 'Day 2',
act: 'Day 2', }, count: 200, } ], display:'Test 1' }, test2: { filters: [ { val:
{ dsp: 'Day 1', act: 'Day 1', }, count: 10, }, { val: { dsp: 'Day 2', act: 'Day
2', }, count: 20, } ], display:'Test 2' }, empty: { filters: [] }, date: {
filters: [ { val: { dsp: 1288323623006, act: 1288323623006, }, count: 10, }, {
val: { dsp: 1467882419615, act: 1467882419615, }, count: 20, } ], 'max-count':20,
display:'Date' }, date2: { filters: [ { val: { dsp: 1288323623006, act:
1288323623006, }, count: 10, }, { val: { dsp: 1467882419615, act: 1467882419615,
}, count: 20, } ], 'max-count':20, display:'date2' }, date3:{ filters:[ { val:{
act: { type: "range", from: "2016-08-05T12:50:30", to: "2016-08-05T12:50:40",
fromExcluded: false, toExcluded:false }, dsp:'2016-08-05T12:50:30Z' }, count:10 }
], 'max-count':10, display:'date3' }, }, results: [] }
ResponseService.setResponse('response', $scope.response); }]);
```

Result

First Pass Files

Day 1	100
Day 2	200

Second Pass Files

Day 1	10
Day 2	20

Date Facet With Switches

☐ Thursday, October 28, 2010	10
☐ Thursday, July 7, 2016	20

Date Facet With Checkboxes

☐ Thursday, October 28, 2010	10
☐ Thursday, July 7, 2016	20

Date Facet With Format

05/08/2016	10

Test 1

Day 1	100
Day 2	200

Test 2

Day 1	10
Day 2	20

Date

1288323623006	10
1467882419615	20

date2

1288323623006	10
1467882419615	20

Date Facet With Format

05/08/2016	10

Show Count - false

Test 1

Day 1
Day 2

Day 2

Test 2

Day 1

Day 2

Date

1288323623006

1467882419615

date2

1288323623006

1467882419615

Date Facet With Format

05/08/2016

search:facet-list

Description

The facet list directive displays a list of facets.

Usage

as element:

```
<search:facet-list [response="{object}"] [styling="{string}"] [facet-names="{string}"] [max-characters="{number}"] [platform="{string}"] [query="{Object}"]>
</search:facet-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response (optional)	object	A response object
styling (optional)	string	The style the facet list is to be rendered in. Accepted values 'shadow' and 'wrappedheader'
facet-names (optional)	string	The names of the facets to output. Can be specified as either a comma separated list or as a wildcard * for all available facets. In addition, you can use the - with a facet name to omit a specific facet (e.g. -cat will omit the cat facet).
max-characters (optional)	number	Limit display value to a certain number of characters, adding '...' if maxCharacters is exceeded. This will not break highlighting code.
platform (optional)	string	The name of the platform to use for search within the facet.
query (optional)	object	Query to be used.

Example

Source

```
<h3 id="example_styling">Styling</h3> <h4 id="example_styling_simple">Simple</h4>
<search:facet-list response="response" facet-names="*"></search:facet-list> <h4
id="example_styling_wrapped-header">Wrapped Header</h4> <search:facet-list
response="response" styling="wrappedheader" facet-names="*"></search:facet-list>
<h4 id="example_styling_indented">Indented</h4> <search:facet-list
response="response" styling="indented" facet-names="*"></search:facet-list> <h4
id="example_styling_shadow">Shadow</h4> <search:facet-list response="response"
styling="shadow" facet-names="*"></search:facet-list> <h4
id="example_styling_hidemetadata">Hidemetadata</h4> <search:facet-list
response="response" styling="hidemetadata" facet-names="*"></search:facet-list>
<h4 id="example_styling_recessed">Recessed</h4> <search:facet-list
response="response" styling="recessed" facet-names="*"></search:facet-list> <h4
id="example_styling_light">Light</h4> <div class="tk-stl-blocksidebar-light">
<aside class="tk-bl-sidebar"> <search:facet-list response="response" facet-
names="*"></search:facet-list> </aside> </div> <h4
id="example_styling_dark">Dark</h4> <div class="tk-stl-blocksidebar-dark"> <aside
class="tk-bl-sidebar"> <search:facet-list response="response" facet-names="*">
</search:facet-list> </aside> </div>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { test1: { id:'test1',
display:'test1', filters: [ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 100,
}, { val: { dsp: 'Day 2', act: 'Day 2', }, count: 200, } ] }, test2: {
id:'test2', display:'test2', filters: [ { val: { dsp: 'Day 1', act: 'Day 1', },
count: 10, }, { val: { dsp: 'Day 2', act: 'Day 2', }, count: 20, } ] }, empty: {
id:'empty', display:'Empty', filters: [ ] }, }, results: [ ] }
ResponseService.setResponse('response', $scope.response); }]);
```

Result

Styling

Simple

test1	
Day 1	100
Day 2	200
test2	
Day 1	10
Day 2	20

Wrapped Header

test1	
Day 1	100
Day 2	200
test2	
Day 1	10
Day 2	20

Indented

test1	
Day 1	100
Day 2	200
test2	
Day 1	10
Day 2	20

Shadow

test1	
Day 1	100
Day 2	200
test2	
Day 1	10
Day 2	20

Hidemetadata

test1

Day 1

Day 2

test2

Day 1

Day 2

Recessed

test1

Day 1

Day 2

100

200

test2

Day 1

Day 2

10

20

Light

TEST1

Day 1

Day 2

100

200

TEST2

Day 1

Day 2

10

20

Dark

TEST1

Day 1

Day 2

100

200

TEST2

Day 1

Day 2

10

20

search:field

Description

The Field directive.

Usage

as element:

```
<search:field name="{string}" [result="{object}"] [query="{object}"] [max-
characters="{string}"] [date-format="{string}"] [number-format="{string}"] [url="{
string}"] [urlfield="{string}"] [target="{string}"] [action="{string}"]
[default-value="{string}"] [label="{string}"] [value-element="{string}"] [label-
element="{string}"] [prefix="{string}"] [suffix="{string}"] [show-more="{
string}"] [show-less="{string}"] [show="{number}"] [show-more-values="{number}"]
[separator="{string}"] [url-prefix="{string}"] [url-suffix="{string}"] [filter-
optional="{string}"]> </search:field>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
name	string	Name of the field to be rendered from the result
result (optional)	object	The result containing the field to be rendered
query (optional)	object	Providing a Query object means that the Value will be displayed as a hyperlink that submits the Query again with this field and its value as a Filt
max-characters (optional)	string	Limit display value to a certain number of characters, adding '...' if maxCharacters is exceeded.
date-format (optional)	string	How to format the date. See #/api/lightning.filter:dateFormat[date format filter] for valid formats.
number-format (optional)	string	Format string for the numbers used using the Java DecimalFormat pattern. (default: ",0.")
url (optional)	string	URL to redirect to when the field is clicked;
urlfield (optional)	string	Field containing a URL to redirect to when the field is clicked;
target (optional)	string	Set an HTML target attribute for the anchor tag. For example use <code>_blank</code> to have the generated link open the results in a new window. (Not applicable to query)
action (optional)	string	Alternative page to submit the query parameters to – defaults to current page. If a 'url' attribute is provided the action is ignored and query parameters are not included in the request.

Param	Type	Details
default-value (optional)	string	Default value to show when no value is present in the index.
label (optional)	string	A label to use instead of the default value provided by the Platform. If this attribute is set to an empty String ("") then the label container HTML is omitted from the output.
value-element (optional)	string	The HTML element to use for the value (Default p). (default: "p")
label-element (optional)	string	The HTML element to use for the label (Default p) (default: "p")
prefix (optional)	string	Text to prefix index values with (not applied to default value). E.g. if the index field 'price' contains the value '10'. A prefix="US\$" would result in 'US\$10' would be displayed.
suffix (optional)	string	Text to append to index values (not applied to default value). E.g. if the index field 'weight' contains the value '10'. A suffix=" kg." would result in '10 kg.' being displayed as the value.
show-more (optional)	string	If the display value is limited by the max-characters, a show more link will be shown. The value of this attribute will be used as the label. (default: "Show more")
show-less (optional)	string	If the display value is limited by the max-characters, a show less link will be shown. The value of this attribute will be used as the label. (default: "Show less")
show (optional)	number	Number of values to show if the field is multivalued. (default: -1)

Param	Type	Details
show-more-values (optional)	number	Set to the number of results you'd like the user to see after clicking the show more link. Set to <code>-1</code> to show all the remaining values for the field.
separator (optional)	string	Separator character or sequence to split a Field into multiple values. Setting the separator to ";" would turn the value "company1; company2; company3" into a list of 3 values.
url-prefix (optional)	string	Text to add as a prefix to the URL.
url-suffix (optional)	string	Text to add as a suffix to the URL.
filter-optional (optional)	string	If the fields filter is optional (or). Can be set to 'value', 'field' or 'none'. Default: none

Example

Source

```
<layout:grid> <layout:block> <h3 id="example_with-response">With Response</h3>
</layout:block> <search:result-list response="response" styling="cards-sm-1
cards-md-2"> <search:result> <layout:block> <div class="tk-stl-card">
<search:field name="name" styling="title" urlfield="url"></search:field>
<search:field name="age" styling="label-left" label="Age"></search:field>
<search:field name="date" styling="label-left" label="Created Date" date-
format="dd/MM/yy"></search:field> <search:field name="blank" styling="label-left"
label="Contact Number" default-value="No Contact Number on File."></search:field>
<search:field name="number" styling="label-left" label="Worth" number-
format="f#,###.##"></search:field> <search:field name="madeup" styling="label-
left" label="Test"></search:field> <search:field name="longstring"
styling="label-left" label="Long String" max-characters="40" show-more>
</search:field> </div> </layout:block> </search:result> </search:result-list>
</layout:grid> <layout:grid> <layout:block> <h3 id="example_with-results">With
Results</h3> </layout:block> <search:result-list results="results"
styling="cards-sm-1 cards-md-2"> <search:result> <layout:block> <div class="tk-
stl-card"> <search:field name="name" styling="title" urlfield="url">
</search:field> <search:field name="age" styling="label-left" label="Age">
</search:field> <search:field name="date" styling="label-left" label="Created
Date" date-format="dd/MM/yy"></search:field> <search:field name="blank"
styling="label-left" label="Contact Number" default-value="No Contact Number on
File."></search:field> <search:field name="number" styling="label-left"
label="Worth" number-format="f#,###.##"></search:field> <search:field
name="madeup" styling="label-left" label="Test"></search:field> </div>
</layout:block> </search:result> </search:result-list> </layout:grid>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, hits:{ act:['long','1234'] },
facets: { }, results: [ { result: { fields: { name: { name: 'name', val:
['Testing Tester'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: {
name: 'age', val: ['Long', 35] }, date: { name: 'date', val: ['2014-11-
11T00:00:00'] }, blank:{ name:'blank', val:null, }, number:{ name:'number', val:
[123456789] }, longstring:{ name:'longstring', val:['Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.']} } }, }, { result: { fields: { name: { name: 'name', val:
['Foo (Bar)'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: { name:
'age', val: ['Long', 26] }, date: { name: 'date', val: ['2015-11-11T00:00:00'] },
blank:{ name:'blank', val:null, }, number:{ name:'number', val:[-123456789] } },
}, ], } $scope.results = $scope.response.results;
ResponseService.setResponse('response', $scope.response); }]);
```

Result

With Response

Testing Tester

Age	Long 35
Created Date	11/11/14
Contact Number	No Contact Number on File.
Worth	£123,456,789
Long String	Lorem ipsum dolor sit amet, consectetur ... show more

Foo (Bar)

Age	Long 26
Created Date	11/11/15
Contact Number	No Contact Number on File.
Worth	-£123,456,789

With Results

Testing Tester

Age	Long 35
Created Date	11/11/14
Contact Number	No Contact Number on File.
Worth	£123,456,789

Foo (Bar)

Age	Long 26
Created Date	11/11/15
Contact Number	No Contact Number on File.
Worth	-£123,456,789

search:fields

Description

A dropdown to select the fields which should be returned in the response, if no fields are selected all fields will be returned.

The fields tag should be wrapped in a *Layout Controls Tag*.

Usage

as element:

```
<search:fields [title="{string}"] [fields="{string}"] [hidden-fields="{string}"]>
</search:fields>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
title (optional)	string	The title to display for the dropdown.
fields (optional)	string	A comma separated list of the fields that can be selected. Fields can be added with a display value like so actualFieldName=Display Field Name, anotherActual=My Field.
hidden-fields (optional)	string	A comma separated list of fields to be included with fields the user selects. Eg. id,firstname,lastname

Example

Source

```
<layout:controls> <search:fields
fields="name=Name,current_address=Address,Gender"></search:fields>
</layout:controls>
```

Result

Fields ▾

search:filter

Description

This has been deprecated see *Query Filter*.

The filter directive can be used to add a filter to the query.

Usage

as element:

```
<search:filter field="{string}" value="{string}" [optional="{boolean}"]  
[exclude="{boolean}"] [hidden="{boolean}"]> </search:filter>
```

Parameters

Param	Type	Details
field	string	The field the filter applies to.
value	string	The value to apply to the field
optional (optional)	boolean	If the filter is optional (or)
exclude (optional)	boolean	If the filter should exclude values matching this filter.
hidden (optional)	boolean	Whether the filter should be hidden (visible in breadcrumbs and URLs) or not.

search:filter-list

Description

Filter List Directive

Usage

as element:

```
<search:filter-list [facet="{expression}"] [facet-name="{string}"] [title="{string}"] [show="{string}"] [show-more="{string}"] [max-characters="{string}"] [select="{string}"] [show-empty="{boolean}"] [show-more-label="{string}"] [show-less-label="{string}"] [show-count="{boolean}"] [action="{string}"] [order="{string}"] [count-number="{string}"] [date-format="{string}"] [measure="{string}"] [exclude="{boolean}"]> </search:filter-list>
```

Directive info

- This directive creates new scope.

Parameters

Par am	Ty pe	Details
facet (optional)	expression	A facet object to display
facet-name (optional)	string	Facet name to use from the response object
title (optional)	string	The title of the facet to display, defaults to the name of the facet
show (optional)	string	The number of filters to show by default
show-more (optional)	string	The number of extra filters to display when the show more link is clicked
max- characters (optional)	string	Limit display value to a certain number of characters, adding '...' if maxCharacters is exceeded.

Par am	Type	Details
select (optional)	string	How to render each FacetFilter. 'click': User selects a filter by clicking the value; 'multiselect': Supports selecting multiple filters that get combined with an 'OR'. 'toggle': Only allows one filter to be selected at a time but shows unselected filters and counts. 'multiselect-instant': Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically. Default: 'click'
show-empty (optional)	boolean	If true then the filter list will also output empty filters where count is zero. Default: false
show-more-label (optional)	string	Text to show on the link that shows more filters (expandable mode). Default: 'Show more'.
show-less-label (optional)	string	Text to show on the link that shows fewer filters (expandable mode). Default: 'Show less'.
show-count (optional)	boolean	Whether to add metadata with frequency counts etc. Default: true
action (optional)	string	Alternative Action to submit the query parameters to - defaults to current page.

Par am	Ty pe	Details
ord er (op tio nal)	st ri ng	Specify a sort order for this filter list. Allowed values are: 'alphabetical', '-alphabetical', '-count' and 'count'. Default is whatever the search engine provides.
cou nt- num ber (op tio nal)	st ri ng	Specify how the number representing the count should be formatted. Optional values are 'plain', 'formatted' and 'rounded'. Rounded means the number is represented in a short format (2k for ~2000). Formatted means that the number is comma formatted (2,000,000 for 2000000).Default: 'formatted'
dat e- for mat (op tio nal)	st ri ng	How to format the date. See #/api/lightning.filter:dateFormat[date format filter] for valid formats.
mea sur e (op tio nal)	st ri ng	What fact to use as an alternative to count as quantitative value for a filter.
exc lud e (op tio nal)	bo ol ean	On filter hover, displays an icon to allow the user to add an exclusion filter for the value. Default: false

search:keyword

Description

Search keyword tag allows you to add the ability for a user to filter the search based on the keyword they enter and the value they select.

Usage

as element:

```
<search:keyword keyword="{string}" facet="{string}" [facet-field="{string}"]  
[description="{string}"] [date="{string}"] [date-format="{string}"] [date-  
display-format="{string}"]> </search:keyword>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
keyword	string	The keyword the user types
facet	string	The facet the keyword is associated with
facet-field (optional)	string	The facet the keyword is associated with
description (optional)	string	The description to display next to the keyword
date (optional)	string	Type of date (on, before, after. Default: on)
date-format (optional)	string	Date format to use for the datapicker. Default yyyy-MM-ddTHH:mm:ss
date-display-format (optional)	string	Date format to use for the datapicker display values. Default yyyy-MM-ddTHH:mm:ss

Example

Source

```
<search:advanced keywords="keywords" response="response" platform="platform">
<search:keyword keyword="from" facet="Start_Date" date="after"
description="Creation date"></search:keyword> <search:keyword keyword="type"
facet="Study_Types" description="Type of study"></search:keyword> <search:keyword
keyword="gender" facet="Gender" description="Patient gender"></search:keyword>
<search:keyword keyword="Conditions"></search:keyword> <search:keyword
keyword="Recruitment"></search:keyword> <search:keyword keyword="Sponsors"
facet="Sponsor_Collaborators"></search:keyword> <instant:result-list
platform="platform"> <search:result> <a ng-href="/#/result/{{
result.fields.id.val | actualValue }}">{{ result.fields.Title.val | displayValue
}}</a> </search:result> </instant:result-list> </search:advanced>
```

search:load

Description

The load directive sends an id or a query to the twigkit API and stores the result on a scoped variable and in the *Response Service*

Usage

as element:

```
<search:load var="{string}" platform="{string}" id="{string}" query="{string}"
[error-page="{string}"]> </search:load>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var	string	The name of the variable to store the document in.
platform	string	The name of the platform to retrieve the document from.
id	string	The ID of the document to retrieve. If both an ID and query are present ID will take precedence.
query	string	The name of the query object. If both an ID and query are present ID will take precedence.
error-page (optional)	string	Error page to redirect to when Platform throws an exception.

search:message-list

Description

Output Messages and QuerySuggestions such as spellchecking or synonyms.

Custom messages within a message-list **must be wrapped in another html tag!** – see *Example Custom*

Usage

as element:

```
<search:message-list response="{string}" [type="{string}"] [title="{string}"]  
[first="{number}"] [last="{number}"]> </search:message-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	The name of the response containing the messages to display.
type (optional)	string	The type of Message to display. Default is to include all messages irrespective of type.
title (optional)	string	Add a title to appear with this element.
first (optional)	number	First message to output from this result set (1 based).
last (optional)	number	Last message to output from this result set.

Example

Source

```
<h3 id="example_all-messages">All messages</h3> <tk-example> <search:message-list
response="response" title="My Messages"></search:message-list> </tk-example> <tk-
code-wrapper> <code hljs language="html"> <search:message-list
response="response" title="My Fuzzy Results"></search:message-list> </code> </tk-
code-wrapper> <hr /> <h3 id="example_only-messages-of-selected-type">Only
messages of selected type</h3> <tk-example> <search:message-list
response="response" type="fuzzy-results" title="My Fuzzy Results">
</search:message-list> </tk-example> <tk-code-wrapper> <code hljs
language="html"> <search:message-list response="response" type="fuzzy-results"
title="My Fuzzy Results"></search:message-list> </code> </tk-code-wrapper> <hr />
<h3 id="example_first-and-last-messages">First and Last Messages</h3> <tk-
example> <search:message-list response="response" first="1" last="2" title="My
Messages"></search:message-list> </tk-example> <tk-code-wrapper> <code hljs
language="html"> <search:message-list response="response" first="1" last="2"
title="My Messages"></search:message-list> </code> </tk-code-wrapper> <hr /> <h3
id="example_custom">Custom</h3> <tk-example> <search:message-list
response="response" title="My Fuzzy Results"> <p><strong>{{message.type}}
</strong> - <span ng-bind-html="message.value"></span> <small>({{message.count}})
</small></p> </search:message-list> </tk-example> <tk-code-wrapper> <code hljs
language="html"> <search:message-list response="response" title="My Fuzzy
Results"> <p> <strong>{{message.type}}</strong> - <span ng-bind-
html="message.value"></span> <small>({{message.count}})</small> </p>
</search:message-list> </code> </tk-code-wrapper>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope', 'ResponseService', function($scope, ResponseService) { $scope.response =
{ "query": { "filters": [], }, "hits": { "act": [ "Long", 0 ] }, "page": 2,
"results": [], "facets": {}, "facts": {}, "time": 19, "messages": [ { "type":
"fuzzy-results", "value": "<em>Lorem</em> ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.", "count": 20 }, { "type": "not-fuzzy-results", "value": "Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur.", "count": 100 }, { "type": "fuzzy-results", "value": "Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.", "count": 0 } ] }
ResponseService.setResponse('response', $scope.response); }]);
```

Result

All messages

EXAMPLE

My Messages

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

```
<search:message-list response="response" title="My Fuzzy Results"></search:message-list>
```

Only messages of selected type

EXAMPLE

My Fuzzy Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

```
<search:message-list response="response" type="fuzzy-results" title="My Fuzzy Results">
</search:message-list>
```

First and Last Messages

EXAMPLE

My Messages

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
<search:message-list response="response" first="1" last="2" title="My Messages">
</search:message-list>
```

Custom

EXAMPLE

My Fuzzy Results

fuzzy-results - *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.* (20)

not-fuzzy-results - Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. (100)

fuzzy-results - Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. (0)

```
<search:message-list response="response" title="My Fuzzy Results">
  <p>
```

```
        <strong>{{message.type}}</strong> - <span ng-bind-html="message.value"></span> <small>
    ({{message.count}})</small>
    </p>
</search:message-list>
```

search:no-results

Description

The no results directive will display a message when no results are found on the response.

This tag should not be put in the `<search:result-list></search:result-list>`

Usage

as element:

```
<search:no-results response="{string}"> </search:no-results>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	The response object containing the results

Example

Source

```
<h5 id="example_source_default-message">Default Message</h5> <search:result-list>
</search:result-list> <search:no-results response="response"></search:no-results>
<hr /> <h5 id="example_source_custom-message">Custom Message</h5> <search:result-
list></search:result-list> <search:no-results response="response"> <h4
id="example_no-results-found">No Results Found!</h4> <p>Try searching for
something else?</p> </search:no-results>
```

```
angular.module('lightning').controller('ExampleController', ['$scope',
'ResponseService', function($scope, ResponseService) { var response = {};
ResponseService.setResponse('response', response); }]);
```

Result

Default Message

Sorry, there are no results that match your search criteria.

If you've checked your spelling, try using more general keywords.

Custom Message

No Results Found!

Try searching for something else?

search:offcanvas-toggle

Description

Creates a toggle button to show/hide the side panel. Adds class to target element when directive is clicked.

Usage

as element:

```
<search:offcanvas-toggle target="{string}" [active-class="{string}"]>
</search:offcanvas-toggle>
```

Parameters

Param	Type	Details
target	string	The element to add the active class to
active-class (optional)	string	The active class, defaults to 'active'

Example

Source

```
<layout:grid> <layout:block tiny="1-4"> <layout:box> <search:offcanvas-toggle
target="#left-sidebar"></search:offcanvas-toggle> </layout:box> </layout:block>
<layout:block tiny="1-2"> <!-- this toggle will not be shown as no element with
the ID no-sidebar exists --> <search:offcanvas-toggle target="#no-sidebar">
</search:offcanvas-toggle> </layout:block> <layout:block tiny="1-4"> <layout:box>
<search:offcanvas-toggle target="#right-sidebar" class="right">
</search:offcanvas-toggle> </layout:box> </layout:block> </layout:grid>
<layout:grid styling="row-offcanvas"> <layout:block medium="1-4" drawer="left"
id="left-sidebar"> <layout:box class="tk-stl-bg-red"> <layout:sidebar> <h4
id="example_left-sidebar-content">Left Sidebar Content</h4> </layout:sidebar>
</layout:box> </layout:block> <layout:block medium="1-2"> <layout:box> <p
styling="text-align:center;">Shrink your browser window to toggle sidebars</p>
</layout:box> </layout:block> <layout:block medium="1-4" drawer="right"
id="right-sidebar"> <layout:box class="tk-stl-bg-blue"> <layout:sidebar> <h4
id="example_right-sidebar-content">Right Sidebar Content</h4> </layout:sidebar>
</layout:box> </layout:block> </layout:grid>
```

Result

Left Sidebar Content

Shrink your browser window to toggle sidebars

Right Sidebar Content

search:pagination

Description

The pagination directive.

Usage

as element:

```
<search:pagination response="{string}" [max-pages="{number}"] [page-limit="{number}"] [first-label="{string}"] [show-last="{boolean}"] [scroll-to="{string|boolean}"]> </search:pagination>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	The name of the response to use to generate the pagination.
max-pages (optional)	number	Maximum number of page links to output (default is 10).
page-limit (optional)	number	Maximum limit for which a pagination link is created (e.g. FAST ESP has a default offset limit of ~4000).
first-label (optional)	string	Label for the link to the first page. Default: '1'
show-last (optional)	boolean	Whether to show a link to the last page. Default: false
scroll-to (optional)	string link: [boolean]	The ID of an element to scroll to when the users selects a page. By default it will scroll to the top. You can disable this automatic scrolling by setting this attribute to <code>false</code>

Example

Source

```
<h3 id="example_normal">Normal</h3> <search:pagination response="response" max-  
pages="3"></search:pagination> <hr /> <h3 styling="text-center"  
id="example_center">Center (pagination-wrapper-center)</h3> <search:pagination  
response="response" max-pages="3" styling="pagination-wrapper-center">  
</search:pagination> <hr /> <h3 style="text-align:right;"  
id="example_right">Right (pagination-wrapper-right)</h3> <search:pagination  
response="response" max-pages="3" styling="pagination-wrapper-right">  
</search:pagination>
```

```
angular.module('lightning').controller('ExampleController', ['$scope',  
'ResponseService', function($scope, ResponseService) { var response = {query:  
{rpp:10},hits:{act:[0,2255]},page:1};  
ResponseService.setResponse('response',response); }]);
```

Result

Normal

1 2 3 next

Center (pagination-wrapper-center)

1 2 3 next

Right (pagination-wrapper-right)

1 2 3 next

search:platform

Description

The platform directive creates an instance of platform for a given host and configuration.

Usage

as element:

```
<search:platform [var="{string}"] [host="{string}"] conf="{string}">
</search:platform>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var (optional)	string	The variable to assign the platform instance to.
host (optional)	string	The host and port name to use (default is to not define it).
conf	string	Platform configuration to use.

search:query

Description

The query directive constructs a query string and stores it in a scoped variable, *Query*. *Filter's* can be added to add or remove filters from the query. You can also use the *Query Custom Tag* to add custom parameters to the query.

Usage

as element:

```
<search:query var="{string}" parameters="{string}" [from="{string}"] [query="{string}"] [modified-query="{string}"] [type="{string}"] [results-per-page="{string}"] [max-results="{string}"] [page="{string}"] [facets="{string}"] [fields="{string}"] [sorts="{string}"] [view="{string}"] [filters="{string}"] [collection="{string}"] [custom="{string}"]> </search:query>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var	string	The variable to assign the query to
parameters	string	The URL parameters to query with.
from (optional)	string	Name of a previously defined query to extend
query (optional)	string	Set the search query value.
modified-query (optional)	string	Specify an alternative query which will be submitted to the search engine in place of the query parameter supplied by the user (but can include user submitted parameter). To the user it will appear as the query parameter had been used, but the modifiedQuery will be submitted to the search engine.
type (optional)	string	Specifies how to interpret multiple query terms. Takes one of three values: any (boolean OR the terms), all (boolean OR/AND the terms), or adv (interpret using search platform-specific advanced query language).
results-per-page (optional)	string	The number of results to display per page.
max-results (optional)	string	The maximum number of results to evaluate. Defaults to -1 which means no upper limit.

Param	Type	Details
page (optional)	string	Set the page of results to display. Default value is 1
facets (optional)	string	Specifies which facets to evaluate and return.
fields (optional)	string	A comma-separated list of fields which should be returned in the Response.
sorts (optional)	string	Specifies fields and order for sorting. Sort order is specified by one of two characters: + for ascending and – for descending. To sort ascending by the field price, the sort would be specified as sorts="+price".
view (optional)	string	Set which search engine view to use. Behaviour will vary by installation.
filters (optional)	string	Specifies filters to apply, specified as an ampersand (&) separated String
collection (optional)	string	Specifies the collection you want to use.
custom (optional)	string	Specifies custom parameters to apply, specified as an ampersand (&) separated String

Example

Source

```
<search:query var="query" filters='zipcode["75216"] ["75216"],bar["foo"] ["foo"] '
max-results="1"> <query:filter field="foo" value="bar"></query:filter>
<query:custom name="foo" value="bar"></query:custom> </search:query>
<search:query var="copyQuery" from="query"></search:query> <div ng-if="query !=
null"> <code>Original - {{query | setQueryParam:'q':'foo'}}</code> </div>
<code>Copy - {{copyQuery}}</code>
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout', function($scope,$timeout) { }]); angular
.module('lightning') .filter('setQueryParameter', setQueryParam);
setQueryParameter.$inject = ['$timeout']; function setQueryParam($timeout) {
return function (query, parameter, value) { if (query && parameter && value) {
query[parameter] = value; return query; } else { return query } }; }
```

Result

```
Original - {"max":1,"filters":[{"exclude":false,"field":"zipcode","val":{"act":"75216","dsp":"75216\""},bar[\"f
Copy - {"max":1,\"filters":[{"exclude":false,\"field\":\"zipcode\", \"val\":{\"act\":\"75216\", \"dsp\":\"75216\"\"},bar[\"foo\"
```

search:response

Description

The response directive sends a query to the twigkit API and stores the response on a scoped variable and in the *Response Service*

Inside of the tag you can access the average response time (`$parent.averageResponseTime`) and the previous response time (`$parent.previousResponseTime`).

Usage

as element:

```
<search:response var="{string}" platform="{object}" query="{object}" [error-  
page="{string}"] [on="{string}"]> </search:response>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var	string	The name of the variable to store the response in
platform	object	The platform to query against
query	object	A query object
error- page (optional)	string	Error page to redirect to when Platform throws an exception.
on (optional)	string	Event that is broadcasted for when the response should reload. Default: reloadSearch.

search:response-statistics

Description

The Response Statistics directive.

Usage

as element:

```
<search:response-statistics response="{object}" [number-format="{string}"]>
</search:response-statistics>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object
number-format (optional)	string	Format string for the numbers used using the Java DecimalFormat pattern. Default: ',0.'

Example

Source

```
<search:response-statistics response="response"></search:response-statistics> <hr
/> <h6 id="example_source_edit-values">Edit Values</h6> <label>Enter Page Number:
<input ng-model='response.page' ng-change="reloadResponse()" /></label><br>
<label>Enter Results Per Page <input ng-model='response.query.rpp' ng-
change="reloadResponse()" /></label><br> <label>Enter Actual Number of Results
<input ng-model='response.hits.act[1]' ng-change="reloadResponse()" /></label>
<br>
```



```
angular.module('lightning') .controller('ExampleController', ['$scope',  
'ResponseService', function($scope, ResponseService) { $scope.response = { hits:{  
act:[264,264] }, page:1, query:{ rpp:10 } }  
ResponseService.setResponse('response',$scope.response); $scope.reloadResponse =  
function(){ var response = angular.copy($scope.response); $scope.response = {};  
$scope.response = response;  
ResponseService.setResponse('response',$scope.response); } }]);
```

Result

Showing 1 - 10 of 264

Edit Values

Enter Page Number:

Enter Results Per Page

Enter Actual Number of Results

search:result

Description

The Result directive.

Usage

as element:

```
<search:result [result="{object}"] [when="{string}"] [equals="{string}"]>
</search:result>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
result (optional)	object	A result object
when (optional)	string	The result will only be rendered if this field exists
equals (optional)	string	The result will only be rendered if this is the value of the when field

search:result-list

Description

The Result List directive.

Accessing result index

Within a search result you can access the index of the result by doing `{{$parent.index}}`. The index is zero-based so for result 1 the index will be 0.

Usage

as element:

```
<search:result-list response="{object}" results="{array}" [infinite-scroll="{string}"] [query="{string}"] [platform="{string}"] [load-more="{boolean}"]>
</search:result-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object (Use response or results not both)
results	array	A list of results (Use response or results not both)
infinite-scroll (optional)	string	Whether infinite scrolling should be enabled (query and platform is required for this functionality to be enabled). Valid values: • false Disables infinite scrolling. • (default) • true Enables infinite-scrolling in window mode (see below). • window When the window is scrolled to the bottom of the result list new results will be loaded. • parent When the parent element is scrollable when this element is scrolled to the bottom new results will load.
query (optional)	string	The name of the query object used for infinite scrolling or load more.
platform (optional)	string	The name of the platform object used for infinite scrolling or load more.
load-more (optional)	boolean	Whether to have a clickable link to load more results.

Example

Source

```
<layout:grid> <layout:block> <h3 id="example_accessing-the-index-of-a-  
result">Accessing the index of a result ($parent.index)</h3> </layout:block>  
<search:result-list response="response" styling="cards-sm-1 cards-md-2">  
<search:result> <layout:block> <div class="tk-stl-card"> <search:field  
name="name" styling="title" urlfield="url"></search:field> <div class="field tk-  
rgrid-g tk-stl-label-left"> <p class="tk-rgrid-u-1 tk-rgrid-u-md-1-3 tk-stl-  
label">Index</p> <div class="tk-rgrid-u-1 tk-rgrid-u-md-2-3"> <p><span  
class="value">{{$parent.index}}</span></p> </div> </div> <search:field name="age"  
styling="label-left" label="Age"></search:field> <search:field name="date"  
styling="label-left" label="Created Date" date-format="dd/MM/yy"></search:field>  
<search:field name="blank" styling="label-left" label="Contact Number" default-  
value="No Contact Number on File."></search:field> <search:field name="number"  
styling="label-left" label="Worth" number-format="f#,###.##"></search:field>  
<search:field name="madeup" styling="label-left" label="Test"></search:field>  
<search:field name="longstring" styling="label-left" label="Long String" max-  
characters="40" show-more></search:field> </div> </layout:block> </search:result>  
</search:result-list> </layout:grid> <layout:grid> <layout:block> </layout:grid>
```

```
angular.module('lightning') .controller('ExampleController',  
 ['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)  
{ $scope.response = { page: 2, query: { rpp: 50 }, hits:{ act:['long','1234'] },  
facets: { }, results: [ { result: { fields: { name: { name: 'name', val:  
['Testing Tester'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: {  
name: 'age', val: ['Long', 35] }, date: { name: 'date', val: ['2014-11-  
11T00:00:00'] }, blank:{ name:'blank', val:null, }, number:{ name:'number', val:  
[123456789] }, longstring:{ name:'longstring', val:['Lorem ipsum dolor sit amet,  
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore  
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris  
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit  
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint  
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim  
id est laborum.' ] } }, }, }, { result: { fields: { name: { name: 'name', val:  
['Foo (Bar)'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: { name:  
'age', val: ['Long', 26] }, date: { name: 'date', val: ['2015-11-11T00:00:00'] },  
blank:{ name:'blank', val:null, }, number:{ name:'number', val:[-123456789] } },  
, }, ], } $scope.results = $scope.response.results;  
ResponseService.setResponse('response', $scope.response); }]);
```

Result

Accessing the index of a result (\$parent.index)

Testing Tester

Index	0
Age	Long 35
Created Date	11/11/14
Contact Number	No Contact Number on File.
Worth	£123,456,789
Long String	Lorem ipsum dolor sit amet, consectetur ... show more

Foo (Bar)

Index	1
Age	Long 26
Created Date	11/11/15
Contact Number	No Contact Number on File.
Worth	-£123,456,789

search:results-per-page

Description

The Results Per Page Directive allows you to change the number of documents that are returned from the platform. The results per page tag should be wrapped in a *Layout Controls Tag*.

Usage

as element:

```
<search:results-per-page response="{object}" [title="{string}"] [options="{string}"]> </search:results-per-page>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object
title (optional)	string	The title to display before the result count options.
options (optional)	string	Which options to provide the user. Should be a comma-separated list of integers.

Example

Source

```
<layout:controls> <search:results-per-page response="response"  
options="10,20,25,50"></search:results-per-page> </layout:controls>
```

```
angular.module('lightning') .controller('ExampleController', ['$scope',  
function($scope) { $scope.response = { } }]);
```

Result

Per page ▼

search:sort

Description

The sort directive for selecting how the data on the page should be sorted. The sort tag should be wrapped in a *Layout Controls Tag*.

Usage

as element:

```
<search:sort response="{string}" fields="{string}" [default-label="{string}"]  
[title="{string}"] [default-order="{string}"] [mode="{string}"]> </search:sort>
```

Directive info

- This directive creates new scope.

Parameters

Par am	T y p e	Details
res pon se	s t r i n g	The name of the response
fie lds	s t r i n g	Fields to provide sorting on (comma-separated).
def aul t- lab el (op tio nal)	s t r i n g	Label to display for the 'default' sort option when no specific sorts are applied. Usually implies sorting on score or relevance. Default: 'relevance'
tit le (op tio nal)	s t r i n g	Add a title to display. Default: 'Sort by'
def aul t- ord er (op tio nal)	s t r i n g	Attribute to set the default sort order of Sort Fields (asc or desc). Default: 'asc'
mod e (op tio nal)	s t r i n g	The mode in which the sorting should work, be default click. click: Selecting a sort will sort in the default order, when clicking on a selected sort this sort will be removed, to change the order use the ascending and descending options at the bottom of the dropdown. toggle: Selecting a sort will sort in the default order, when clicking on the selected sort again it will reverse the order.

Example

Source

```
<layout:grid> <layout:block md="1-1"> <h5 id="example_source_click-mode">Click
Mode</h5> <layout:controls> <search:sort response="response"
fields="IncidentNum=Incident Number,Foo=Bar"></search:sort> </layout:controls>
</layout:block> <layout:grid> <layout:block md="1-1"> <h5
id="example_source_toggle-mode">Toggle Mode</h5> <layout:controls> <search:sort
response="response" fields="IncidentNum=Incident Number,Foo=Bar" mode="toggle"
class="left"></search:sort> </layout:controls> </layout:block> </layout:grid>
```

```
angular.module('lightning').controller('ExampleController', ['$scope',
'ResponseService', function($scope, ResponseService) { var response = {};
ResponseService.setResponse('response', response); }]);
```

Result

Click Mode

Sort by relevance ▼

Toggle Mode

Sort by relevance ▼ | Incident Number | Bar

search:tabs

Description

The Tabs Directive

Usage

as element:

```
<search:tabs response="{object}" tabs="{string}" [field-name="{string}"] [all-label="{string}"] [platform="{string}"] [query="{string}"] [count-number="{string}"]> </search:tabs>
```

Directive info

- This directive creates new scope.

Parameters

Parameter	Type	Details
response	object	A response object
tabs	string	The tabs to output. Specified as a comma separated list. When using filter tabs, you can either specify the actual filter name, or the actual filter name with an alternative display name. To substitute actual filter names with alternative display names use the following notation: 'programme=Programmes'. Multiple filters can be grouped under a single tab by separating the actual filter names with a character, e.g.: 'Videos Podcasts=Multimedia'.
field-name (optional)	string	The field to apply filters to when in 'filter' mode.
all-label (optional)	string	Label for the option to turn off all filters. If this attribute is specified, an additional link will be displayed to show all Results (e.g. not applying a Filter).
platform (optional)	string	The Platform to use when submitting the facet without any filters applied. This is used to to get the filter counts for the original query after a user clicks on a filter (e.g. drills down).

Parameter	Type	Details
query (optional)	string	The name of the query object used to get the count values.
count-number (optional)	string	Specify how the number representing the count should be formatted. Optional values are 'plain', 'formatted', 'rounded' and 'none'.

Events

tab_changed

When the tab is changed on this directive an event called `tab_changed` will be fired with the name of the tab that is now active.

This event can be listened to like so.

```
$scope.$on('tab_changed', function(e, tab){
    console.log('New Tab = ' + tab);
});
```

Type:

broadcast

Target:

root scope

search:typeahead

Description

Creates a typeahead style search box that can be customized to allow the user to add filters using the *search keyword tag*. You can also include the *instant result list tag* to allow the user to select a result based on the search query and filters.

Usage

as element:

```
<search:typeahead platform="{string}" [auto-search="{string}"] [action="{string}"]> </search:typeahead>
```

Parameters

Param	Type	Details
platform	string	The name of the platform to use to search for matching keyword values
auto-search (optional)	string	Milliseconds to wait after the search is automatically submitted
action (optional)	string	The page to send the search query to

Example

Source

```
<search:typeahead platform="platform" datepicker-min-view="year"
action="laureates" class="searchbox"> <search:keyword keyword="country"
facet="Country of affiliation" description="Country of affiliation">
</search:keyword> <search:keyword keyword="gender" facet="gender"
description="laureate's gender"></search:keyword> <search:keyword
keyword="category" facet="Prize Category" description="Award category">
</search:keyword> <search:keyword keyword="before" facet="year" date="before"
description="Awarded before this year" date-format="yyyy" date-display-
format="yyyy"></search:keyword> <search:keyword keyword="after" facet="year"
date="after" description="Awarded after this year" date-format="yyyy" date-
display-format="yyyy"></search:keyword> <search:keyword keyword="year"
facet="year" date="on" description="Awarded in this year" date-format="yyyy"
date-display-format="yyyy"></search:keyword> <instant:result-list
platform="platform"> <search:result> <search:field name="full_name" label=""
styling="label-inline" url="/#/laureates/{{result.fields.id.val | actualValue}}">
</search:field> <search:field name="Prize Category" label="" styling="label-
inline" class="pill"></search:field> </search:result> </instant:result-list>
</search:typeahead>
```

Result

security:user-actions

Description

User actions tag to log in, log out and navigate home.

To enable this you need to make sure that your security.conf file contains the line `webservice-enabled: true`

Your security.conf file can be found in `resources/conf/security/security.conf`.

For spring security you will need the following rule to allow access to the user api, if parts of the site are public.

```
<intercept-url pattern="/twigkit/api/user" access="isAnonymous() or isAuthenticated()"/>
```

Usage

as element:

```
<security:user-actions> </security:user-actions>
```

Directive info

- This directive creates new scope.

Example

Source

```
<security:user-actions></security:user-actions>
```

Result



series:facet

Description

To render information in facets use the Series Facet tag, has to be nested within a *Chart Display* tag.

You can use the `response` attribute to load the data from the response or alternatively you can add a `platform` and this will get only the data the series needs to be rendered. You will also need to include the `query` directly on this tag if you wish for the user to be able to filter using the data of this series or alternatively add the `query` to the *Chart Display* tag.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see here.

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

Usage

as element:

```
<series:facet response="{object}" platform="{string}" name="{string}" type="{string}" [title="{string}"] [show-marker="{string}"] [number-to-show="{number}"] [action="{string}"] [query="{object}"] [date-format="{string}"] [measure="{string}"] [selected-by-default="{boolean}"] [select="{string}"]> </series:facet>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object (please include either a response or a platform not both!).
platform	string	The name of the platform to get the data from (please include either a response or a platform not both!).
name	string	Name of facet.
type	string	Chart type
title (optional)	string	The title of the series that appears in the legend and tooltip.
show-marker (optional)	string	Whether to show marker or not.
number-to-show (optional)	number	Number of filters to render on the graph.
action (optional)	string	The page/action to which clicking the chart should be submitted. By default the action points to the current page.
query (optional)	object	The search query from which to build links.
date-format (optional)	string	How to format the date. See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.
measure (optional)	string	What fact to use as an alternative to count as quantitative value for a filter.
selected-by-default (optional)	boolean	Whether this series should be shown when the <code>chart:display</code> is using <code>select-series="true"</code> mode. Default: true.

Param	Type	Details
select (optional)	string	How the data should be filtered when clicking on a part of the chart, a valid query must be present. Valid values: • false User selects a filter by clicking the value; (default) • multiselect Supports selecting multiple filters that get combined with an 'OR'. • multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.

Example

Source

```
<chart:display response="response" title="Files Indexed" subtitle="Per Day"
type="bar"> <series:facet name="test1" title="First Pass"></series:facet>
<series:facet name="test2" title="Second Pass"></series:facet> </chart:display>
<hr /> <chart:display response="response" title="Files Indexed" subtitle="Per
Day"> <series:facet name="test1" type="column" title="First Pass"></series:facet>
<series:facet name="test2" type="column" title="Second Pass"></series:facet>
</chart:display> <hr /> <div> <h3 id="example_number-to-show-attribute">Number To
Show Attribute</h3> <chart:display response="response" title="Population"
type="line"> <series:facet name="population" title="Current Population" number-
to-show="5"></series:facet> <series:facet name="population2000" title="Population
in 2000" number-to-show="5"></series:facet> </chart:display> </div> <div> <h3
id="example_show-marker-attribute">Show Marker Attribute</h3> <chart:display
response="response" title="Population" type="line"> <series:facet
name="population" title="Current Population" show-marker="false"></series:facet>
<series:facet name="population2000" title="Population in 2000" show-
marker="false"></series:facet> </chart:display> </div> <div id="colour-demo"> <h3
id="example_colour">Colour (since 0.3.0)</h3> <p>Add an ID to the element
wrapping your chart and configure the background colour and series colours like
so.</p> <pre> #colour-demo .highcharts-color-0{ fill:pink; stroke:pink; }
#colour-demo .highcharts-color-1{ fill:yellow; stroke:yellow; } #colour-demo
.highcharts-background{ fill:#f5f0f0; } </pre> <p>For more detail on what you can
change and set using CSS see <a href="http://www.highcharts.com/docs/chart-
design-and-style/style-by-css" target="_blank">here.</a></p> <chart:display
response="response" title="Population" type="column"> <series:facet
name="population" title="Current Population"></series:facet> <series:facet
name="population2000" title="Population in 2000"></series:facet> </chart:display>
```

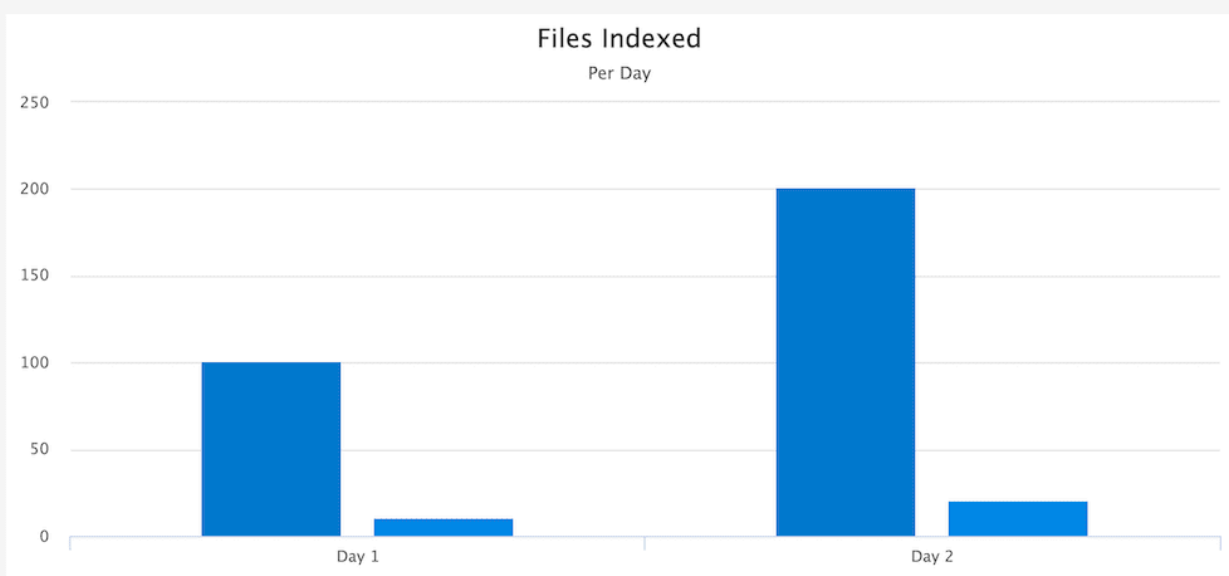
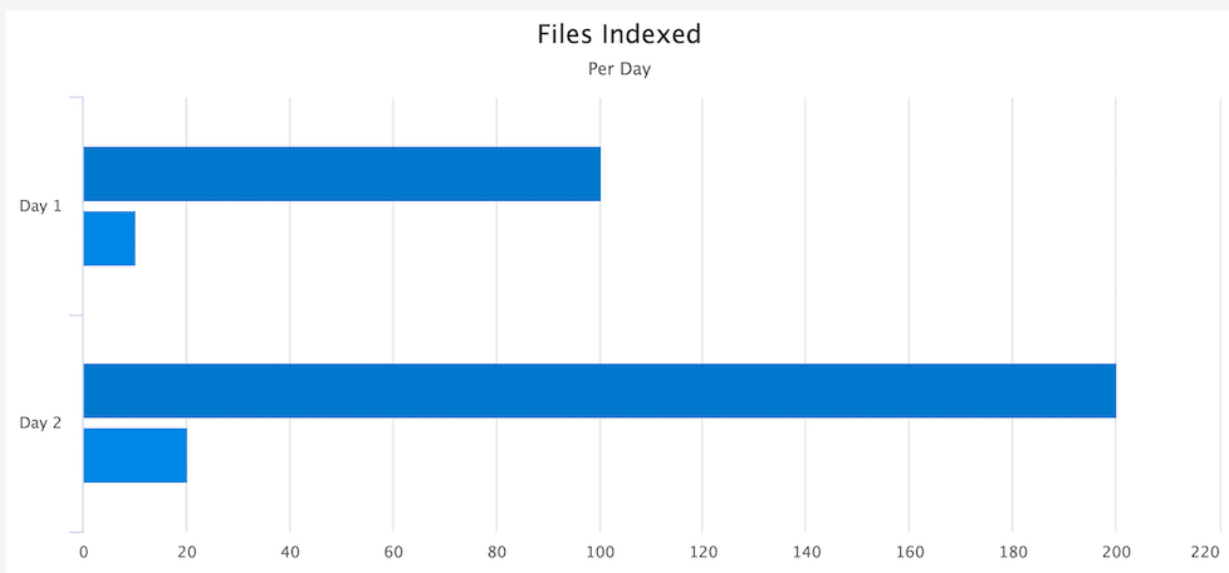
```
#colour-demo .highcharts-color-0 { fill: pink; stroke: pink; colour-demo
.highcharts-color-1 { fill: yellow; stroke: yellow; colour-demo .highcharts-
background { fill: #f5f0f0;
```

```

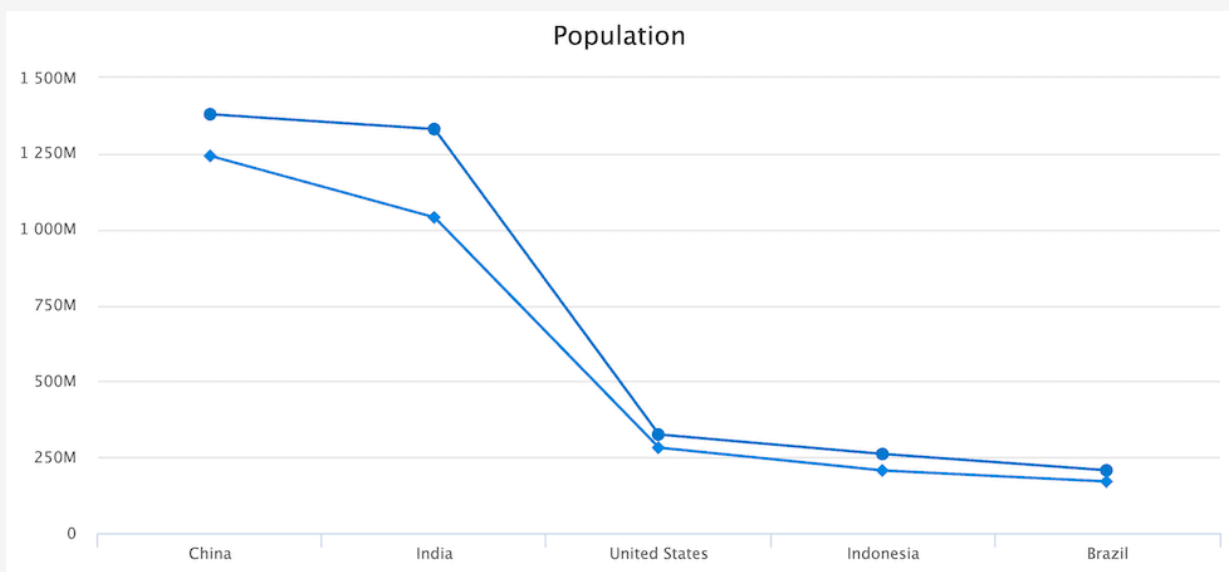
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,
ResponseService) { $scope.response = { page: 2, query: { rpp: 50 }, facets: {
test1: { filters: [ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 100, }, { val:
{ dsp: 'Day 2', act: 'Day 2', }, count: 200, } ] }, test2: { filters: [ { val: {
dsp: 'Day 1', act: 'Day 1', }, count: 10, }, { val: { dsp: 'Day 2', act: 'Day 2',
}, count: 20, } ] }, population:{ filters: [ { val: 'China', count: 1379790000,
}, { val: 'India', count: 1330780000, }, { val: 'United States', count:
324897000, }, { val: 'Indonesia', count: 260581000, }, { val: 'Brazil', count:
206927000, }, { val: 'Pakistan', count: 194772000, }, { val: 'Nigeria', count:
186987000, }, { val: 'Bangladesh', count: 161401000, }, { val: 'Russia', count:
146727405, }, { val: 'Japan', count: 126693000, }, ] }, population2000:{ filters:
[ { val: 'China', count: 1242612300, }, { val: 'India', count: 1040000000, }, {
val: 'United States', count: 281421923, }, { val: 'Indonesia', count: 206264595,
}, { val: 'Brazil', count: 170000000, }, { val: 'Pakistan', count: 140000000, },
{ val: 'Nigeria', count: 110000000, }, { val: 'Bangladesh', count: 130000000, },
{ val: 'Russia', count: 147000000, }, { val: 'Japan', count: 127000000, }, ] },
}, results: [] } ResponseService.setResponse('response', $scope.response); }]);

```

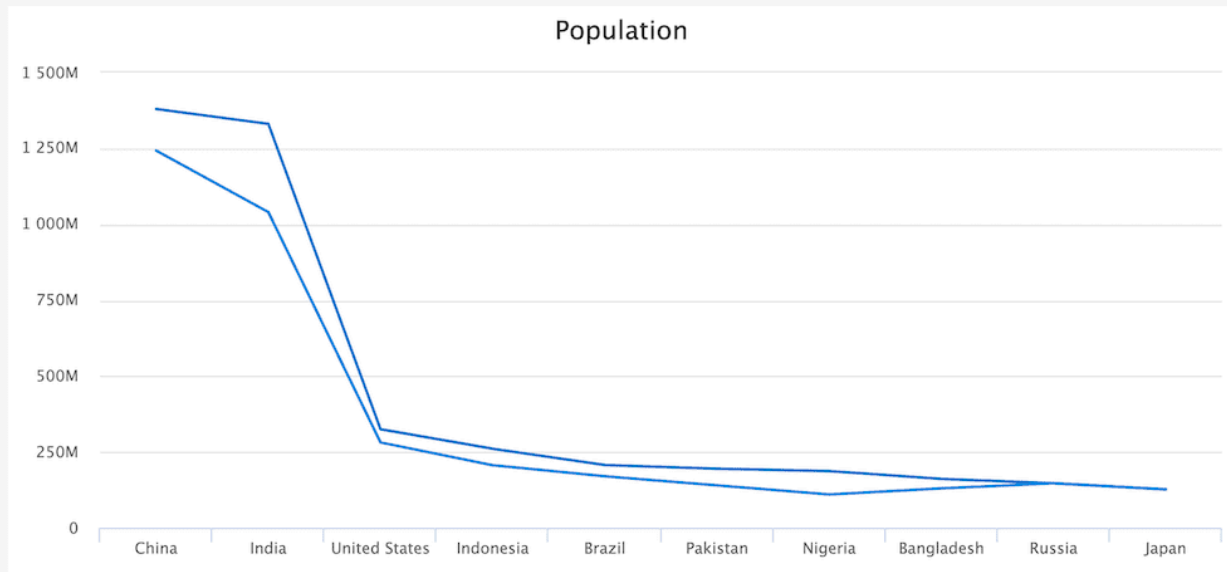
Result



Number To Show Attribute



Show Marker Attribute

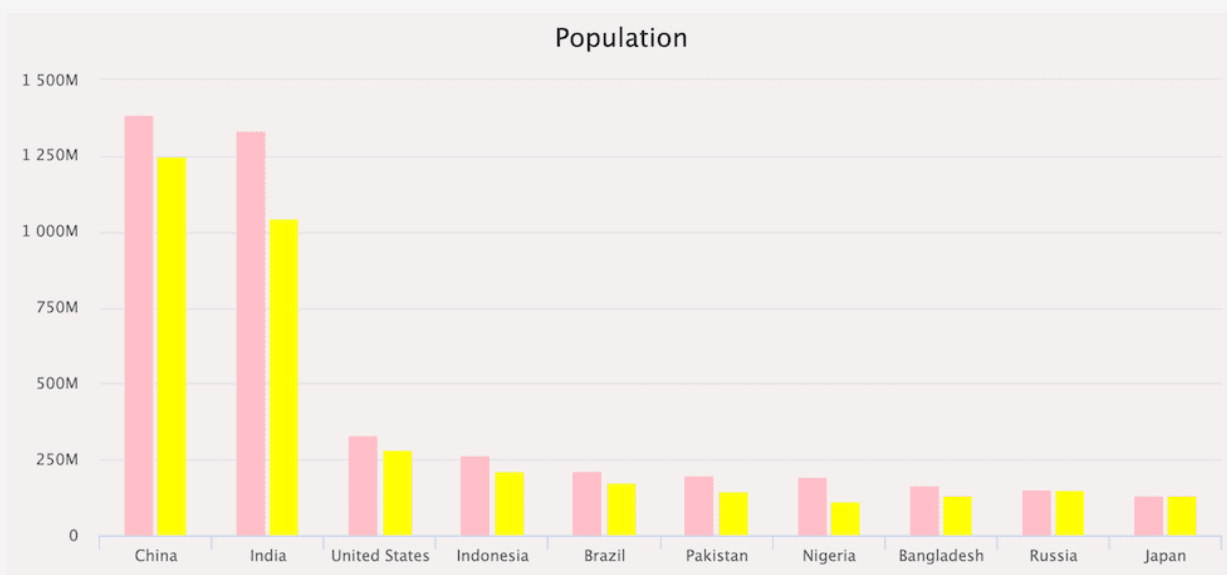


Colour (since 0.3.0)

Add an ID to the element wrapping your chart and configure the background colour and series colours like so.

```
#colour-demo .highcharts-color-0{
  fill:pink;
  stroke:pink;
}
#colour-demo .highcharts-color-1{
  fill:yellow;
  stroke:yellow;
}
#colour-demo .highcharts-background{
  fill:#f5f0f0;
}
```

For more detail on what you can change and set using CSS see [here](#).



series:pivot

Description

To render information in a pivot facet use the Series Pivot tag, this tag has to be nested within a *Chart Display tag*. Please note it will only render the second level of relevant filters. For more details see the data object used in the examples below.

You can use the `response` attribute to load the data from the response. You will also need to include the `query` directly on this tag if you wish for the user to be able to filter using the data of this series or alternatively add the `query` to the *Chart Display tag*.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see here.

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

When rendering pivot facet data it is recommended that you always include the *Chart Legend tag* within your *Chart Display tag* as the examples below show.

Usage

as element:

```
<series:pivot response="{string}" name="{string}" type="{string}" [show-marker="{string}"] [action="{string}"] [query="{object}"] [select="{string}"]>
</series:pivot>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	string	The name of the response to get the facet from.
name	string	Name of facet.
type	string	Chart type
show-marker (optional)	string	Whether to show marker or not.
action (optional)	string	The page/action to which clicking the chart should be submitted. By default the action points to the current page.
query (optional)	object	The search query from which to build links.
select (optional)	string	How the data should be filtered when clicking on a part of the chart, a valid query must be present. Valid values: • false User selects a filter by clicking the value; (default) • multiselect Supports selecting multiple filters that get combined with an 'OR'. • multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.

series:result-list

Description

To render information using fields from a result list you can use the Series Result List tag and use the label-field and value-field attributes. This tag must be nested in the *Chart Display* tag.

You can use the `response` attribute to load the data from the response or alternatively you can add a platform and this will get only the data the series needs to be rendered. You will also need to include the `query` directly on this tag if you wish for the user to be able to filter using the data of this series or alternatively add the `query` to the *Chart Display* tag.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see [here](http://www.highcharts.com/docs/chart-design-and-style/style-by-css).

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

Usage

as element:

```
<series:result-list response="{object}" results="{object}" platform="{string}"
label-field="{string|expression}" value-field="{string|expression}" [class-
field="{string|expression}"] type="{string}" [title="{string|expression}"] [show-
marker="{string}"] [url-field="{string}"] [number-to-show="{number}"] [action="{
string}"] [query="{object}"] [selected-by-default="{boolean}"] [select="{
string}"]> </series:result-list>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object (please include either a response, results, or a platform not multiple!).
results	object	An array of results (please include either a response, results, or a platform not multiple!).
platform	string	The name of the platform to get the data from (please include either a response, results, or a platform not multiple!).
label-field	string link : [expression]	Name of field containing the label.
value-field	string link : [expression]	Name of field containing the value.
class-field (optional)	string link : [expression]	Name of the field containing the value you wish to add as a class to the point
type	string	The type of chart to render. Types available - 'pie', 'line', 'area', 'column', 'bar', or 'scatter'.
title (optional)	string link : [expression]	The title of the series that appears in the legend and tooltip.
show-marker (optional)	string	Whether to show marker or not.
url-field (optional)	string	The field in the result list containing the URL from which to make the visualisation clickable.
number-to-show (optional)	number	Number of results to render on the graph.
action (optional)	string	The page/action to which clicking the chart should be submitted. By default the action points to the current page.

Param	Type	Details
query (optional)	object	The search query from which to build links.
selected-by-default (optional)	boolean	Whether this series should be shown when the chart:display is using select-series="true" mode. Default: true.
select (optional)	string	How the data should be filtered when clicking on a part of the chart, a valid query must be present. Valid values: • false User selects a filter by clicking the value; (default) • multiselect Supports selecting multiple filters that get combined with an 'OR'. • multiselect-instant Supports selecting multiple filters that get combined with an 'OR', filters will be applied automatically.

Example

Source

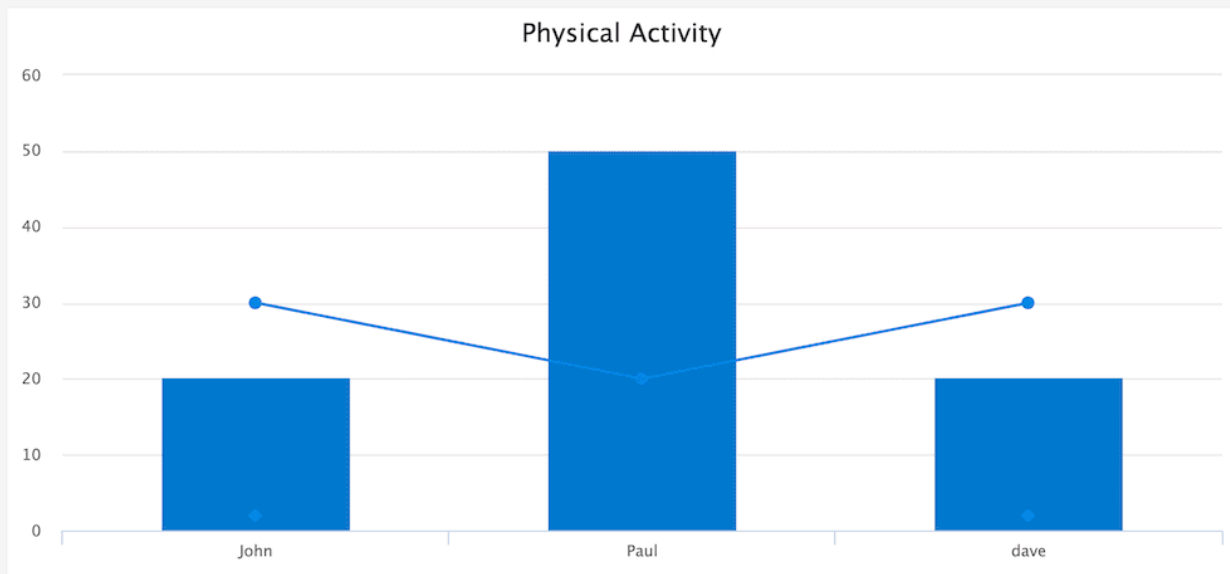
```
<h3 id="example_tooltip-tag">Tooltip Tag</h3> <chart:display response="response"
title="Physical Activity"> <series:result-list response="response" label-
field="name" value-field="runs" type="column"></series:result-list>
<series:result-list response="response" label-field="name" value-field="rides"
type="line"></series:result-list> <series:result-list response="response" label-
field="name" value-field="swims" type="area"></series:result-list>
</chart:display> <div id="colour-demo"> <h3 id="example_colour">Colour (since
0.3.0)</h3> <p>Add an ID to the element wrapping your chart and configure the
background colour and series colours like so.</p> <pre> #colour-demo .highcharts-
color-0{ fill:pink; stroke:pink; } #colour-demo .highcharts-color-1{ fill:yellow;
stroke:yellow; } #colour-demo .highcharts-color-2{ fill:purple; stroke:purple; }
#colour-demo .highcharts-background{ fill:#f5f0f0; } </pre> <p>For more detail on
what you can change and set using CSS see <a
href="http://www.highcharts.com/docs/chart-design-and-style/style-by-css"
target="_blank">here.</a></p> <chart:display response="response" title="Physical
Activity"> <series:result-list response="response" label-field="name" value-
field="runs" type="column"></series:result-list> <series:result-list
response="response" label-field="name" value-field="rides" type="line">
</series:result-list> <series:result-list response="response" label-field="name"
value-field="swims" type="area"></series:result-list> </chart:display> </div>
<div id="class-field"> <h3 id="example_class-field">Class Field</h3> <p>Add an ID
to the element wrapping your chart and configure the point based on the content
of the class field you provide.</p> <pre> #class-field .athlete{ fill:teal;
stroke:teal; } #class-field .iron-man{ fill:red; stroke:red; } #class-field
.cyclist{ fill:blue; stroke:blue; } </pre> <p>For more detail on what you can
change and set using CSS see <a href="http://www.highcharts.com/docs/chart-
design-and-style/style-by-css" target="_blank">here.</a></p> <chart:display
response="response" title="Physical Activity"> <series:result-list
response="response" label-field="name" value-field="runs" type="column" class-
field="type"></series:result-list> </chart:display>
```

```
#colour-demo .highcharts-color-0 { fill: pink; stroke: pink; colour-demo
.highcharts-color-1 { fill: yellow; stroke: yellow; colour-demo .highcharts-
color-2 { fill: purple; stroke: purple; colour-demo .highcharts-background {
fill: #f5f0f0; class-field .athlete { fill: teal; stroke: teal; class-field
.iron-man { fill: red; stroke: red; class-field .cyclist { fill: blue; stroke:
blue;
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { }, results: [ {
result: { fields: { name:{ val:'John', }, runs: { val:20, }, rides:{ val:30, },
swims:{ val:2, }, type:{ val:'cyclist' } } }, { result: { fields: { name: {
val:'Paul', }, runs: { val:50, }, rides:{ val:20, }, type:{ val:'athlete' } } },
hits: { act: ['Hits', 264] } }, { result: { fields: { name:{ val:'dave', }, runs:
{ val:20, }, rides:{ val:30, }, swims:{ val:2, }, type:{ val:'iron man' } } },
] } ResponseService.setResponse('response',$scope.response); }]);
```

Result

Tooltip Tag

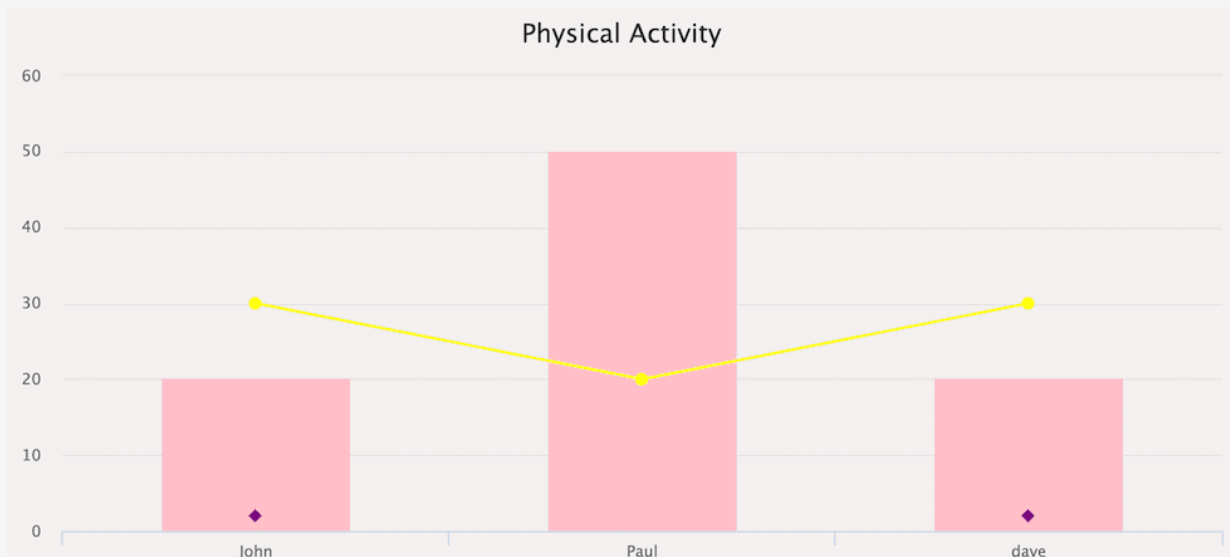


Colour (since 0.3.0)

Add an ID to the element wrapping your chart and configure the background colour and series colours like so.

```
#colour-demo .highcharts-color-0{
  fill:pink;
  stroke:pink;
}
#colour-demo .highcharts-color-1{
  fill:yellow;
  stroke:yellow;
}
#colour-demo .highcharts-color-2{
  fill:purple;
  stroke:purple;
}
#colour-demo .highcharts-background{
  fill:#f5f0f0;
}
```

For more detail on what you can change and set using CSS see [here](#).

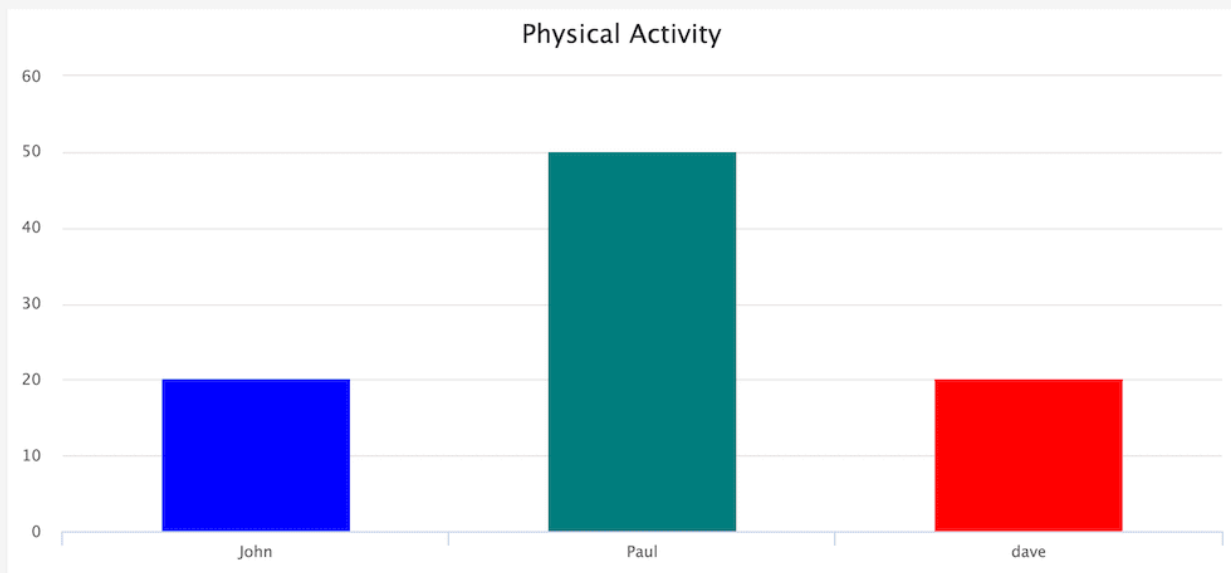


Class Field

Add an ID to the element wrapping your chart and configure the point based on the content of the class field you provide.

```
#class-field .athlete{  
  fill:teal;  
  stroke:teal;  
}  
#class-field .iron-man{  
  fill:red;  
  stroke:red;  
}  
#class-field .cyclist{  
  fill:blue;  
  stroke:blue;  
}
```

For more detail on what you can change and set using CSS see [here](#).



series:time

Description

The series time directive shows a time series on a chart, in order to use this you must set type to 'time' on the *chart display tag*.

For detail on how to change the color and background color see the demo below, for more detail on what you can change and set using CSS see [here](http://www.highcharts.com/docs/chart-design-and-style/style-by-css).

(<http://www.highcharts.com/docs/chart-design-and-style/style-by-css>)

Usage

as element:

```
<series:time response="{object}" platform="{string}" name="{string}" value-  
field="{string|expression}" label-field="{string|expression}" [title="  
{string|expression}"] [show-marker="{string}"] [type="{string}"] [range-  
selector="{boolean}"] [navigator="{boolean}"] [query="{object}"] [measure="  
{string}"] [sort="{boolean}"] [number-to-show="{Number}"] [selected-by-default="  
{boolean}"]> </series:time>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
response	object	A response object (please include either a response or a platform not both!).
platform	string	The name of the platform to get the data from (please include either a response or a platform not both!).
name	string	Name of facet. (Use either name or value-field and label-field not both!)
value-field	string link: [expression	Field from the result containing the count. (Use either name or value-field and label-field not both!)
label-field	string link: [expression	Field from the result containing the label. (Use either name or value-field and label-field not both!)
title (optional)	string link: [expression	The title of the series that appears in the legend and tooltip.
show-marker (optional)	string	Whether to show marker or not.
type (optional)	string	Type of chart to render.
range-selector (optional)	boolean	Where the input boxes should be enabled (from and to) Default:true
navigator (optional)	boolean	Whether to enable the navigator beneath the chart. Default:true
query (optional)	object	The search query from which to build links.
measure (optional)	string	What fact to use as an alternative to count as quantitative value for a filter.
sort (optional)	boolean	Whether the data should be sort by the directive. Defaults to true.
number-to-show (optional)	Number	The number of points to show on the graph. Applies only when using the platform attribute. (Default:25)

Param	Type	Details
selected-by-default (optional)	boolean	Whether this series should be shown when the chart:display is using select-series="true" mode. Default: true.

show:when

Description

Show or hide element based on expression passed to it.

Usage

as attribute

```
<ANY show-when="{expression}"> ... </ANY>
```

Parameters

Param	Type	Details
show-when	expression	Expression to evaluate.

Example

Source

```
<p><a href="javascript:void(0);" ng-click="alert()" styling="button">{{foo ?  
'Hide' : 'Show'}}!</a> </p> <h3 show-when="foo" id="example_hello-world">Hello  
World</h3>
```

```
angular.module('lightning') .controller('ExampleController',  
 ['$scope', '$timeout', function($scope,$timeout) { $scope.foo = false;  
 $scope.alert = function(){ $scope.foo = !$scope.foo }; }]);
```

Result

Show!

slide:toggle

Description

The toggle panes directive

Usage

as attribute

```
<ANY slide-toggle target="{string}"> ... </ANY>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
target	string	CSS selector to get slideable element

Example

Source

```
<button class="tk-stl-button" widget-slide-toggle="#foo1">Expand 1</button> <div id="foo1" class="widget-slide-down"> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.</p> <br> <p padding-bottom="2">Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.</p> </div>
```

Result

Expand 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.

Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.

social:comment-form

Deprecated API

Description

Social Comment Form directive has been deprecated, please see *Collaborate Comment Form tag*

Usage

as element:

```
<social:comment-form> </social:comment-form>
```

social:comment-list

Deprecated API

Description

Social Comment List directive has been deprecated, please see *Collaborate Comment List tag*

Usage

as element:

```
<social:comment-list> </social:comment-list>
```

styling

Description

Extends the HTML style attribute to apply twigkit classes to the element as specified in the style attribute

Usage

as attribute

```
<ANY styling="{string}"> ... </ANY>
```

Parameters

Param	Type	Details
styling	string	Space separated list of styles to apply to the element

table:body

Description

Creates a table body `<tbody>`

Usage

as element:

```
<table:body [infinite-scrolling="{boolean}"] [query="{string}"] [platform="{object}"]> </table:body>
```

Parameters

Param	Type	Details
infinite-scrolling (optional)	boolean	Whether infiniteScrolling should be enabled. (Not supported when using results instead of response in table:result-list)
query (optional)	string	The name of the query used for infinite scrolling or load more. (Not supported when using results instead of response in table:result-list)
platform (optional)	object	The platform object used for infinite scrolling or load more. (Not supported when using results instead of response in table:result-list)

table:cell

Description

Creates a table cell `<td>`

Usage

as element:

```
<table:cell name="{string}" [default-value="{string}"] [separator="{string}"]  
[date-format="{string}"] [number-format="{string}"] [show="{Number}"] [prefix="  
{string}"] [suffix="{string}"] [url="{string}"] [url-field="{string}"] [target="  
{string}"] [action="{string}"] [query="{string}"]> </table:cell>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
name	string	Name of the field to render
default-value (optional)	string	Default value to show when no value is present in the index.
separator (optional)	string	Separator character or sequence to split a Field into multiple values. Setting the separator to ";" would turn the value "company1; company2; company3" into a list of 3 values.
date-format (optional)	string	How to format the date. See <code>#/api/lightning.filter:dateFormat[date format filter]</code> for valid formats.
number-format (optional)	string	Format string for the numbers used using the Java DecimalFormat pattern. Default: ',0.'
show (optional)	Number	Number of values to show if the field is multivalued. Default: No limit (-1)
prefix (optional)	string	Text to prepend to index values (not applied to default value). E.g. if the index field 'money' contains the value '10'. A prefix="£" would result in '£10' being displayed as the value.
suffix (optional)	string	Text to append to index values (not applied to default value). E.g. if the index field 'weight' contains the value '10'. A suffix=" kg." would result in '10 kg.' being displayed as the value.
url (optional)	string	URL to redirect to when the field is clicked;

Param	Type	Details
url-field (optional)	string	Field containing a URL to redirect to when the field is clicked;
target (optional)	string	Set an HTML target attribute for the anchor tag. For example use _blank to have the generated link open the results in a new window. (Not applicable to query)
action (optional)	string	Alternative page to submit the query parameters to – defaults to current page. If a 'url' attribute is provided the action is ignored and query parameters are not included in the request.
query (optional)	string	When set, the value of the named field will be displayed as a hyperlink that submits the query again with this field and its value as a search constraint.

table:header

Description

Creates a table header `<thead>`

Usage

as element:

```
<table:header> </table:header>
```

table:header-cell

Description

Creates a table header cell `<th>`

Usage

as element:

```
<table:header-cell [label="{string}"] [name="{string}"] [sort="{boolean}"]>
</table:header-cell>
```

Parameters

Param	Type	Details
label (optional)	string	Label for the header.
name (optional)	string	Name of the field to sort.
sort (optional)	boolean	Whether to enable sorting.

table:result-list

Description

Creates a table

The following tags can be nested within the table result list tag.

- *Table Header*
- *Table Header Cell*
- *Table Body*
- *Table Row*
- *Table Cell*

Usage

as element:

```
<table:result-list response="{string}" results="{array}"> </table:result-list>
```

Parameters

Param	Type	Details
response	string	The name of the response to populate the table with. (Use results or response not both.)
results	array	A list of results to be rendered in the table. (Use results or response not both.)

Example

Source

```
<h3 id="example_table">Table</h3> <table:result-list response="response">
<table:header> <table:header-cell sort="true">Name</table:header-cell>
<table:header-cell label="Urk"></table:header-cell> <table:header-cell
name="Date"></table:header-cell> <table:header-cell name="Number"></table:header-
cell> <table:header-cell name="multi"></table:header-cell> <table:header-cell
name="Prefix"></table:header-cell> <table:header-cell name="Suffix">
</table:header-cell> <table:header-cell name="Empty"></table:header-cell>
</table:header> <table:body> <table:row> <table:cell name="name"></table:cell>
<table:cell name="url" url-field="url" target="_blank"></table:cell> <table:cell
name="date" date-format="fullDate"></table:cell> <table:cell name="number"
number-format="$#,###.00"></table:cell> <table:cell name="multi" show="2">
</table:cell> <table:cell name="prefix" prefix="f"></table:cell> <table:cell
name="suffix" suffix=" kg." url="https://en.wikipedia.org/wiki/Kilogram">
</table:cell> <table:cell name="empty" default-value="No Value"></table:cell>
</table:row> </table:body> </table:result-list> <hr /> <h3
id="example_table">Table (with results)</h3> <table:result-list
results="response.results"> <table:header> <table:header-cell
sort="true">Name</table:header-cell> <table:header-cell label="Urk">
</table:header-cell> <table:header-cell name="Date"></table:header-cell>
<table:header-cell name="Number"></table:header-cell> <table:header-cell
name="multi"></table:header-cell> <table:header-cell name="Prefix">
</table:header-cell> <table:header-cell name="Suffix"></table:header-cell>
<table:header-cell name="Empty"></table:header-cell> </table:header> <table:body>
<table:row> <table:cell name="name"></table:cell> <table:cell name="url" url-
field="url" target="_blank"></table:cell> <table:cell name="date" date-
format="fullDate"></table:cell> <table:cell name="number" number-
format="$#,###.00"></table:cell> <table:cell name="multi" show="2"></table:cell>
<table:cell name="prefix" prefix="f"></table:cell> <table:cell name="suffix"
suffix=" kg." url="https://en.wikipedia.org/wiki/Kilogram"></table:cell>
<table:cell name="empty" default-value="No Value"></table:cell> </table:row>
</table:body> </table:result-list> <hr /> <h3 id="example_vertical-
comparison">Vertical Comparison (class="tbl-compare")</h3> <table:result-list
response="response" class="tbl-compare"> <table:header> <table:header-cell
sort="true">Name</table:header-cell> <table:header-cell label="Urk">
</table:header-cell> <table:header-cell name="Date"></table:header-cell>
<table:header-cell name="Number"></table:header-cell> <table:header-cell
name="multi"></table:header-cell> <table:header-cell name="Prefix">
</table:header-cell> <table:header-cell name="Suffix"></table:header-cell>
<table:header-cell name="Empty"></table:header-cell> </table:header> <table:body>
<table:row> <table:cell name="name"></table:cell> <table:cell name="url" url-
field="url" target="_blank"></table:cell> <table:cell name="date" date-
format="fullDate"></table:cell> <table:cell name="number" number-
format="$#,###.00"></table:cell> <table:cell name="multi" show="2"></table:cell>
<table:cell name="prefix" prefix="f"></table:cell> <table:cell name="suffix"
suffix=" kg." url="https://en.wikipedia.org/wiki/Kilogram"></table:cell>
<table:cell name="empty" default-value="No Value"></table:cell> </table:row>
</table:body> </table:result-list> <hr /> <h3 id="example_stacked">Stacked
(class="tbl-stacked")</h3> <table:result-list response="response" class="tbl-
stacked"> <table:header> <table:header-cell sort="true">Name</table:header-cell>
<table:header-cell label="Urk"></table:header-cell> <table:header-cell
name="Date"></table:header-cell> <table:header-cell name="Number"></table:header-
cell> <table:header-cell name="multi"></table:header-cell> <table:header-cell
name="Prefix"></table:header-cell> <table:header-cell name="Suffix">
</table:header-cell> <table:header-cell name="Empty"></table:header-cell>
</table:header> <table:body> <table:row> <table:cell name="name"></table:cell>
<table:cell name="url" url-field="url" target="_blank"></table:cell> <table:cell
name="date" date-format="fullDate"></table:cell> <table:cell name="number"
number-format="$#,###.00"></table:cell> <table:cell name="multi" show="2">
</table:cell> <table:cell name="prefix" prefix="f"></table:cell> <table:cell
name="suffix" suffix=" kg." url="https://en.wikipedia.org/wiki/Kilogram">
```

```

</table:cell> <table:cell name="empty" default-value="No Value"></table:cell>
</table:row> </table:body> </table:result-list> <hr /> <h3 id="example_compare-
combination">Compare Combination (class="tbl-combo-compare")</h3> <table:result-
list response="response" class="tbl-combo-compare"> <table:header> <table:header-
cell sort="true">Name</table:header-cell> <table:header-cell label="Urk">
</table:header-cell> <table:header-cell name="Date"></table:header-cell>
<table:header-cell name="Number"></table:header-cell> <table:header-cell
name="multi"></table:header-cell> <table:header-cell name="Prefix">
</table:header-cell> <table:header-cell name="Suffix"></table:header-cell>
<table:header-cell name="Empty"></table:header-cell> </table:header> <table:body>
<table:row> <table:cell name="name"></table:cell> <table:cell name="url" url-
field="url" target="_blank"></table:cell> <table:cell name="date" date-
format="fullDate"></table:cell> <table:cell name="number" number-
format="$#,###.00"></table:cell> <table:cell name="multi" show="2"></table:cell>
<table:cell name="prefix" prefix="f"></table:cell> <table:cell name="suffix"
suffix=" kg." url="https://en.wikipedia.org/wiki/Kilogram"></table:cell>
<table:cell name="empty" default-value="No Value"></table:cell> </table:row>
</table:body> </table:result-list>

```

```

angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, hits:{ act:['long','1234'] },
facets: { }, results: [ { result: { fields: { name: {name: 'name', val:
['Testing Tester'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: {
name: 'age', val: ['Long', 35] }, date: { name: 'date', val: ['2014-11-
11T00:00:00'] }, blank:{ name:'blank', val:null, }, number:{ name:'number', val:
[123456789] }, multi:{ name:'multi', val:['Multi','Value','Field'] }, prefix:{
name:'prefix', val:['10'] }, suffix:{ name:'suffix', val:['10'] }, empty:{
name:'empty', val:[] } }, }, }, { result: { fields: { name: { name: 'name', val:
['Foo Bar'] }, url: { name: 'url', val: ['http://twigkit.com'] }, age: { name:
'age', val: ['Long', 26] }, date: { name: 'date', val: ['2015-11-11T00:00:00'] },
blank:{ name:'blank', val:null, }, number:{ name:'number', val:[-123456789] },
multi:{ name:'multi', val:['Multi','Value','Field'] }, prefix:{ name:'prefix',
val:['-10'] }, suffix:{ name:'suffix', val:['-10'] }, empty:{ name:'empty',
val:null } }, }, }, ], } ResponseService.setResponse('response',
$scope.response); }]);

```

Result

Table

Name	Urk	Date	Number	multi	Prefix	Suffix	Empty
Testing Tester	http://twigkit.com	Tuesday, November 11, 2014	\$123,456,789.00	Multi Value	£10	10 kg.	No Value
Foo Bar	http://twigkit.com	Wednesday, November 11, 2015	-\$123,456,789.00	Multi Value	£-10	-10 kg.	No Value

Table (with results)

Name	Urk	Date	Number	multi	Prefix	Suffix	Empty
Testing Tester	http://twigkit.com	Tuesday, November 11, 2014	\$123,456,789.00	Multi Value	£10	10 kg.	No Value
Foo Bar	http://twigkit.com	Wednesday, November 11, 2015	-\$123,456,789.00	Multi Value	£-10	-10 kg.	No Value

Vertical Comparison (class="tbl-compare")

Name	Urk	Date	Number	multi	Prefix	Suffix	Empty
Testing Tester	http://twigkit.com	Tuesday, November 11, 2014	\$123,456,789.00	Multi Value	£10	10 kg.	No Value
Foo Bar	http://twigkit.com	Wednesday, November 11, 2015	-\$123,456,789.00	Multi Value	£-10	-10 kg.	No Value

Stacked (class="tbl-stacked")

Name	Urk	Date	Number	multi	Prefix	Suffix	Empty
Testing Tester	http://twigkit.com	Tuesday, November 11, 2014	\$123,456,789.00	Multi Value	£10	10 kg.	No Value
Foo Bar	http://twigkit.com	Wednesday, November 11, 2015	-\$123,456,789.00	Multi Value	£-10	-10 kg.	No Value

Compare Combination (class="tbl-combo-compare")

Name	Testing Tester	Foo Bar
Urk	http://twigkit.com	http://twigkit.com
Date	Tuesday, November 11, 2014	Wednesday, November 11, 2015
Number	\$123,456,789.00	-\$123,456,789.00
multi	Multi Value	Multi Value
Prefix	£10	£-10
Suffix	10 kg.	-10 kg.
Empty	No Value	No Value

table:row

Description

Creates a table row `<tr>`

Usage

as element:

```
<table:row> </table:row>
```

task:form

Description

Posts form data to create a Twigkit Task. Any form fields with a name prefixed with `attr-` will be added as attributes to the created task.

State classes of `submitting`, `success`, and `error` are added to the form element to allow the UI to be customised.

Usage

as element:

```
<task:form task="{string}" [redirect="{string}"]> </task:form>
```

Parameters

Param	Type	Details
task	string	The task name. For example, if a configuration <code>logger.conf</code> is found under <code>resources/conf/task/</code> , the task name would be <code>task.logger</code>
redirect (optional)	string	If set, the form will redirect to this URL on successful submission

task:list

Description

Displays the current user's task list and regularly checks for updates. Any HTML within the tag will have access to a `task` variable for the current task being iterated over in the list. As well as the `stop`, `start`, and `delete` methods for interacting with the task.

Usage

as element:

```
<task:list [empty-text="{string}"] [interval="{Number}"] [date-format="{string}"]  
[show-expiry="{string}"]> </task:list>
```

Parameters

Param	Type	Details
emptyText (optional)	string	The message to show if the user has no tasks. Default: "You have no tasks."
interval (optional)	Number	How often statues updates should be requested in milliseconds. Default 1000. Set to false to disable.
date-format (optional)	string	Date format to display all dates in. Default 'medium'.
show-expiry (optional)	string	Date format to display all dates in. Default 'medium'.

toggle:controls

Description

The toggle controls directive

Usage

as element:

```
<toggle:controls labels="{string}" panes="{string}" [title="{string}"]>
</toggle:controls>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
labels	string	Labels that describe each corresponding pane. There must be exactly the same number of labels as there are panes.
panes	string	This parameter associates the controls with the corresponding panes by referencing the panes' id parameter.
title (optional)	string	The title of the pane controls.

Example

Source

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="panes"></toggle:controls>
<toggle:panes id="panes"> <toggle:pane> <h3 id="example_pane-1">Pane 1</h3>
</toggle:pane> <toggle:pane> <h3 id="example_pane-2">Pane 2</h3> </toggle:pane>
<toggle:pane> <h3 id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes>
```

Result

Pane 1 Pane 2 Pane 3

toggle:pane

Description

The toggle pane directive.

Usage

as element:

```
<toggle:pane [visible="{boolean}"]> </toggle:pane>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
visible (optional)	boolean	Used to denote if the pane should be shown by default.

Example

Source

```
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="panes"></toggle:controls>
<toggle:panes id="panes"> <toggle:pane> <h3 id="example_pane-1">Pane 1</h3>
</toggle:pane> <toggle:pane> <h3 id="example_pane-2">Pane 2</h3> </toggle:pane>
<toggle:pane> <h3 id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes>
```

Result

Pane 1 Pane 2 Pane 3

toggle:panes

Description

The toggle panes directive

Usage

as element:

```
<toggle:panes id="{string}" [allow-multiple="{boolean}"]> </toggle:panes>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
id	string	Id of the element
allow-multiple (optional)	boolean	Whether or not multiple widgets should be shown at once. Default: false.

Example

Source

```
<p>Default tabs</p> <toggle:controls labels="Pane 1, Pane 2, Pane 3"
panes="panes"></toggle:controls> <toggle:panes id="panes"> <toggle:pane> <h3
id="example_pane-1">Pane 1</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes> <br> <p>Line</p>
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="line" styling="line">
</toggle:controls> <toggle:panes id="line"> <toggle:pane> <h3 id="example_pane-
1">Pane 1</h3> </toggle:pane> <toggle:pane> <h3 id="example_pane-2">Pane 2</h3>
</toggle:pane> <toggle:pane> <h3 id="example_pane-3">Pane 3</h3> </toggle:pane>
</toggle:panes> <br> <p>Pills</p> <toggle:controls labels="Pane 1, Pane 2, Pane
3" panes="pills" styling="pill"></toggle:controls> <toggle:panes id="pills">
<toggle:pane> <h3 id="example_pane-1">Pane 1</h3> </toggle:pane> <toggle:pane>
<h3 id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes> <br> <p>Simple</p>
<toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="simple" styling="simple">
</toggle:controls> <toggle:panes id="simple"> <toggle:pane> <h3 id="example_pane-
1">Pane 1</h3> </toggle:pane> <toggle:pane> <h3 id="example_pane-2">Pane 2</h3>
</toggle:pane> <toggle:pane> <h3 id="example_pane-3">Pane 3</h3> </toggle:pane>
</toggle:panes> <p>Tabs Bordered</p> <toggle:controls labels="Pane 1, Pane 2,
Pane 3" panes="bordered" styling="bordered"></toggle:controls> <toggle:panes
id="bordered"> <toggle:pane> <h3 id="example_pane-1">Pane 1</h3> </toggle:pane>
<toggle:pane> <h3 id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane>
<h3 id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes> <br> <p>Line
Bordered</p> <toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="line-b"
styling="line bordered"></toggle:controls> <toggle:panes id="line-b">
<toggle:pane> <h3 id="example_pane-1">Pane 1</h3> </toggle:pane> <toggle:pane>
<h3 id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes> <br> <p>Pills
Bordered</p> <toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="pills-b"
styling="pill bordered"></toggle:controls> <toggle:panes id="pills-b">
<toggle:pane> <h3 id="example_pane-1">Pane 1</h3> </toggle:pane> <toggle:pane>
<h3 id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes> <br> <p>Simple
Bordered</p> <toggle:controls labels="Pane 1, Pane 2, Pane 3" panes="simple-b"
styling="simple bordered"></toggle:controls> <toggle:panes id="simple-b">
<toggle:pane> <h3 id="example_pane-1">Pane 1</h3> </toggle:pane> <toggle:pane>
<h3 id="example_pane-2">Pane 2</h3> </toggle:pane> <toggle:pane> <h3
id="example_pane-3">Pane 3</h3> </toggle:pane> </toggle:panes>
```

Result

Default tabs

Pane 1 Pane 2 Pane 3

Line

Pane 1 Pane 2 Pane 3

Pills

Pane 1 Pane 2 Pane 3

Simple

Pane 1 Pane 2 Pane 3

Tabs Bordered

Pane 1 Pane 2 Pane 3

Line Bordered

Pane 1 Pane 2 Pane 3

Pills Bordered

Pane 1 Pane 2 Pane 3

Simple Bordered

Pane 1 Pane 2 Pane 3

tooltip

Description

Creates a tooltip.

Usage

as attribute

```
<ANY tooltip="{string}"> ... </ANY>
```

Parameters

Param	Type	Details
tooltip	string	The text to display in the tooltip

Example

Source

```
<layout:grid styling="textcenter"> <layout:block medium="1-1"> <h3
id="example_postion">Postion</h3> </layout:block> <layout:block medium="1-3"> <h3
tooltip="Top Left" styling="tooltip-top-left" id="example_top-left">Top-Left</h3>
</layout:block> <layout:block medium="1-3"> <h3 tooltip="Top!" styling="tooltip-
top" id="example_top">Top</h3> </layout:block> <layout:block medium="1-3"> <h3
tooltip="Top Right" styling="tooltip-top-right" id="example_top-right">Top-
Right</h3> </layout:block> </layout:grid> <layout:grid styling="textcenter">
<layout:block medium="1-3"> <h3 tooltip="Left" styling="tooltip-left"
id="example_left">Left</h3> </layout:block> <layout:block medium="1-3"> <h3
id="example">.</h3> </layout:block> <layout:block medium="1-3"> <h3
tooltip="Right" styling="tooltip-right" id="example_right">Right</h3>
</layout:block> </layout:grid> <layout:grid styling="textcenter"> <layout:block
medium="1-3"> <h3 tooltip="Bottom Left" styling="tooltip-bottom-left"
id="example_bottom-left">Bottom-Left</h3> </layout:block> <layout:block
medium="1-3"> <h3 tooltip="Bottom!" styling="tooltip-bottom"
id="example_bottom">Bottom</h3> </layout:block> <layout:block medium="1-3"> <h3
tooltip="Bottom Right" styling="tooltip-bottom-right" id="example_bottom-
right">Bottom-Right</h3> </layout:block> </layout:grid> <hr /> <layout:grid
styling="textcenter"> <layout:block medium="1-1"> <h3
id="example_sizes">Sizes</h3> </layout:block> <layout:block medium="1-3"> <button
tooltip="You can show very very long sentences inside tooltips by using various
available size variation classes." styling="button tooltip-top tooltip-
small">Small</button> </layout:block> <layout:block medium="1-3"> <button
tooltip="You can show very very long sentences inside tooltips by using various
available size variation classes." styling="button tooltip-top tooltip-
medium">Medium</button> </layout:block> <layout:block medium="1-3"> <button
tooltip="You can show very very long sentences inside tooltips by using various
available size variation classes." styling="button tooltip-top tooltip-
large">Large</button> </layout:block> </layout:grid> <hr /> <layout:grid
styling="textcenter"> <layout:block medium="1-1"> <h3
id="example_colours">Colours</h3> </layout:block> <layout:block medium="1-4">
<button tooltip="This is an error tooltip" styling="button bg-red tooltip-top
tooltip-error">Red</button> </layout:block> <layout:block medium="1-4"> <button
tooltip="This is a warning tooltip" styling="button bg-yellow tooltip-top
tooltip-warning">Yellow</button> </layout:block> <layout:block medium="1-4">
<button tooltip="This is an info tooltip" styling="button tooltip-top tooltip-
info">Blue</button> </layout:block> <layout:block medium="1-4"> <button
tooltip="This is a success tooltip" styling="button button-green tooltip-top
tooltip-success">Green</button> </layout:block> </layout:grid>
```


Result

Postion		
Top-Left	Top	Top-Right
Left	.	Right
Bottom-Left	Bottom	Bottom-Right

Sizes		
Small	Medium	Large

Colours			
Red	Yellow	Blue	Green

track:clicks

Description

Usage

as element:

```
<track:clicks [query="{query}"] [type="{string}"] [response="{string}"] [fields="{string|expression}"] [any_attribute="{string}"]> </track:clicks>
```

Parameters

Param	Type	Details
query (optional)	query	The name of the query associated with the click.
type (optional)	string	Type of click event, (Default: link or result if click event happens in a search:result)
response (optional)	string	The name of the response to get the fact called <code>tracking</code> from so that this information can be sent to the activity endpoint. If a response is not provided and the click happens in a <i>Search Result List</i> we will get the name of the response from the result-list if applicable.
fields (optional)	string]link: [expression	If the click happens within a result send the following fields (comma separated) to the endpoint. (e.g - <code>name,title,url</code>)
any_attribute (optional)	string	You can add any custom attribute here such as <code>foo="bar"</code> and this data will be sent to the endpoint. The attribute must not start with <code>ng-</code> and also cannot be <code>url,label,type,query,offset,pos,fields,id,response</code> or <code>class</code> .

Example

Source

```
<layout:grid> <layout:block md="1-1"> <strong>To see the data being sent to the
endpoint click one of the links below with the network tab open, close the window
that opens and view the request which fails with a 405 in the network tab of your
console </strong> <h3 id="example_wrapped-around-a-field">Wrapped Around a
Field</h3> <search:result-list response="response" styling="cards-sm-1 cards-md-
2"> <search:result> <layout:block> <div class="tk-stl-card"> <search:field
name="name" styling="title" urlfield="url"></search:field> <search:field
name="age" styling="label-left" label="Age"></search:field> <search:field
name="date" styling="label-left" label="Created Date" date-format="dd/MM/yy">
</search:field> <track:clicks> <search:field name="blank" styling="label-left"
label="Contact Number" default-value="No Contact Number on File."
url="http://twigkit.com" target="_blank"></search:field> </track:clicks>
<track:clicks> <search:field name="number" styling="label-left" label="Worth"
number-format="f#,###.##" url="http://twigkit.com" target="_blank">
</search:field> </track:clicks> <search:field name="madeup" styling="label-left"
label="Test"></search:field> </div> </layout:block> </search:result>
</search:result-list> <h3 id="example_wrapped-around-a-result-list">Wrapped
Around a Result List</h3> <track:clicks type="foo"> <search:result-list
response="response" styling="cards-sm-1 cards-md-2"> <search:result>
<layout:block> <div class="tk-stl-card"> <collaborate:bookmark>
</collaborate:bookmark> <search:field name="name" styling="title" urlfield="url">
</search:field> <search:field name="age" styling="label-left" label="Age">
</search:field> <search:field name="date" styling="label-left" label="Created
Date" date-format="dd/MM/yy"></search:field> <search:field name="blank"
styling="label-left" label="Contact Number" default-value="No Contact Number on
File." url="http://twigkit.com" target="_blank"></search:field> <search:field
name="number" styling="label-left" label="Worth" number-format="f#,###.##"
url="http://twigkit.com" target="_blank"></search:field> <search:field
name="madeup" styling="label-left" label="Test"></search:field> </div>
</layout:block> </search:result> </search:result-list> </track:clicks>
</layout:block> </layout:grid>
```

```
angular.module('lightning').controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, hits:{ act:['long','1234'] },
facets: { }, results: [ { result: { id:"123456abcdefg", fields: { name: { name:
'name', val: ['Testing Tester'] }, url: { name: 'url', val:
['http://twigkit.com'] }, age: { name: 'age', val: ['Long', 35] }, date: { name:
'date', val: ['2014-11-11T00:00:00'] }, blank:{ name:'blank', val:null, },
number:{ name:'number', val:[123456789] }, }, }, ],{ result: {
id:"abcdefg123456", fields: { name: { name: 'name', val: ['Foo (Bar)'] }, url: {
name: 'url', val: ['http://twigkit.com'] }, age: { name: 'age', val: ['Long', 26]
}, date: { name: 'date', val: ['2015-11-11T00:00:00'] }, blank:{ name:'blank',
val:null, }, number:{ name:'number', val:[-123456789] } }, }, ], }
$scope.results = $scope.response.results; ResponseService.setResponse('response',
$scope.response); }]);
```

Result

To see the data being sent to the endpoint click one of the links below with the network tab open, close the window that opens and view the request which fails with a 405 in the network tab of your console

Wrapped Around a Field

Testing Tester

Age

Long

35

Created Date

11/11/14

Contact Number

No Contact Number on File.

Worth

£123,456,789

Foo (Bar)

Age

Long

26

Created Date

11/11/15

Contact Number

No Contact Number on File.

Worth

-£123,456,789

Wrapped Around a Result List



Testing Tester

Age

Long

35

Created Date

11/11/14

Contact Number

No Contact Number on File.

Worth

£123,456,789



Foo (Bar)

Age

Long

26

Created Date

11/11/15

Contact Number

No Contact Number on File.

Worth

-£123,456,789

track:query

Description

Usage

as element:

```
<track:query [var="{string}"] [host="{string}"] [page-title="{string}"] [path="{string}"] [query="{string}"]> </track:query>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
var (optional)	string	The variable to assign the query to. Default: 'queryToken'
host (optional)	string	Endpoint where the twigkit instance is to send the data to.
page-title (optional)	string	Title of the page where the query is sent to. (Defaults to current page title).
path (optional)	string	Relative path of the page where the query is sent to. (Defaults to current URL).
query (optional)	string	The name of the query to send to the server, defaults to parameters in the URL if no query is entered.

trigger:event

Description

Triggers \$rootScope event on click

Usage

as attribute

```
<ANY trigger-event="{string}" [trigger-delay="{string}"]> ... </ANY>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
trigger-event	string	Name of the event to trigger
trigger-delay (optional)	string	Milliseconds to delay event trigger

url:tabs

Description

The Url Tabs Directive by clicking on a tab will not add the filter to the URL it will redirect the user to '/action/tab_value' if the action attribute is present on the tag. If not it will just redirect to '/tab_value'.

Usage

as element:

```
<url:tabs response="{object}" tabs="{string}" [field-name="{string}"] [all-label="{string}"] [platform="{string}"] [query="{string}"] [count-number="{string}"] [action="{string}"]> </url:tabs>
```

Directive info

- This directive creates new scope.

Parameters

Par am	Ty pe	Details
res pon se	ob je ct	A response object
tab s	st ri ng	The tabs to output. Specified as a comma separated list. When using filter tabs, you can either specify the actual filter name, or the actual filter name with an alternative display name. To substitute actual filter names with alternative display names use the following notation: 'programme=Programmes'.
fie ld- nam e (op tio nal)	st ri ng	The field to apply filters to when in 'filter' mode.
all - lab el (op tio nal)	st ri ng	Label for the option to turn off all filters. If this attribute is specified, an additional link will be displayed to show all Results (e.g. not applying a Filter).
pla tfo rm (op tio nal)	st ri ng	The Platform to use when submitting the facet without any filters applied. This is used to to get the filter counts for the original query after a user clicks on a filter (e.g. drills down).
que ry (op tio nal)	st ri ng	The name of the query object used to get the count values.
cou nt- num ber (op tio nal)	st ri ng	Specify how the number representing the count should be formatted. Optional values are 'plain', 'formatted' and 'rounded'.

Par am	Ty pe	Details
act ion (op tio nal)	st ri ng	Url to redirect.

Events

tab_changed

When the tab is changed on this directive an event called `tab_changed` will be fired with the name of the tab that is now active.

This event can be listened to like so.

```
$scope.$on('tab_changed', function(e, tab){
    console.log('New Tab = ' + tab);
});
```

Type:

broadcast

Target:

root scope

Example

Source

```
<search:platform var="platform" conf="platforms.test"></search:platform>
<search:query var="query" parameters='*></search:query> <h3 id="example_without-
counts">Without Counts</h3> <url:tabs response="response" tabs="Day 1,Day 2"
field-name="test1" all-label="All Records"></url:tabs> <hr /> <h3
id="example_with-counts">With Counts</h3> <h4 id="example_with-counts_formatted-
count-number">Formatted Count Number</h4> <url:tabs response="response" tabs="Day
1,Day 2,Day 3" field-name="test1" all-label="All Records" platform="platform"
query="query" count-number="formatted"></url:tabs> <hr /> <h4 id="example_with-
counts_plain-count-number">Plain Count Number</h4> <urlrch:tabs
response="response" tabs="Day 1,Day 2,Day 3" field-name="test1" all-label="All
Records" platform="platform" query="query" count-number="plain"></urlrch:tabs>
<hr /> <h4 id="example_with-counts_rounded-count-number">Rounded Count
Number</h4> <url:tabs response="response" tabs="Day 1,Day 2,Day 3,Day 4" field-
name="test1" all-label="All Records" platform="platform" query="query" count-
number="rounded"></url:tabs> <hr />
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService', function($scope,$timeout,ResponseService)
{ $scope.response = { page: 2, query: { rpp: 50 }, facets: { test1: { id:'test1',
display:'test1', field:'test1', filters: [ { val: { dsp: 'Day 1', act: 'Day 1',
}, count: 1000, }, { val: { dsp: 'Day 2', act: 'Day 2', }, count: 2000, }, { val:
{ dsp: 'Day 3', act: 'Day 3', }, count: 2990, }, { val: { dsp: 'Day 4', act: 'Day
4', }, count: 0, } ] }, test2: { id:'test2', display:'test2', field:'test2',
filters: [ { val: { dsp: 'Day 1', act: 'Day 1', }, count: 10, }, { val: { dsp:
'Day 2', act: 'Day 2', }, count: 20, } ] }, empty: { id:'empty', display:'Empty',
field:'empty', filters: [] }, }, results: [] }
ResponseService.setResponse('response', $scope.response); }]);
```

Result

Without Counts

All Records Day 1 Day 2

With Counts

Formatted Count Number

All Records Day 1 Day 2 Day 3

Plain Count Number

Rounded Count Number

All Records Day 1 Day 2 Day 3 Day 4

widget:carousel

Description

Creates a horizontal carousel

Usage

as element:

```
<widget:carousel [carousel-name="{string}"] [carousel-watch="{Object}"] [loop="{boolean}"] [auto-play="{number}"] [random-start="{boolean}"] [controls="{boolean}"] [height="{number}"] [pagination="{boolean}"]> </widget:carousel>
```

as attribute

```
<ANY widget-carousel [carousel-name="{string}"] [carousel-watch="{Object}"] [loop="{boolean}"] [auto-play="{number}"] [random-start="{boolean}"] [controls="{boolean}"] [height="{number}"] [pagination="{boolean}"]> ... </ANY>
```

Parameters

Param	Type	Details
carousel-name (optional)	string	The name of the carousel so it can be retrieved from the Carousel service. This can be used to control the carousel from another element.
carousel-watch (optional)	Object	A variable to watch for changes to the slides data. When creating the slides in an ngRepeat the object should be set here to watch for changes.
loop (optional)	boolean	Whether the carousel should loop. Default true.
autoplay (optional)	number	Milliseconds to wait until animating to the next slide. By default the carousel will not auto slide
random-start (optional)	boolean	Randomises starting slide
controls (optional)	boolean	Adds left and right arrows for navigation. Default true.
height (optional)	number	Set the height of the carousel. By default the height is dynamically set to the height of the active slide.
pagination (optional)	boolean	Whether to show pagination dots. Default false.

Example

Source

```
<widget:carousel> <slidecontainer> <slide class="demo">Slide 1</slide> <slide  
class="demo">Slide 2</slide> <slide class="demo">Slide 3</slide>  
</slidecontainer> </widget:carousel>
```

```
.demo { min-height: 300px; text-align: center; margin-top: 140px; } .tk-stl-  
carousel { display: block; width: 100%; height: 300px; white-space: nowrap;  
position: relative; overflow: hidden; transition: height .2s ease } .tk-stl-  
carousel slidecontainer { display: block; position: absolute; top: 0; width:  
100%; height: 100%; z-index: 1; font-size: 0 } .tk-stl-carousel  
slidecontainer.carousel-ignore-first-slide { -webkit-transform: translate(-100%,  
0); -ms-transform: translate(-100%, 0); transform: translate(-100%, 0) } .tk-stl-  
carousel slidecontainer.carousel-animate { transition: -webkit-transform .5s  
ease-out 0s; transition: transform .5s ease-out 0s; transition: transform .5s  
ease-out 0s, -webkit-transform .5s ease-out 0s } .tk-stl-carousel slidecontainer  
slide { position: relative; display: inline-block; vertical-align: top; font-  
size: 16px; width: 100%; height: 100%; background-size: cover; background-  
position: center } .tk-stl-carousel slidecontainer slide img { pointer-events:  
none; margin: 0 auto; display: block } .tk-stl-carousel .carousel-arrow {  
display: inline-block; display: -webkit-flex; display: -ms-flexbox; display:  
flex; -webkit-justify-content: center; -ms-flex-pack: center; justify-content:  
center; -webkit-align-items: center; -ms-flex-align: center; align-items: center;  
position: absolute; top: 0; bottom: 0; z-index: 1; width: 100px; text-align:  
center; cursor: pointer } .tk-stl-carousel .carousel-arrow.carousel-arrow-left {  
left: 0 } .tk-stl-carousel .carousel-arrow.carousel-arrow-right { right: 0 } .tk-  
stl-carousel .carousel-arrow .icon-angle-left, .tk-stl-carousel .carousel-arrow  
.icon-angle-right { font-size: 2.5em }
```

Result



Slide 1



widget:date-range

Description

Creates two date-time picker inputs for setting a date range filter.

Usage

as element:

```
<widget:date-range field="{string}" [date-format="{string}"] [from-label="{string}"] [to-label="{string}"] [on-set="{function}"] [min-view="{string}"] [from-model="{string}"] [to-model="{string}"]> </widget:date-range>
```

Parameters

Param	Type	Details
field	string	Field to apply filter to
date-format (optional)	string	Date format to display
from-label (optional)	string	From input label
to-label (optional)	string	To input label
on-set (optional)	function	Function to call when a date has been set instead of applying filters. Two arguments are passed to the function from Date object, to Date object
min-view (optional)	string	The minimum view of the date picker to display. Values: 'year/month/day/hour/minute'. Default 'minute'.
from-model (optional)	string	A binding to a scoped variable, the from date picker will be set using this value.
to-model (optional)	string	A binding to a scoped variable, the to date picker will be set using this value.

Example

Source

```
<widget:date-range field="date"></widget:date-range>
```

Result

From	To
From Date	To Date

widget:datetimepicker

Description

The Header Search directive

Usage

as element:

```
<widget:datetimepicker ng-model="{Object}" [on-set-time="{Object}"] [datepicker-config="{Object}"] date-from="{String}"> </widget:datetimepicker>
```

Parameters

Param	Type	Details
ngModel	Object	Scope variable to update
onSetTime (optional)	Object	Function to call when the date/time has been selected
datepickerConfig (optional)	Object	Config for datepicker. \{minView: 'year/month/day/hour'}
dateFrom	String	Earliest pickable date, updates automatically when changed.

Example

Source

```
<input type="text" ng-model="data | date:'fullDate'" /> <widget:datetimepicker data-ng-model="data" data-on-set-time="onTimeSet(newDate, oldDate)"> </widget:datetimepicker>
```

```
angular.module('lightning') .controller('ExampleController', ['$scope', function($scope) { $scope.data = new Date(); $scope.onTimeSet = function(newDate, oldDate) { console.log(newDate, oldDate); } }]);
```

Result

Friday, May 21, 2021

widget:file-hierarchy

Description

Creates a horizontal column view for a file hierarchy

Usage

as element:

```
<widget:file-hierarchy files="{mixed}" on-submit="{function}" button-label="{string}"> </widget:file-hierarchy>
```

Parameters

Param	Type	Details
files	mixed	The hierarchy of files
on-submit	function	A function to call when the submit button is clicked
button-label	string	Label for the button, button only shows if a label is not empty

widget:icon

Description

Creates an icon

Usage

as element:

```
<widget:icon icon="{string}"> </widget:icon>
```

Parameters

Param	Type	Details
icon	string	The name of the icon to display.

Example

Source

```
<layout:grid> <layout:block medium="1-3"> <h3 id="example_icons">Icons</h3>
<widget:icon icon="heart"></widget:icon> heart<br /> <widget:icon icon="heart-
empty"></widget:icon> heart-empty<br /> <widget:icon icon="bookmark">
</widget:icon> bookmark<br /> <widget:icon icon="bookmark-empty"></widget:icon>
bookmark-empty<br /> <widget:icon icon="search"></widget:icon> search<br />
<widget:icon icon="download-cloud"></widget:icon> download-cloud<br />
<widget:icon icon="upload-cloud"></widget:icon> upload-cloud<br /> <widget:icon
icon="star"></widget:icon> star<br /> <widget:icon icon="star-empty">
</widget:icon> star-empty<br /> <widget:icon icon="star-half"></widget:icon>
star-half<br /> <widget:icon icon="star-half-alt"></widget:icon> star-half-alt<br
/> <widget:icon icon="lock"></widget:icon> lock<br /> <widget:icon icon="lock-
open-alt"></widget:icon> lock-open-alt<br /> <widget:icon icon="attention">
</widget:icon> attention<br /> <widget:icon icon="cog"></widget:icon> cog<br />
</layout:block> <layout:block medium="1-3"> <h3 id="example_styles">Styles</h3>
<p>Tiny</p> <widget:icon icon="heart" style="tiny"></widget:icon> <br />
<p>Small</p> <widget:icon icon="heart" style="small"></widget:icon> <br />
<p>Large</p> <widget:icon icon="heart" style="large"></widget:icon> <br />
<p>Huge</p> <widget:icon icon="heart" style="huge"></widget:icon> <br />
<p>Massive</p> <widget:icon icon="heart" style="massive"></widget:icon> <br />
</layout:block> <layout:block medium="1-3"> <h3 id="example_colours">Colours</h3>
<p> <widget:icon icon="heart" style="massive text-carrot"></widget:icon> Carrot
</p> <p> <widget:icon icon="heart" style="massive text-sunflower"></widget:icon>
Sunflower </p> <p> <widget:icon icon="heart" style="massive text-grass">
</widget:icon> Grass </p> <p> <widget:icon icon="heart" style="massive text-
mint"></widget:icon> Mint </p> <p> <widget:icon icon="heart" style="massive text-
sky"></widget:icon> Sky </p> <p> <widget:icon icon="heart" style="massive text-
lavender"></widget:icon> Lavender </p> <p> <widget:icon icon="heart"
style="massive text-strawberry"></widget:icon> Strawberry </p> </layout:block>
</layout:grid>
```

Result

Icons	Styles	Colours
♥ heart		
♡ heart-empty	Tiny	♥ Carrot
🔖 bookmark	♥	
🔖 bookmark-empty	Small	♥ Sunflower
🔍 search	♥	♥ Grass
☁ download-cloud	Large	♥ Mint
☁ upload-cloud	♥	♥ Sky
★ star	Huge	♥ Lavender
☆ star-empty	♥	♥ Strawberry
★ star-half		
★ star-half-alt	Massive	
🔒 lock	♥	
🔒 lock-open-alt		
⚠ attention		
⚙ cog		

widget:in-view

Description

Wrap around any element and it will only be loaded when it is in view.

Usage

as element:

```
<widget:in-view> </widget:in-view>
```

widget:license

Description

Shows the Twigkit license expiration date. When transcluded the following scoped variables are available: (bool) licenseExpired (int) licenseDaysRemaining (string) licenseExpiryDate (bool) show

Usage

as element:

```
<widget:license [days-left="{string}"] [date-format="{string}"]>
</widget:license>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
daysLeft (optional)	string	The number of days left on the license. Also accepts numbers in the format <5 or >5 to specify a range
dateFormat (optional)	string	The date format to show the expiration date in

widget:login-form

Description

Creates a login form

Usage

as element:

```
<widget:login-form [logo="{string}"] [logo-alt="{string}"] [title="{string}"]  
[message="{string}"] [action="{string}"] [method="{string}"] [username="  
{string}"] [password="{string}"] [username-placeholder="{string}"] [password-  
placeholder="{string}"] [remember="{string}"] [submit-label="{string}"] [access-  
denied="{string}"] [error-message="{string}"] [reset-link="{string}"] [create-  
link="{string}"] [username-error-message="{string}"] [password-error-message="  
{string}"]> </widget:login-form>
```

Parameters

Param	Type	Details
logo (optional)	string	The path to the logo
logo-alt (optional)	string	The alt text for the logo
title (optional)	string	The title text that displays below the logo
message (optional)	string	The message to display below the title
action (optional)	string	URL to send the form data to
method (optional)	string	The form method to use. Default POST
username (optional)	string	The name of the the username input
password (optional)	string	The name of the the password input
username-placeholder (optional)	string	The placeholder of the the username input
password-placeholder (optional)	string	The placeholder of the the password input
remember (optional)	string	The name of the the remember input. Set to false to hide remember me checkbox (default: "remember")
submit-label (optional)	string	The text on the submit button. (default: "Login")
access-denied (optional)	string	Checks the URL for the value of access-denied, if found the error message will be displayed E.g. access=denied

Param	Type	Details
error-message (optional)	string	The error message to display when the login is rejected. (default: "Invalid username or password")
reset-link (optional)	string	Provide a link to reset the user's password
create-link (optional)	string	Provide a link to for a user to request an account
username-error-message (optional)	string	Message to display if the username is not entered or valid. (default: "This field is required.")
password-error-message (optional)	string	Message to display if the password is not entered or valid. (default: "This field is required.")

Example

Source

```
<widget:login-form></widget:login-form>
```

Result

widget:login-page

Description

Creates a login page with a form. All of the attributes are passed to the form.

Usage

as element:

```
<widget:login-page [logo="{string}"] [logo-alt="{string}"] [title="{string}"]  
[message="{string}"] [action="{string}"] [method="{string}"] [username="  
{string}"] [password="{string}"] [username-placeholder="{string}"] [password-  
placeholder="{string}"] [remember="{string}"] [submit-label="{string}"] [access-  
denied="{string}"] [error-message="{string}"]> </widget:login-page>
```

Parameters

Param	Type	Details
logo (optional)	string	The path to the logo
logoAlt (optional)	string	The alt text for the logo
title (optional)	string	The title text that displays below the logo
message (optional)	string	The message to display below the title
action (optional)	string	URL to send the form data to
method (optional)	string	The form method to use. Default POST
username (optional)	string	The name of the username input
password (optional)	string	The name of the password input
username- placeholder (optional)	string	The placeholder of the username input
password- placeholder (optional)	string	The placeholder of the password input
remember (optional)	string	The name of the remember input. Default: remember. Set to false to hide remember me checkbox (default: "remember")
submit-label (optional)	string	The text on the submit button. Default: Login (default: "Login")
access- denied (optional)	string	Checks the URL for the value of access-denied, if found the error message will be displayed E.g. access=denied

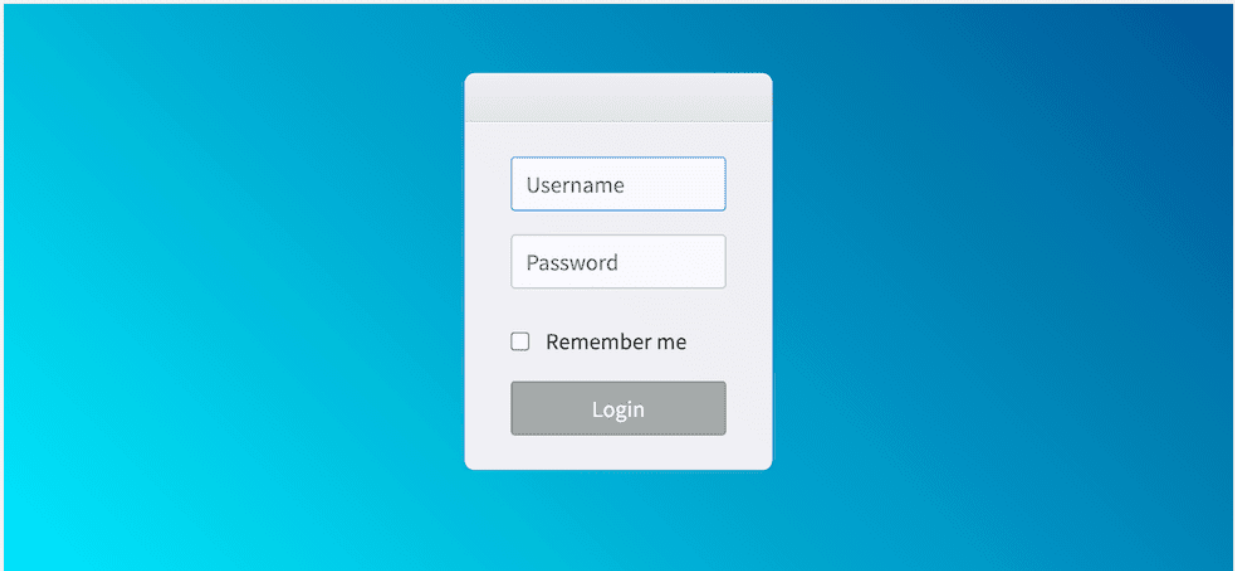
Param	Type	Details
error- message (optional)	string	The error message to display when the login is rejected. Default: Invalid username or password (default: "Invalid username or password")

Example

Source

```
<widget:login-page></widget:login-page>
```

Result



widget:logout

Description

Redirects the user to `<app context>/logout/` when clicked

Usage

as element:

```
<widget:logout> </widget:logout>
```

Directive info

- This directive creates new scope.

Example

Source

```
<widget:logout></widget:logout>
```

Result

Logout

widget:modal

Description

Creates a modal. The directive is automatically moved to the body so it works consistently.

Usage

as element:

```
<widget:modal id="{string}" [title="{string}"] [modal-classes="{string}"]>
</widget:modal>
```

Parameters

Param	Type	Details
id	string	An ID that can be used to trigger the modal to open
title (optional)	string	Title for the modal
modal-classes (optional)	string	Add classes to the modal elements

Example

Source

```
<button class="tk-stl-button" widget-open-modal="id">Open Modal</button>
<widget:modal title="Hello World" id="id"> <search:platform var="platform"
conf="platforms.solr"></search:platform> <search:query var="query" query="docs">
</search:query> <search:response var="response" query="query"
platform="platform"></search:response> <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.</p> </widget:modal>
```

Result

Open Modal

widget:open-modal

Description

Applies a class to the target when toggled on.

Usage

as attribute

```
<ANY widget-open-modal="{string}"> ... </ANY>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
widget-open-modal	string	The ID of the modal to open

Example

Source

```
<button class="tk-stl-button" widget-open-modal="id">Open Modal</button>
<widget:modal title="Hello World" id="id"> <p>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.</p> </widget:modal>
```

Result

Open Modal

widget:popover

Description

The popover widget allows the popover content to be displayed in a popup when the popover trigger is clicked.

Usage

as element:

```
<widget:popover [direction="{string}"] [mode="{string}"] [close-on-click="{string}"] [close-on-mouseleave="{string}"] [on-close="{function}"] [hover-timeout="{number}"]> </widget:popover>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
direction (optional)	string	Where the popover should appear (top, left, right, bottom, bottom-left, bottom-center-left, bottom-center-right). Default: bottom
mode (optional)	string	How the popover should open (hover or click). Default: click
close-on-click (optional)	string	Close the popover when the popover is clicked on. Default: false
close-on-mouseleave (optional)	string	Close the popover when the mouse leaves the element.
on-close (optional)	function	Function to call when the popover is closed.
hover-timeout (optional)	number	A number to determining the number of milliseconds the popover should be displayed for after the user leaves the popover-trigger. (Only applicable to hover mode.) Default: 1000

Example

Source

```
<layout:box> <widget:popover direction="bottom" styling="popover-medium">
<popover-trigger> <button styling="button">Bottom</button> </popover-trigger>
<popover-content> <h3 id="example_title">Title</h3> <p>Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Consequuntur cum distinctio dolores enim esse
fuga incidunt laboriosam maxime nam odio pariatur provident, quae quam quidem
quod saepe voluptas. Maiores, obcaecati.</p> </popover-content> </widget:popover>
</layout:box> <layout:box> <widget:popover direction="top" styling="popover-
small"> <popover-trigger><button styling="button">Top</button></popover-trigger>
<popover-content> <h3 id="example_title">Title</h3> <p>Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Consequuntur cum distinctio dolores enim esse
fuga incidunt laboriosam maxime nam odio pariatur provident, quae quam quidem
quod saepe voluptas. Maiores, obcaecati.</p> </popover-content> </widget:popover>
</layout:box> </div> <layout:box> <widget:popover direction="left"> <popover-
trigger><button styling="button">Left</button></popover-trigger> <popover-
content> <h3 id="example_title">Title</h3> <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit. Consequuntur cum distinctio dolores enim esse fuga
incidunt laboriosam maxime nam odio pariatur provident, quae quam quidem quod
saepe voluptas. Maiores, obcaecati.</p> </popover-content> </widget:popover>
</layout:box> <layout:box> <widget:popover direction="right" styling="popover-
large"> <popover-trigger><button styling="button">Right</button></popover-
trigger> <popover-content> <h3 id="example_title">Title</h3> <p>Lorem ipsum dolor
sit amet, consectetur adipisicing elit. Consequuntur cum distinctio dolores enim
esse fuga incidunt laboriosam maxime nam odio pariatur provident, quae quam
quidem quod saepe voluptas. Maiores, obcaecati.</p> </popover-content>
</widget:popover> </layout:box> <layout:box> <widget:popover mode="hover">
<popover-trigger><button styling="button">Hover</button></popover-trigger>
<popover-content> <h3 id="example_hover">Hover</h3> <p>Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Accusantium aliquam debitis deserunt eum
minus molestiae mollitia perspiciatis possimus provident quae. Architecto
corporis dicta ea harum odio saepe suscipit, totam ut!</p> </popover-content>
</widget:popover> </layout:box>
```

Result

Bottom

Top

Left

Right

Hover

widget:scroll-to

Description

When clicked scrolls to the top of the page or to the specified element by ID. Styles can be added to fix the scroll to top to the bottom left right or center of the page by adding the style tag with 'bottom-right' OR 'bottom-center' OR 'bottom-left'

Usage

as element:

```
<widget:scroll-to [scroll-to-id="{string}"]> </widget:scroll-to>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
scroll-to-id (optional)	string	The id of the element to scroll to.

widget:slide-down

Description

The toggle panes directive

Usage

as attribute

```
<ANY widget-slide-down [duration="{string}"] [easing="{string}"] [closed="{boolean}"] [height="{Number}"] [response="{Object}"]> ... </ANY>
```

as class

```
<ANY class="widget-slide-down [duration: {string};] [easing: {string};] [closed: {boolean};] [height: {Number};] [response: {Object};]"> ... </ANY>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
duration (optional)	string	Duration of the animation. Default 0.3 (seconds)
easing (optional)	string	CSS timing function i.e. ease-out
closed (optional)	boolean	Starts the slidable element off as open
height (optional)	Number	Set the height of the opened panel. Useful if the content is dynamically loaded.
response (optional)	Object	Wait for a response before calculating the height and check for updates

Example

Source

```
<button class="tk-stl-button" widget-slide-toggle="#foo1">Expand 1</button> <div id="foo1" class="widget-slide-down"> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.</p> <br> <p padding-bottom="2">Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.</p> </div>
```

Result

Expand 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.

Ad adipisci deserunt, doloremque ducimus enim nemo officia pariatur quibusdam quos ratione suscipit, tempora temporibus voluptatem. Ea eveniet incidunt magni minus quaerat.

widget:slider

Description

Creates a slider

Usage

as element:

```
<widget:slider [response="{object}"] [query="{string}"] [facet-name="{string}"]  
[facet="{string}"] [field-name="{string}"] [min="{number}"] [max="{number}"]  
[step="{number}"] [button-label="{string}"] [to-label="{string}"] [prefix="  
{string}"] [suffix="{string}"] [invert-range="{boolean}"] [multiplier="{number}"]  
[date-format="{string}"] [number-format="{string}"] [sort="{boolean}"]>  
</widget:slider>
```

Parameters

Param	Type	Details
response (optional)	object	A response object
query (optional)	string	The name of the query to build links from.
facet-name (optional)	string	Facet name to use from the response object
facet (optional)	string	A facet object to display
field-name (optional)	string	The field name tells the slider which field to use when creating range filters.
min (optional)	number	The minimum value for the slider. (Defaults to 0, not required for facets)
max (optional)	number	The maximum value for the slider. (Defaults to 100, not required for facets)
step (optional)	number	The interval between each step of the slider. (Defaults to 1, not required for facets)
button-label (optional)	string	The text to be used for the 'Apply' button.
to-label (optional)	string	The string to be inserted in the filter's display value between the 'from' and 'to' value. (Defaults to '-')

Param	Type	Details
prefix (optional)	string	Add a string before the display value. Useful for currency symbols, for instance.
suffix (optional)	string	Add a string after the display value. Useful for adding units, such as “k” for thousands.
invert-range (optional)	boolean	When invertRange is set to true, the from and to values of the resulting range filter are swapped.
multiplier (optional)	number	The multiplier parameter accepts a number and multiplies the slider’s value by the multiplier when creating a filter.
date-format (optional)	string	How to format the date. See #/api/lightning.filter:dateFormat[date format filter] for valid formats.
number-format (optional)	string	Format string for the numbers used using the Java DecimalFormat pattern. Default: ',0.'
sort (optional)	boolean	Whether the data should be sort by the directive. Defaults to true.

Example

Source

```
<h3 id="example_slider">Slider</h3> <widget:slider field-name="foo" min="0"
max="100" step="1"></widget:slider> <hr /> <h3 id="example_slider-with-date-
formatting">Slider with Date Formatting</h3> <widget:slider response="response"
facet-name="test1" date-format="fullDate"></widget:slider> <hr /> <h3
id="example_facet_slider">Facet Slider</h3> <widget:slider response="response"
facet-name="test2"></widget:slider> <hr /> <chart:histogram facet="response |
facet:'Last Modified'" show="100" show-count="true" date-format="MM-yyyy">
</chart:histogram> <widget:slider response="response" facet-name="Last Modified"
date-format="MM-yyyy"></widget:slider>
```

```

angular.module('lightning').controller('ExampleController',
['$scope','$timeout', 'ResponseService', function($scope,$timeout,
ResponseService) { $scope.response = { page: 2, query: { rpp: 50, filters:[] },
facets: { test1: { filters: [ { val: { dsp: 1288323623006, act: 1288323623006, },
count: 10, }, { val: { dsp: 1467882419615, act: 1467882419615, }, count: 20, } ],
'max-count':20 }, test2: { filters: [ { val: { dsp: '- 1998-12-31', act: {
type:'range', to:'1998-12-31T00:00:00Z' }, }, count: 10, }, { val: { dsp: '1999-
01-01 - 1999-12-31', act: { type:'range', from:'1999-01-01T00:00:00', to:'1999-
12-31T00:00:00Z' }, }, count: 20, }, { val: { dsp: '2000-01-01 - ', act: {
type:'range', from:'2000-01-01T00:00:00Z', }, }, count: 20, } ], 'max-count':20
}, 'Last Modified':{'id':"Last Modified","field":"Last Modified","filters":
[{"field":"Last
Modified","optional":"none","match":"exact","exclude":false,"count":294,"weight":
0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2001-01-
01T00:00:00","to":"2002-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2001-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":403,"weight":
0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2002-01-
01T00:00:00","to":"2003-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2002-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":592,"weight":
0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2003-01-
01T00:00:00","to":"2004-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2003-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":488,"weight":
0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2004-01-
01T00:00:00","to":"2005-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2004-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":1610,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2005-01-
01T00:00:00","to":"2006-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2005-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":4947,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2006-01-
01T00:00:00","to":"2007-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2006-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":4058,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2007-01-
01T00:00:00","to":"2008-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2007-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":7837,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2008-01-
01T00:00:00","to":"2009-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2008-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":6011,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2009-01-
01T00:00:00","to":"2010-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2009-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":7584,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2010-01-
01T00:00:00","to":"2011-01-
01T00:00:00","fromExcluded":false,"toExcluded":true},"dsp":"2010-01-
01T00:00:00Z"},"has-more-rel":false,"rel":[]},{field:"Last
Modified","optional":"none","match":"exact","exclude":false,"count":6284,"weight"
:0,"facts":{},"grp":-1,"val":{"act":{"type":"range","from":"2011-01-

```



```

01T00:00:00", "to": "2012-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2011-01-
01T00:00:00Z", "has-more-rel": false, "rel": [], {"field": "Last
Modified", "optional": "none", "match": "exact", "exclude": false, "count": 13003, "weight
": 0, "facts": {}, "grp": -1, "val": {"act": {"type": "range", "from": "2012-01-
01T00:00:00", "to": "2013-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2012-01-
01T00:00:00Z", "has-more-rel": false, "rel": [], {"field": "Last
Modified", "optional": "none", "match": "exact", "exclude": false, "count": 32742, "weight
": 0, "facts": {}, "grp": -1, "val": {"act": {"type": "range", "from": "2013-01-
01T00:00:00", "to": "2014-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2013-01-
01T00:00:00Z", "has-more-rel": false, "rel": [], {"field": "Last
Modified", "optional": "none", "match": "exact", "exclude": false, "count": 156357, "weigh
t": 0, "facts": {}, "grp": -1, "val": {"act": {"type": "range", "from": "2014-01-
01T00:00:00", "to": "2015-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2014-01-
01T00:00:00Z", "has-more-rel": false, "rel": [], {"field": "Last
Modified", "optional": "none", "match": "exact", "exclude": false, "count": 42345, "weight
": 0, "facts": {}, "grp": -1, "val": {"act": {"type": "range", "from": "2015-01-
01T00:00:00", "to": "2016-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2015-01-
01T00:00:00Z", "has-more-rel": false, "rel": [], {"field": "Last
Modified", "optional": "none", "match": "exact", "exclude": false, "count": 53638, "weight
": 0, "facts": {}, "grp": -1, "val": {"act": {"type": "range", "from": "2016-01-
01T00:00:00", "to": "2017-01-
01T00:00:00", "fromExcluded": false, "toExcluded": true}, "dsp": "2016-01-
01T00:00:00Z", "has-more-rel": false, "rel": []}, "facts": {}, "params":
{"display": "Last Modified", "max-count": 156357.0, "min-count": 0.0, "sum-of-
counts": 338193.0}, }, results: [] }; ResponseService.setResponse('response',
$scope.response); });

```

Result

Slider



Slider with Date Formatting



Facet Slider



widget:spelling-suggestions

Description

Usage

as element:

```
<widget:spelling-suggestions response="{string}" query="{string}">
</widget:spelling-suggestions>
```

Parameters

Param	Type	Details
response	string	The name of the response from which to retrieve the spelling suggestion messages.
query	string	The name of the query originally submitted by the user. This can differ from the one on the Response which was sent to the search platform.

Example

Source

```
<h3 id="example_not-auto-corrected">Not Auto-Corrected</h3> <h4 id="example_not-
auto-corrected_default">Default</h4> <tk-example> <widget:spelling-suggestions
response="notAutoCorrected" query="query"></widget:spelling-suggestions> </tk-
example> <tk-code-wrapper> <code hljs hljs-no-escape>&lt;widget:spelling-
suggestions response="notAutoCorrected" query="query"&gt;&lt;/widget:spelling-
suggestions&gt;</code> </tk-code-wrapper> <hr /> <h4 id="example_not-auto-
corrected_with-label-and-suffix">With Label and Suffix</h4> <tk-example>
<widget:spelling-suggestions response="notAutoCorrected" query="query"
label="Were you looking for:" suffix="?"></widget:spelling-suggestions> </tk-
example> <tk-code-wrapper> <code hljs hljs-no-escape>&lt;widget:spelling-
suggestions response="notAutoCorrected" query="query" label="Were you looking
for:" suffix="?"&gt;&lt;/widget:spelling-suggestions&gt;</code> </tk-code-
wrapper> <hr /> <h3 id="example_auto-corrected">Auto-Corrected</h3> <h4
id="example_auto-corrected_default">Default</h4> <tk-example> <widget:spelling-
suggestions response="autoCorrected" query="query"></widget:spelling-suggestions>
</tk-example> <tk-code-wrapper> <code hljs hljs-no-escape>&lt;widget:spelling-
suggestions response="autoCorrected" query="query"&gt;&lt;/widget:spelling-
suggestions&gt;</code> </tk-code-wrapper> <hr /> <h4 id="example_auto-
corrected_with-label-and-suffix">With Label and Suffix</h4> <tk-example>
<widget:spelling-suggestions response="autoCorrected" query="query" original-
label="Couldn't find anything for:" original-suffix="." :(" corrected-label="How
about:" corrected-suffix="?" :)"></widget:spelling-suggestions> </tk-example> <tk-
code-wrapper> <code hljs hljs-no-escape>&lt;widget:spelling-suggestions
response="autoCorrected" query="query" original-label="Couldn't find anything
for:" original-suffix="." :(" corrected-label="How about:" corrected-suffix="?"
:)"&gt;&lt;/widget:spelling-suggestions&gt;</code> </tk-code-wrapper> <hr />
```

```
angular.module('lightning') .controller('ExampleController',
['$scope','$timeout','ResponseService','QueryService',
function($scope,$timeout,ResponseService,QueryService) { $scope.notAutoCorrected
= { page: 2, query: { rpp: 50 }, facets: {}, results: [], "messages": [ { "type":
"spelling", "count": 0, "label": "Twigkit", "query": "Twigkit", "auto-corrected":
false } ] }; $scope.autoCorrected = { page: 2, query: { rpp: 50 }, facets: {},
results: [], "messages": [ { "type": "spelling", "count": 0, "label": "Twigkit",
"query": "Twigkit", "auto-corrected": true } ] }; $scope.query = { q:'tiwkit' };
ResponseService.setResponse('notAutoCorrected', $scope.notAutoCorrected);
ResponseService.setResponse('autoCorrected', $scope.autoCorrected);
QueryService.setQuery('query',$scope.query); }]);
```

Result

Not Auto-Corrected

Default

EXAMPLE

Did you mean: [Twigkit](#)

```
<widget:spelling-suggestions response="notAutoCorrected" query="query"></widget:spelling-suggestions>
```

With Label and Suffix

EXAMPLE

Were you looking for: [Twigkit](#)?

```
<widget:spelling-suggestions response="notAutoCorrected" query="query" label="Were you looking for:" suffix="?"></widget:spelling-suggestions>
```

Auto-Corrected

Default

EXAMPLE

No results found for tiwkit. Showing results for [Twigkit](#).

```
<widget:spelling-suggestions response="autoCorrected" query="query"></widget:spelling-suggestions>
```

With Label and Suffix

EXAMPLE

Couldn't find anything for: tiwkit. :(How about: [Twigkit](#)? :)

```
<widget:spelling-suggestions response="autoCorrected" query="query" original-label="Couldn't find anything for:" original-suffix=". :(" corrected-label="How about:" corrected-suffix="?"></widget:spelling-suggestions>
```

widget:spinner

Description

Creates a loading spinner

Usage

as element:

```
<widget:spinner> </widget:spinner>
```

Example

Source

```
<widget:spinner></widget:spinner>
```

Result



widget:tutorial

Description

Checks a cookie to determine whether to show the content and sets the cookie when an event is triggered

Usage

as element:

```
<widget:tutorial cookie="{string}" [event="{string}"] [delete="{string}"]>
</widget:tutorial>
```

Directive info

- This directive creates new scope.

Parameters

Param	Type	Details
cookie	string	The name of the cookie to check and set
event (optional)	string	The name of the event to set the cookie. Defaults to cookie attribute '_set'
delete (optional)	string	The name of the event to delete the cookie. Defaults to cookie attribute '_delete'

App Studio Concepts

The concepts section contains articles that are explanatory in nature. It provides a high-level description the various components, functions, and aspects of App Studio.

To get started, see *Downloads*.

For reference information, see the *reference section*.

- *App Development*
App Studio supports customization using an array of markup elements, or tags, that handle all the complexities of connecting to Fusion and visualizing the response to your queries.
- *App Studio Enterprise*
App Studio Enterprise is a template for building a standalone Appkit application for Fusion. All of the instructions are self contained in the README and relevant configuration files and will lead you through the simple and easy process of configuring an App Studio application.
- *Configuration*
The articles in this section describe how to configure a search application.
- *Extending Appkit*
The Appkit application framework adheres to standard Java web application methodologies making custom development using generally accepted principles perfectly acceptable. In addition to that Appkit provides a number integration hooks for any aspect of the lifecycle and extension points for adding custom capabilities and features to Appkit application.
- *Layout and Styling*
The articles in this section describe the basics of layouts and styling.
- *Navigation*
The articles in this section describe Appkit tags used for navigation.
- *Overview*
The Appkit library of search application components has simple building blocks that can be put together to build complex applications for search and information retrieval.
- *Rendering Search Results*
Search results are the focus of every search interface, and Appkit makes it easy to forge the results precisely to your liking.

App Development

When you first create a UI using the setup wizard, it configures the default search view, located at `app-studio/app/views/search.html`.

App Studio supports customization using an array of markup elements, or tags, that handle all the complexities of connecting to Fusion and visualizing the response to your queries.

These topics explain the details of how to customize your search interface:

- *Introduction to App Studio Markup*
- *Anatomy of A View*
- *Display Search Results*
- *Style an App*
- *Add Navigation Tools*
- *Search Queries*
- *Search Filters*
- *Configuring Autocomplete*
- *Security*
- *Connecting to other search providers*
- *Demo Deployment*

You can edit the default view in your browser using the *code editor* or using any text editor or IDE.

- In *standalone mode*, your changes become available as soon as you save the file. Just reload the page in your browser.
- When your search UI is published as a *Fusion Web app*, you must re-publish the app to make your changes available.
- You can get detailed information about every App Studio markup tag in the *Reference Guides*.
- For ideas and reusable code, see the *App Studio Gallery*.

App Studio Enterprise

App Studio Enterprise is a template for building a standalone Appkit application for Fusion. All of the instructions are self contained in the README and relevant configuration files and will lead you through the simple and easy process of configuring an App Studio application.

To get started, *download the latest copy*, and check out the instructions *here*.

Starting in Fusion 5.0, App Studio is no longer included in the Fusion UI. However, you can use App Studio Enterprise (ASE) to create apps in Fusion 5 and Fusion 4. See *How to Deploy App Studio Enterprise* for instructions.

App Studio Enterprise Downloads

Version	Compatible with	Download link
App Studio Enterprise 4.14.0	Fusion 4.x, 5.x	Zip (https://storage.googleapis.com/docs-downloads/app-studio-enterprise-4.14.0.zip) SHA-256 encrypted zip (https://storage.googleapis.com/docs-downloads/app-studio-enterprise-4.14.0.zip.sha256)
App Studio Enterprise 4.13.0	Fusion 4.x, 5.x	Zip (https://storage.googleapis.com/docs-downloads/app-studio-enterprise-4.13.0.zip) SHA-256 encrypted zip (https://storage.googleapis.com/docs-downloads/app-studio-enterprise-4.13.0.zip.sha256)
App Studio Enterprise 4.11.0	Fusion 4.x, 5.x	Zip (https://download.lucidworks.com/app-studio-enterprise-4.11.0/app-studio-enterprise-4.11.0.zip) SHA-256 encrypted zip (https://download.lucidworks.com/app-studio-enterprise-4.11.0/app-studio-enterprise-4.11.0.zip.sha256)
App Studio Enterprise 4.10.0	Fusion 4.x, 5.0.x, 5.1.x	Zip (http://download.lucidworks.com/app-studio-enterprise-4.10.0/app-studio-enterprise-4.10.0.zip) SHA-256 encrypted zip (http://download.lucidworks.com/app-studio-enterprise-4.10.0/app-studio-enterprise-4.10.0.zip.sha256)
App Studio Enterprise 4.9.0	Fusion 4.x, 5.0.x, 5.1.x	Zip (https://download.lucidworks.com/app-studio-enterprise-4.9.0/app-studio-enterprise-4.9.0.zip) SHA-256 encrypted zip (https://download.lucidworks.com/app-studio-enterprise-4.9.0/app-studio-enterprise-4.9.0.zip.sha256)
App Studio Enterprise 4.8.0	Fusion 4.x, 5.0.x, 5.1.x	Zip (https://download.lucidworks.com/app-studio-enterprise-4.8.0/app-studio-enterprise-4.8.0.zip) SHA-256 encrypted zip (https://download.lucidworks.com/app-studio-enterprise-4.8.0/app-studio-enterprise-4.8.0.zip.sha256)

Get Started with ASE

With App Studio Enterprise you can build a mobile-ready enterprise search application in a matter of minutes. Follow the steps below to get up and running!

Development tools

In building out a search application, we recommend the use of the following tools:

- Java 1.8 JDK
- NodeJS & npm for front-end dependency management
 - NodeJS v16.70.0 (or v16 LTS)
 - npm v9.2.0
- An IDE (IntelliJ, VS Code, Eclipse..)

Connecting to Fusion

We just need to edit a few config files – fire up your favorite text editor! All of the config files can be found within `src/main/resources/conf`.

Edit `services/api/fusion.conf` – this file controls which Fusion server to use:

- Set the value of `host` to be the hostname of your Fusion server, e.g. `host: localhost` or `host: fusion.example.com`
- Set the value of `port` to be the port of your Fusion server – this is `8764` by default, which is preconfigured for you.

Edit `platforms/fusion/data.conf` – this file configures which data in Fusion to use:

- Replace the text `YOUR_QUERY_PROFILE` with the name of the Fusion Query Profile you wish to direct your queries through. If you have created an app via the Fusion UI called `MyAwesomeApp`, then you will have a default Query Profile called `MyAwesomeApp` as well – this is a good starting point: e.g. `query-profile: MyAwesomeApp`.

Edit `platforms/fusion/social.conf` – this file configures the Lucidworks Appkit Social Module:

- Replace the text `YOUR_FUSION_APP` with the name of your Fusion app, as used above: e.g. `collection: YOUR_FUSION_APP_user_prefs` becomes `collection: MyAwesomeApp_user_prefs`. This configures the Appkit SDK to store social data in the `_user_prefs` sidecar collection associated with `MyAwesomeApp`.

Edit `platforms/fusion/suggestions.conf` – this file configures search suggestions:

- Replace the text `YOUR_QUERY_PROFILE` with the name of the Fusion Query Profile you wish to direct your search suggestion queries through. This can be the same Query Profile as you are using for search queries, e.g. `query-profile: MyAwesomeApp`.

Edit `message/service/fusion.conf` – this file configures how Signals are sent to Fusion:

- Replace the text `YOUR_QUERY_PROFILE` with the name of your query profile, as used above: e.g. `query-profile: MyAwesomeApp`.

Starting and Stopping

Linux, Mac OS X

To start the application, run the following command from a terminal:

- `./app-studio start`

To stop the application, run the following command from a terminal:

- `./app-studio stop`

To package the application into a WAR file, run the following command from a terminal:

- `./app-studio package`

Windows

To start the application, run the following command from a command prompt:

- `app-studio.bat start`

To stop the application, run the following command from a Powershell command prompt:

- `app-studio.bat stop`

To package the application into a WAR file, run the following command from a command prompt:

- `app-studio.bat package`

`See Additional script commands for more commands.`

Accessing App Studio

Once started, the application is served up at `http://localhost:8080`. You will be asked to authenticate, for which you can use any of your Fusion "native realm" users – if you are just getting started with Fusion, this will be your Fusion `admin` username and password.

When building an application for production environments the security realm can be configured in the `src/main/resources/conf/security/fusion.conf` file.

Using Your Data

The file `src/main/webapp/views/search.html` defines the main page layout of your application. App Studio Enterprise is pre-configured to work out of the box with Lucidworks Fusion, but you may wish to customize which fields you use in your results.

Let us get started!

Result title

The title of each result document can be configured like so:

Find the tag that looks like this:

```
<search:field name="title_s" styling="title" urlfield="id"></search:field>
```

- `styling="title"` means that it will be used as the result title – no need to change this, but it will help you find the right tag!
- `name="title_s"` configures the application to use the value of the `title_s` field as the displayed title text for each result. If your document titles are stored in a different field, replace `title_s` with the name of that field.
- `urlfield="id"` configures the application to use the value of the `id` field as the hyperlink – when users click on the title, they will be navigated to whatever URL is in the `id` field for that result. If your document URLs are stored in a different field, replace `id` with the name of that field.

Result description

The text description of each result document can be configured like so:

Find the tag that looks like this:

```
<search:field name="description_s" label="Description" max-characters="240" show-more></search:field>
```

- `label="Description"` will help you find the right tag.
- `name="description_s"` configures the application to use the value of the `description_s` field for the description text of each search result. If your descriptions are stored in a different field, replace `id` with the name of that field.

Facets

Facets are a handy way to refine your search. By default, App Studio Enterprise will attempt to show facets for the `keywords_s` field, if it exists. We can add or remove facets very simply:

Find the tag that looks like this:

```
<search:query var="query" parameters="*" results-per-page="12" facets="keywords_s"></search:query>
```

This tag creates the overall page query – `facets="keywords_s"` adds the ability to facet on the `keywords_s` field to the query. We can remove this, or add some more facets – this parameter takes a comma-separated list of field names.

Try `facets="keywords_s,charSet_s"`, or replace the value entirely with a list of your own facet fields.

Activating your changes

To see your changes in action: – Save the file – Refresh your browser – Yes, that is all!

Development

Resources

App Studio Enterprise is preconfigured to automatically reload when you make changes to your code during development.

The following file types are monitored and re-processed when changed: – HTML (`.html`) – JavaScript (`.js`) – LESS (`.less`) – CSS (`.css`) – JSP (`.jsp`) – Config (`*.conf`)

There will be a slight pause between saving the file and being able to view the changes as App Studio Enterprise processes the file. Refresh your browser after about 2 seconds to see the changes.

Java

Hot reloading of Java class files requires some extra steps. The Java Virtual Machine supports hot-swapping of class files as long as the changes are not structural – changes to method bodies and field values are supported, but adding or renaming methods are not. You will need to use an IDE to set this up – the following steps assume you are using JetBrains IntelliJ IDEA:

1. Import the project into IntelliJ
2. Add a new Maven "Run Configuration" – from the "Run" menu, select "Edit Configurations". Click the "+" button, and select "Maven" from the dropdown.
3. Enter a name for your configuration – e.g. "Jetty Run"

4. Enter the following for the "Command Line": `clean jetty:run -Pdevelopment --settings ./bin/settings.xml -Dtwigkit.conf.watch=true`, then click "Ok".
5. Make sure that you are not already running App Studio Enterprise.
5. From the "Run" menu, click "Debug 'Jetty Run'". If you have named your configuration something other than 'Jetty Run', look for "Debug '<your configuration name here>'"
7. As App Studio Enterprise starts, you will see the console output within your IDE
We are now set up to hot reload Java files:

1. Edit a Java file
2. Save the file
3. From the "Build" menu, select "Recompile <your class name>.java"
4. You will see a popup notifying you that a class file has changed, and asking if you would like to reload classes – click "Yes".
5. Your Java change has been deployed into the running Java Virtual Machine – you can now test your running App Studio Enterprise webapp and see your change in action.

License

App Studio Enterprise comes with a 30-day trial license in `app-studio.lic`. Once this expires, contact Lucidworks for a new license: <https://lucidworks.com/contact/>
(<https://lucidworks.com/contact/>)

Additional script commands

Each command and flag is also applicable to the batch script on windows. `=== start`

```
./app-studio start [-f] [-p port] [-P profile] [-m memory] [-t timeout] [-V]
```

`-f` Start App Studio in foreground.

`-p <port>` Specify the port to start the App Studio web server on; default is 8080.

`-P <profile>` Specify the Appkit profile to use when App Studio starts; defaults to development.

`-m <memory>` Sets the min (`-Xms`) and max (`-Xmx`) heap size for the JVM, such as: `-m 4g` results in: `-Xms4g -Xmx4g`; by default, this script sets the heap size to 512m.

`-t <timeout>` Sets the startup timeout in seconds (defaults to 240).

`-V` Verbose messages from the startup script.

stop

```
./app-studio stop [-p port] [-V]
```

`-p <port>` Specify the port the App Studio HTTP listener is bound to.

package

```
./app-studio package [-P profile]
```

-P <profile> Specify the Appkit profile to use when App Studio is packaged; defaults to development.

This command will build a new JAR and WAR file with App Studio and place it in `./dist.`

docker

```
./app-studio docker [package] [start] [-P profile] [-i image_name]
```

package Package App Studio Enterprise then build a Docker image.

start Package App Studio Enterprise, build a Docker image, then run a container on port 8080.

-P <profile> Specify the Appkit profile to use when App Studio Enterprise is packaged; defaults to development.

-i <image_name> Specify the image name to use when App Studio Enterprise is built as a Docker image; defaults to app-studio-enterprise. The image will be tagged as the latest.

Advanced Configuration

Add Custom CSS

Add **class** to any component that needs to be customized in an `.html` file. for example: **customClass** has been added.

```
<search:field name="field_name" label="label" styling="description"  
class="customClass"></search:field>
```

Then add the custom class in **custom.less** which is located in `src/main/webapp/styles/includes/custom.less`.

```
.customClass { font-weight: bold; font-style: normal; }
```

Add Custom JavaScript

App Studio is configured out of the box to allow you to simply add your JavaScript inside the controller function in **search.controller.js** and have it trigger and function on every page in your app.

```
// Print the current page to the console $scope.currentPage =  
window.location.href; console.log($scope.currentPage);
```

Running the app over SSL

The webapp can also be served over HTTPS using SSL encryption. This is done using the following parameters when invoking the startup scripts:

- `-Dtwigkit.ssl=true` required to turn on secure mode

We include a keystore file with a default self-signed key for development / testing of secure mode. However in order to establish proper trust you will need to import your own valid SSL certificate into the file `keystore.jks` or create your own keystore file. You can then configure SSL via:

- `-Dtwigkit.https.port` optionally sets the HTTPS port (defaults to 8765)
- `-Dtwigkit.http.port` optionally sets the HTTP port (defaults to 8080)
- `-Dtwigkit.keystore.file` keystore filename / location relative to webapp (defaults to 'keystore.jks')
- `-Dtwigkit.keystore.password` password to the keystore (defaults to 'p4ssw0rd')
- `-Dtwigkit.keystore.alias` name of the key in the keystore to be used (defaults to 'default-key')

Twigcrypt value encryption utility

A best practice that we advocate is to encrypt any sensitive configuration values using our Twigcrypt utility. Twigcrypt's function is to allow you to quickly encrypt sensitive string values using the command line. These files can be found at [/bin/twigcrypt.zip](#).

For more details on how to use Twigcrypt, refer to the README inside the zip file.

Appkit

The Appkit library of search application components has simple building blocks that can be put together to build complex applications for search and information retrieval.

Appkit provides a generic API for searching and indexing content that completely abstracts the search engine or database in use. This means the application developer does not need to know the ins and outs of each search and database technology, but instead only needs to know how to work with Appkit.

At the most fundamental level:

1. Appkit sends generic queries (which are platform independent) to a platform that translates them into something the underlying search engine can work with.
2. Appkit maps the response from the search engine to a generic Appkit search response which other components can work with and that Appkit UI widgets can render.

Appkit Concepts

Platform

Appkit integrates with a variety of *search platforms*. These are the back-end technologies responsible for indexing the documents for efficient retrieval of search results. These are often referred to as 'search engines' or 'search technologies', but throughout the Appkit framework, we standardize on the term 'platform'.

A platform provides the means to query a collection of data, returning a list of results. The original format of the data could be full-text documents, media (audio/video/images), or structured relational data.

In our documentation, we assume that the target search platforms for the application you are developing are configured and ready to query.

Query

A query is built up from the variety of search controls available in the application to form a bundle of parameters that the platform will understand. Appkit will automatically convert the query into the platform's native format. It will then execute the query against the platform and return the results matching the query.

Queries vary in complexity from a search on a single term to combinations of many 'clauses' with boolean logic (AND/OR/NOT). As far as a user is concerned the search controls provided in Appkit make it easy to build queries which best represent their information need by following intuitive and well established user experience (UX) patterns.

Response

A search response wraps all the information a platform returns from a query including the result list and facets.

Result list

A result list is the collection of data matching a query issued against a platform.

Result

Each result in a result list will contain information stored in named fields. This information can take any form. Appkit provides tags for displaying the fields within a result including styling for displaying common field types. Every result has an `id` field that allows it to be uniquely identified amongst others in the collection of data being searched.

Facets

Facets can be thought of as groupings or categories of results. Typically a platform will return all the values of facets within the results along with a count of the number of results within that category matching the given query.

The common terminology for returning facet values for a particular field is '*faceting on a field*'. Platforms handle facets differently and the facets to return can usually either be statically configured to return with all search results, or dynamically specified with each query sent to the platform. The sensible defaults in Appkit let you easily display all available facets without prior knowledge of those which are statically configured.

Facets allow for a richer search experience as users can click on a particular facet value to limit the results to those containing only the selected value of the field which is faceted on.

Pagination

Nearly all search pages have a limit to the number of results displayed on a page. Pagination refers to the ability to navigate between the pages of results. This can vary in implementation and context. For example when viewing results on a desktop or tablet, you might be given a list of page numbers to choose from, whereas the same page on a

mobile device might display only Next and Previous links. Pagination is usually combined with a control that lets the user choose between some options for the number of search results which appear on a page (for example, 5, 10, 50, 100).

Sort order

The order in which results appear is dependent on how the platform is configured and the 'sort' options applied (if any) in the query. The default *sort order* for most platforms organizes the results by relevance to the query. This is usually based on a calculation using a number of factors. Full-text relevancy factors include: the number of times terms in the query appear in the document, the rarity of the terms across the collection of documents, the number of clauses in a query which are matched by the document and normalization factors such as the length of a field in which a match is found.

Usually in a search application some options for re-sorting a result list based on a particular field are provided (for example, publication date, customer rating).

Relevancy ranking (score)

Most platforms sort results using a calculation of the relevance of a document in the result list to the query. This is often returned as a number in a 'score' field within each result representing the ranking of a document relative to others in the result list. Scores are usually fairly meaningless on their own and only make sense when compared to other results in the same result list.

Filters

Filter is a general term for anything that limits the search results. Many of the Appkit controls allow the user to apply filters to narrow down the results to those they seek. For specific types of fields, there are different options for how a filter can be represented. For example date fields can be filtered on using date pickers to choose a range or plain text to represent distinct ranges available to filter on.

Most field values represented in an Appkit application are exposed as links which can be clicked on to allow a user to add a filter for that specific value. An example of this is that every slice of a pie chart visualization of the results is a clickable link which will filter to only those results which match the value represented by the slice of the pie chart.

Careful thought to filters and how best to represent them in a search application can improve the quality of the search experience and efficiency of the search.

Appkit tags

Appkit provides a range of client-side tags for building and executing search queries as well as displaying, formatting and visualization of search results.

These tags are packaged as AngularJS libraries divided into a number of namespaces. See the *Tag Reference* for full documentation on each of the tags.

Most Appkit tags can be used in various combinations and are designed to work together within the framework to deliver a high quality search experience for the user.

Common usage

This example demonstrates how the `result-list` tag works in line with our principles of *configuration over code* and *sensible defaults*.

To render a simple results list displaying every field in each document, this code is all that is required in the JSP:

```
<search:result-list response="response"></search:result-list>
```

If you want to go ahead and customize the output of a result list, you can easily do so along these lines:

```
<search:result-list response="response"> <search:result> <search:field  
name="title" style="title"></search:field> <search:field name="url" style="url"  
label="Link:"></search:field> <search:field name="description"  
style="description" label="Abstract:"></search:field> </search:result>  
</search:result-list>
```

This demonstrates a common pattern in Appkit of basic tags having default rendering to get you started, with the ability to customize the rendering of elements within them to a suitable level of granularity.

Configuration

Many tags have reasonable default configurations, but they can also pick up attributes from configuration files, avoiding the must change any JSP code:

Here is a platform tag which looks for properties in the configuration file at `src/main/resources/conf/platforms/myplatform/dev.conf`

```
<search:platform var="platform" conf="platform.myplatform.dev"></search:platform>
```

See section on *configuration files* for more information.

Architecture and Technology Stack

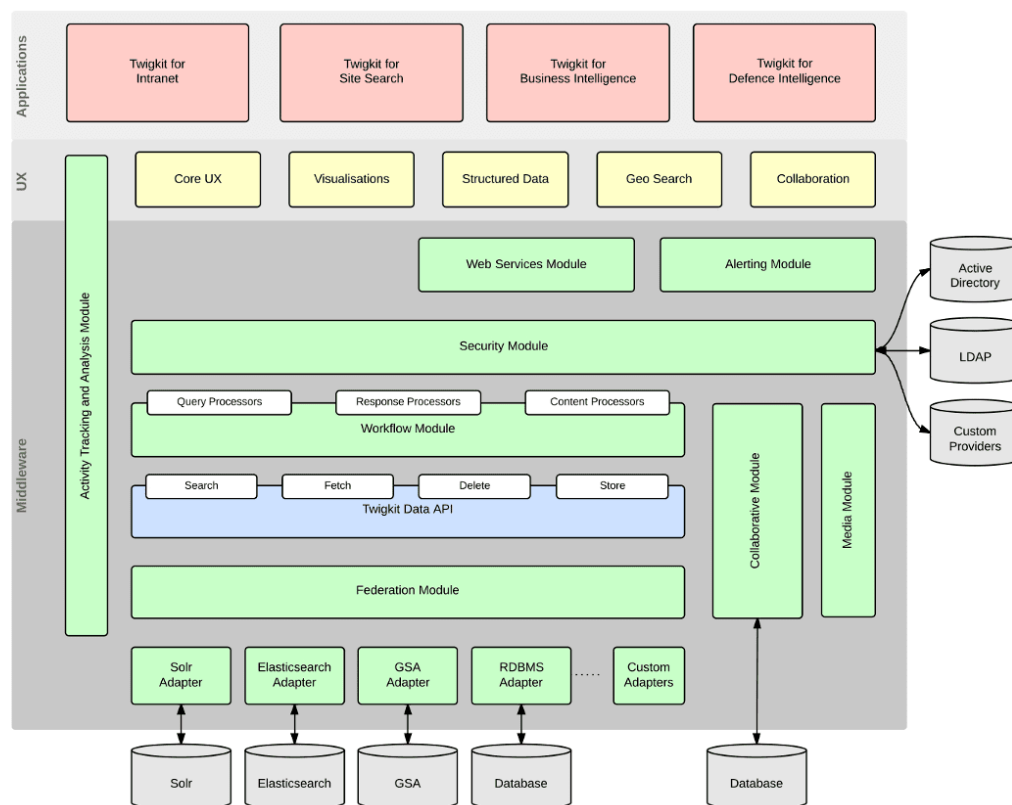
Appkit is a stateless, modular, 100% Java Standard Edition software framework.

Architecture

Appkit consists of:

- A layer of *data-abstraction middleware*
- A collection of functional *modules* for data manipulation
- *User interface components* for rendering and visualizing the data

An Appkit search application consists of modular building blocks, which are pieced together through the use of special *Appkit tags*, which can be mixed in with standard HTML and JSP code.



Technology stack

The Appkit stack is designed to be lightweight from the point of view of efficiency and ease of development. No programming or Java code is necessary to build a search application with Appkit.

Software that Appkit uses

Appkit uses:

- Maven for server side dependencies and build management
- npm for client side dependencies
- Standard Java Servlets and JSPs
- Standards-compliant AngularJS, HTML, CSS, and JavaScript

What Appkit provides

Appkit provides:

- A self-service application generator for project generation
- AngularJS based tag libraries for all framework components
- Web services to expose all search functionality
- Extension points for custom tags, JavaScript, and CSS

Deployment options

Appkit applications can be:

- Deployed locally in a development environment using the Maven Jetty plugin
- Built to a standard WAR file for deployment on Tomcat (or any other servlet container/application server)

Data Sources

Appkit can obtain and present information from multiple sources simultaneously, and seamlessly integrate with all common search engines, data warehouses, web service endpoints. The platform abstraction covers both the storage and sides, meaning that Appkit implements for each platform the necessary protocols to save and index content. A number of platform adapters are available out-of-the-box with Appkit, and *adding new ones* is a straightforward exercise.

Appkit is not limited to presenting information from search engines. It also supports data warehouses, web service endpoints, and other information sources that expose an API.

Platform adapters

The layer that relays Appkit specific instructions to the underlying data provider is referred to as as Platform Adapters. This is responsible for translating for example, an Appkit query to a corresponding command the underlying platform will understand; and conversely an engine specific response to a generic Appkit response.

The platform abstraction covers both search and storage sides, meaning that Appkit will implement for each platform the necessary protocols to save or index content for any platform.

A number of platform adapters are available out-of-the-box with Appkit, and adding new ones is a fairly trivial exercise.

Search engines

- *Fusion*
- *Solr*
- *SolrCloud*
- *Elasticsearch*

Other data sources

- *Wolfram|Alpha*
- *Wikipedia*
- *Pipl*
- *YouTube*
- *Twitter*
- *Google Custom Search*

Store, search, load, delete

Appkit (assuming storage backend capabilities) provides support for storing or indexing, deleting, fetching individual records, and searching the backing data store.

Task support

Appkit supports long-running, managed objects called *Tasks*. This is useful for operations that can take longer to complete such as Map/Reduce processes. In that case Appkit will manage the lifecycle, initiate the task, track and report progress, and finally present the response. After a response is received, it can be rendered using Appkit user interface components, or even passed to another platform for persistence which might give you low-latency searching and faceting.

Batch operation support

Appkit can cursor and retrieve entire indices by combining Task Support and our Batch Platform Wrapper which will automatically offset through any number of results from a platform and combine the responses.

Routing and URL Rules

This page explains the general URL patterns used in the application to map URI paths to JSP pages and resources within an Appkit application.

The URL routing is handled by the Tuckey URL rewrite filter (<http://tuckey.org/urlrewrite/>), and configured using this file: `src/main/webapp/WEB-INF/url-rules.xml`

This filter allows particular URLs to be served by specific JSPs, without forwarding or redirecting the user, thus the URL will remain the same in their browser. It provides a flexible way to define rules for mapping URLs in the application using regex (http://en.wikipedia.org/wiki/Regular_expression).

By default everything under the application context is passed through the filter, however most Appkit applications define a pattern which are passed straight through unmodified (essentially by-passing the filter):

```
<rule> <name>Ignore file, non-jsp and non-query requests</name> <condition
type="request-uri" next="or">^/twigkit/.*/</condition> <from>.*</from> <to
last="true">$0</to> </rule>
```

The remaining requests for URLs not matching the patterns to be ignored (shown above) will be processed through rules designed for the bulk of user requests such as search queries. A typical configuration for an Appkit application is shown below:

```
<rule> <from>^/login/</from> <to last="true">/login.jsp</to> </rule> <rule>
<from>^/login</from> <to last="true">/login.jsp</to> </rule> <rule match-
type="wildcard"> <name>Ignore files on URL path.</name> <from>/**.*</from> <to
last="true">/$0</to> </rule> <rule match-type="wildcard"> <name>HTML 5 URL
Rewrite</name> <from>/**</from> <to last="true">/index.jsp</to> </rule>
```

The first two rules routes all requests for '/login' or '/login/' to the login page. This will typically be the user's entry point to the application, though if security is enabled and the user is unauthenticated, any other page name given will direct the user to the login page by default.

The third rule matches all requests that have a period in the path (for example, 'mypage.html') and breaks the request.

The fourth rule will route all remaining requests to the index.jsp page, whereby the angular code will interpret the URL via the angular routing mechanism, and point to the correct view.

Each of these rules can be modified to support different patterns, depending on the requirements of the application. By default all GET request parameters, such as Appkit's *q* parameter representing the search query (**?q=my+search+terms**), will be passed straight through to the page that the request is routed to. Before modifying these rules, we recommend that you read the URL rewrite filter documentation (<https://cdn.rawgit.com/paultuckey/urlrewritefilter/master/src/doc/manual/4.0/index.html>), and specifically the section about the `urlrewrite.xml` configuration file (<https://cdn.rawgit.com/paultuckey/urlrewritefilter/master/src/doc/manual/4.0/index.html#configuration>), before modifying these rules.

Signals in Appkit

Appkit provides support for the *Fusion REST Signals API*, allowing user clicks and the creation of annotations (bookmarks, topics etc) to be recorded and tracked.

To learn how to use these signals with Appkit, see *Incorporating Fusion Signals in Appkit Applications*.

Creating a collection for signals tracking

Using the *Fusion UI*, when a primary data collection is created, this also creates a collection for the signals. By default, the name of this signals collection is by `<primary collection>_signals`, where `<primary collection>` should be substituted with the name of the primary data collection. After data has been indexed and ingested into the primary collection, the Appkit Fusion Signals module can be used to populate the signals collection.

Tracking user clicks and annotations

Currently, Appkit will send information on user clicks that trigger URL requests as well as when a user creates an annotation, for example, a bookmark or comment.

A signal event of type `click` will include, amongst other things, information about the URL that was clicked as well as the number of times that URL was clicked.

A signal event of type `annotation` will include, amongst other things, information about the annotation target, the collection, and the creator.

For a full list of properties that are stored, refer to the signals collection associated with your application.

Authentication

Activity tracking requires authentication. Both *service account impersonation* and the *Fusion Sessions API* can be used for this purpose.

Service account impersonation requires credentials for the service account and these are stored in the Fusion platform configuration. These credentials can be accessed by setting the name of the Fusion platform configuration via the `platform-config` parameter in the Fusion activity tracking configuration.

If the credentials for a service account are not provided or cannot be found, a check will be made to see if a Fusion session cookie is available. Typically, this will be present when the user is known to the Fusion server and the Session API has been set up accordingly.

Social and Collaboration Tools

The social and collaboration tools provided by the Social Module extend the search application with the ability to store user-generated information on individual documents and document collections, and to share this information among groups of authenticated users. For example, documents can be tagged or bookmarked, and users may save search queries and add comments to individual documents.

User-generated content can be stored in a relational database or any other kind of data store, using our pluggable architecture.

Authentication and authorization of users is managed via the Security module.

Terminology

While much of the language used to describe collaborative tools is carefully chosen to be self explanatory, there are several general concepts which are used throughout and are introduced in this section.

Annotations

Key to the collaboration enabled by inclusion of this module, there are several types of *annotation* which can be applied to documents. These include bookmarks, comments, and likes, which are commonly found in many applications and provide a versatile way for users to contribute and save information during their search.

Topics

A *topic* is a general term for a subject that user(s) are intending to collect information about. This module provides facilities for users to create a named group with which they can associate any number of annotated documents. Topics themselves can be created and selected using the *topic form tag*. Many of the collaborative tags included in this module have a *topic* attribute which is used to link the saved information to a specific topic (which can be picked up from the URL – more on this in the Topic pages module).

Topics can be defined by the developer as being 'namespaced' so they can be global (available for all users to collaborate on) or local (only viewable by the individual user that created them).

Collections

Some annotations can be categorized by *collection*. This is a mechanism for separating annotations of different types of document. There is a subtle difference between topics and collections, in that collections are usually determined by the categories of information in the data and generally pre-defined by the application developer offering a small number of collections with which to associate annotations. Collections are intended for a finite number of categorical groups whereas any number of topics can be created.

Combining social and collaboration tools

Topics and collections can even be combined such that a user can create a named 'bucket' to group information together, but have the saved information partitioned. One example is enabling the user to construct a wishlist of items that are automatically separated by category.

User Interface Components

Appkit has distilled years of research, design, and usability testing into a library of configurable user interface components responsively designed for both desktop and mobile.

These include support for:

- **Faceted navigation.** All common interaction modes, including text links and checkboxes, lists and hierarchies.
- **Advanced search.** Advanced search form components that utilize various matching strategies for deeper searching.
- **Suggestions.** As-you-type suggestions driven by a static list of terms, based on a field in the index, or powered by the platform.
- **Data visualizations.** From simple bar charts, to pie charts, to sophisticated area and line graphs for search-driven BI.
- **Geospatial mapping.** Plot search results on interactive maps from any provider, such as Google or Bing.
- **Search controls.** All the essentials you would expect: sorting, pagination, results per page, breadcrumbs, tabs.
- **Topic pages.** Search-driven topic pages using human, readable, "pretty" URLs, such as: `mywebsite.com/companies/amazon/`.

The Appkit framework removes the need for low-level code when developing search applications. Everything from establishing a connection to the search platform through to rendering results and facets can be granularly controlled via simple markup alongside regular HTML. Applications built using Appkit are **mobile-optimized** through responsive design and will correctly adapt to tablet and mobile sized devices.

In general, there is no must create a separate site for a mobile view, although that is also supported. Appkit provides an optimized experience for modern browsers (using HTML5 document types) but degrades gracefully and is functional back to IE11.

Applications can be fully **localized** (Java I18N), and our media module lets you normalize image size and resolution, on-the-fly in an efficient (cacheable) manner.

Configuration

The articles in this section describe how to configure a search application.

- *Basic Configuration*
Appkit gives you full flexibility to customize your search application, whilst providing sensible default settings out of the box. A typical search page consists of two parts: *preparation of the data model* (preparing a query, configuring one or more search platforms, and submitting the search query), and *presentation of the search results*. Both the data model and all UI components are configured through JSP tags, which allows for rapid prototyping and agile development.
- *Configuration Files*
- *Externalising Configuration Files*
A commonly asked question is whether you can obtain configuration files from a location external to the application, that is, somewhere else on the file system. The answer is yes.
- *Overlaying Configuration Values*
To fine-tune your configuration hierarchy prior to production, Appkit has added the ability to overlay configuration hierarchies via the Jetty Maven plugin. While developing an application, you can experiment with a partial configuration and add/update only those properties that you actually must update without touching the application's main configuration. When the application starts these changes are merged with the application's main configuration hierarchy.
- *Structure of a Web Application*

Basic Configuration

Appkit gives you full flexibility to customize your search application, whilst providing sensible default settings out of the box. A typical search page consists of two parts: *preparation of the data model* (preparing a query, configuring one or more search platforms, and submitting the search query), and *presentation of the search results*. Both the data model and all UI components are configured through JSP tags, which allows for rapid prototyping and agile development.

Preparing the model

Platform tags

Platform tags are used to configure a search platform, which is the service endpoint that you query to get back search results (a document result set). To configure each search platform requires a set of parameters, such as the host name of the search engine, which facets to retrieve, etc. These can either be defined inline in the JSP page (through attributes of the platform tags) or via central *configuration files*.

Since all platforms present the same interface to the end-user, you can either configure a platform using the *generic platform tag*, or the engine specific variants which will guide you in providing all the parameters they must function.

Query tags

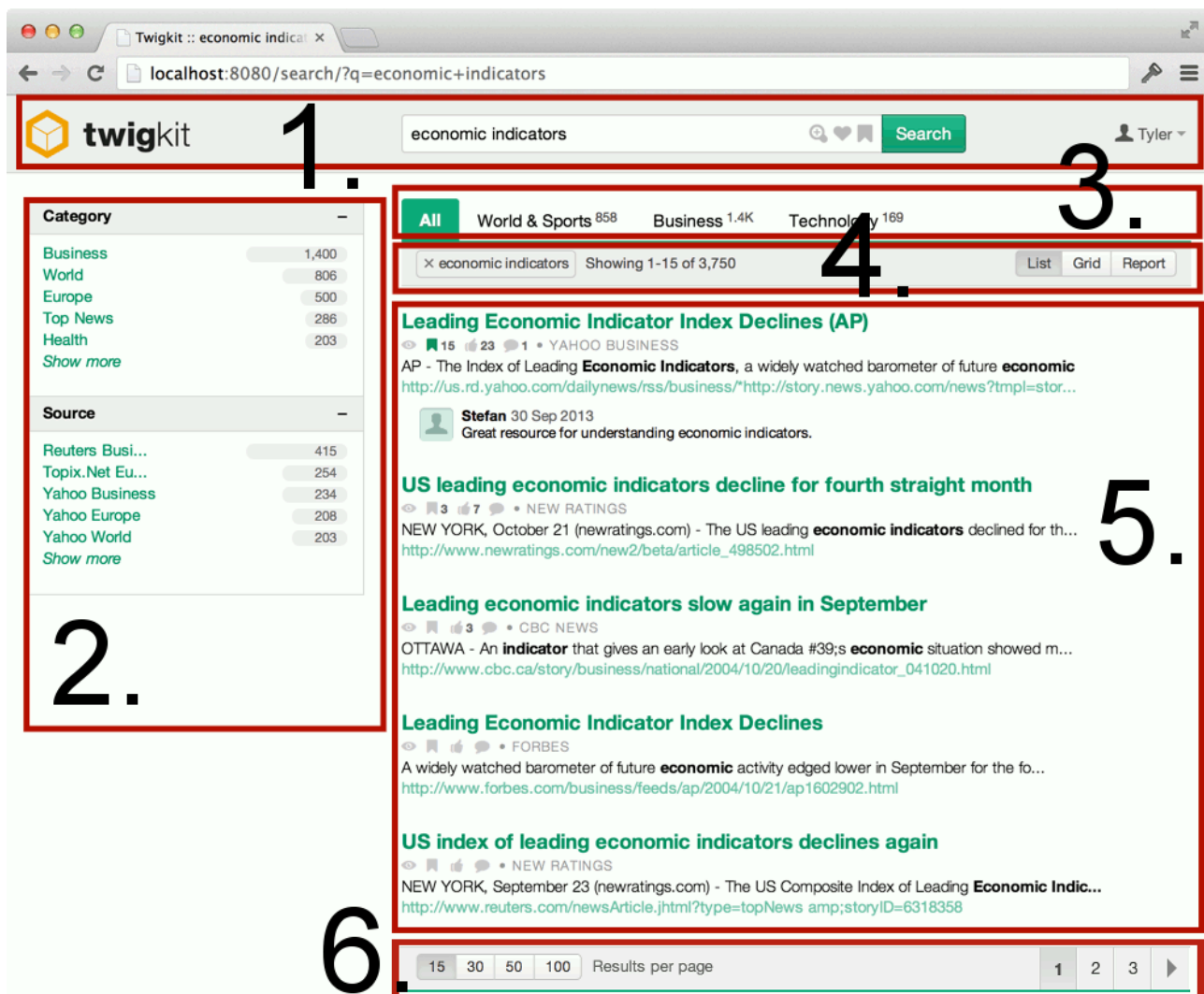
An Appkit query is made up of a number of query parameters. You can either create a query providing all the necessary parameters inline via the *Query tag* or by using query parameters submitted using HTTP request attribute. This, and the available parameters, are covered in detail on the *Query tag configuration tags page*.

Processing results on the fly

If the Workflow Module is installed, then you can use *Workflow Processors* to *modify queries* before they are submitted to the search engine, or alter the response before it is rendered.

Rendering search results

This figure shows a typical search page, with a mixture of faceted navigation, collaborative tools, and search controls. The composition of the page through atomic Appkit building blocks is outlined in detail below.



1. Search header

Most search pages will contain a header with the main search box which creates a starting point for a user to form their initial query.

Appkit search headers typically consist of:

- Logo and Branding
- Search box (with optional advanced search, bookmarks and saved search icons)
- Details of the current user including an avatar, username and user actions such as 'log out'

Associated tags:

- `<search:box>`
- `<security:user-actions>`

2. Faceted navigation

A staple of the search experience, faceted navigation allows a user to quickly refine their search, as well as giving an overview of the type of results matching their query.

Appkit provides great defaults for displaying facets, but can also be configured to allow for combining (OR'ing) filters within a category or removing all results with a particular value (adding NOT filters).

Associated tags:

- `<search:facetList />`
- `<search:facet />`

- `<layout:sidebar>`
- `<search:facet-list>`
- `<search:facet>`

3. Tabs

Tabs allow the user to navigate between divisions of the data which are typically based on separate types of data in the results.

Appkit's tab component provides an 'All' tab which is selected by default and provides a cross-search of everything. This feature can be disabled or renamed as required. Appkit also provides a count of the results matching the query next to each tab.

Associated tags:

- `<search:tabs>`

4. Breadcrumbs and controls

This horizontal bar presents information relating to the query, allowing the user to broaden their search and modify the display of results.

In an Appkit application, you will find 'breadcrumbs' allowing the user to see the search terms and filters applied and remove them to broaden the search. The number of search results shown on the page is displayed as well as the total number of matching hits. On the right side, a number of options might be provided, depending on the context of the application. These include the ability to sort the result list, change the formatting of the results, and display visualizations and charts associated with the results of the search.

Associated tags:

- `<search:breadcrumbs response="response" ...>`
- `<toggle:controls labels="List`
- `<search:sort fields="date=Date" query="query" ...>`
- `<search:response-statistics response="response" ...>`

5. Results

This is where a representation of the search results will appear. Appkit allows a high level of customization for the presentation of results while providing plenty of options for easily displaying common data types with a small amount of code.

Any number of the fields within the data can be presented with pre-configured styling for:

- Titles
- URLs
- Descriptions
- Inline field display
- Labelled fields
- Categories
- File formats and data types
- Dates
- Document previews

This is where links for actions such as bookmarking, likes and commenting can also be displayed.

There are a large number of tags for formatting results so we suggest referring to the documentation about *rendering search results*.

Secondary results

Often a search page will provide sections above or along-side the main results containing results from a different source. These help to give the user the option to navigate to a different type of data that might be more appropriate for the query they are trying to fulfill.

This can be implemented in a variety of ways within Appkit, see an example below of results pulled from the GSA (<http://www.google.com/enterprise/search/products/gsa.html>) platform OneBox feature:

The screenshot shows a web browser window with the URL `localhost:8080/content/?q=john`. The page features a Google search bar with the query 'john'. To the left is a sidebar with filters: Type (Athletes: 4, Venues: 4), Venue (Earl's Court: 1, ExCel: 1, Lord's Cricket Ground: 1, St James' Park: 1), Capacity (8001 - 12000: 1, 12001 - 20000: 1, 20000 - 50000: 1, 50002 - : 1), Legacy (+), Sport (+), and Location (Central London: 2, East London Office: 2, East London: 1, Islington: 1, Newcastle: 1, Newcastle Upon Tyne: 1, Windsor Castle: 1). The main content area is titled 'Content People' and shows 'Showing 1-15 of 20' results. It displays four profile cards for Dr J J McDermott, Dr G J Florence, Dr D J Culpin, and Dr T J Edwards. Below these are two travel information sections for Lord's Cricket Ground and St James' Park, both mentioning the 2012 Olympics and London 2012.

6. Search footer

The search footer usually contains *pagination* to allow the user to move between pages of results, and occasionally a control to choose how many results to display on each page.

The tags provided allow for configuration of parameters such as the maximum number of pages to display and the options provided for the results per page.

Associated tags:

- `<search:pagination response="response" ...>`
- `<search:results-per-page query="query" ...>`

Configuration Files

Configuration files

You can find the configuration files in `conf/` in the code editor, or in `src/main/resources/conf/` in the project directory. The configuration files are organized by function or by module, as follows:

```
/conf /activity /message /platforms /processors /security /services
```

These configuration files support hierarchical cascading of the files via the open-source Fig (<https://github.com/twigkit/fig/>) project.

This section describes the most important configuration files in the `/config` folder:

- `config/twigkit.conf`
Global configuration settings.
- `config/cors.conf`
Cross-Origin Resource Sharing (CORS) configuration.
- `config/platforms/fusion/fusion.conf`
Specify Fusion platform settings.
- `config/platforms/fusion/data.conf`
Specify which Fusion *query profile* to use.
- `config/security/fusion.conf`
Configure the Fusion security realm.

`config/twigkit.conf`

The App Studio application looks up global settings from a `twigkit.conf` configuration file that the application tries to locate relative to `config` on the runtime classpath.

Parameter	Type	Description
<code>license-file</code>	string	The path to the license file relative to the <code>app-studio/</code> directory, such as <code>file:///./twigkit.lic</code> .
<code>ui.trim-white-space</code>	string	"True" to trim white space in the application UI.
<code>bundleTimeToLive</code>	long	The number of milliseconds a resource bundle (for example, a file like <code>languages_en.properties</code>), can remain in the Java resource bundle cache without being validated against the source file from which it was constructed. The value 0 indicates that a bundle must be validated each time it is retrieved from the cache. If <code>bundleTimeToLive</code> is negative, or missing, then the bundle cache will have no expiration control (i.e. entries are only evicted from cache due to runtime or memory constraints).

`config/cors.conf`

CORS (<http://www.w3.org/TR/cors/>) or Cross-Origin Resource Sharing is a recent W3C (<http://www.w3.org>) effort to introduce a standard mechanism for enabling cross-domain requests from web browsers to servers that wish to handle them. App Studio supports CORS filtering by default. You can control the options for the particular header attributes using the `cors.conf` file at the root of the `/conf` folder.

Parameter	Type	Description	Default
<code>cors.allowOrigin</code>	string	Whitespace-separated list of origins that the CORS filter must allow. Requests from origins not included here will be refused with an HTTP 403 "Forbidden" response. If set to <code>*</code> (asterisk), any origin is allowed.	<code>*</code>
<code>cors.supportedMethods</code>	string	List of the supported HTTP methods. These are advertized through the <code>Access-Control-Allow-Methods</code> (http://www.w3.org/TR/cors/#access-control-allow-methods-response-header) header and must also be implemented by the actual CORS web service. Requests for methods not included here will be refused by the CORS filter with an HTTP 405 "Method not allowed" response.	<code>GET, POST, DELETE, HEAD</code>
<code>cors.supportsCredentials</code>	boolean	Indicates whether user credentials, such as cookies, HTTP authentication or client-side certificates, are supported. The CORS filter uses this value in constructing the <code>Access-Control-Allow-Credentials</code> (http://www.w3.org/TR/cors/#access-control-allow-credentials-respons) header.	<code>false</code>
<code>cors.maxAge</code>	integer	Indicates how long the results of a preflight request can be cached by the web browser, in seconds. If <code>-1</code> unspecified. This information is passed to the browser via the <code>Access-Control-Max-Age</code> (https://www.w3.org/TR/cors/#access-control-max-age-response-header) header.	<code>3600</code>

Hierarchical configuration

To make version control and specialization of configuration easier, the configuration is loaded hierarchically. For example, in the `/platforms` directory are subdirectories for `/fusion` and `/solr`:

```
/config /platforms /fusion /solr
```

Within the `/fusion` folder is `/fusion/fusion.conf`, which may contain the following general attributes:

```
name: twigkit.search.fusion.Fusion backwardsCompatible: true timeout: 30000
resultIDField: id highlight: true defaultQuery: *:*
```

Within the same folder, you might find one or more configuration files which inherit or extend this one, such as `data.conf` or `people.conf`. To access a given configuration, use dot notation, such as `platforms.fusion.data` or `platforms.fusion.people`. The configuration system will traverse the hierarchy (no matter how deep) and aggregate the configuration files, overwriting attributes from higher-level files when those attributes also appear in lower-level files.

For example, if `fusion.conf` contains a `defaultQuery` attribute and you create `internal.conf` which also contains a `defaultQuery`, then the value from `internal.conf` is used. This allows you to create variations on the same platform configuration. You can then refer to any platform instance in other configuration files or in the `search:platform` tag:


```
<search:platform conf="platforms.fusion.internal" />
```

In this case App Studio will use the platform as configured centrally, irrespective of the search engine behind it, effectively abstracting the data provider from the view.

Configuration locations

The configuration files can be placed outside of the application and centrally accessed by multiple instances. To reference these in a different location, set the `twigkit.conf` system property to the absolute file path of the `/conf` folder, like this:

```
-Dtwigkit.conf=file:///${path-to-your-conf-folder}
```

Adding special characters to key names in configuration files

To add special characters to key names in configuration files you need to escape the special characters. For example, if a key name contained a whitespace, such as `My key: value` then you would need to escape the whitespace for the configuration to be correctly loaded. In this case, the correct syntax would be:

```
My key: value
```

This rule does not just apply to whitespace but any special character that you may wish to use within a key name.

When you first install App Studio by unzipping the project archive, the `app-studio` directory contains `app`, `config`, `lib`, `resources`, and `build` subdirectories. The first time you start App Studio, an additional `logs` subdirectory is created.

Cookies

AppKit uses three cookies: `JSESSIONID` for authentication, and `tk.query.ctx_null` and `tk.session` to track session metadata.

- `JSESSIONID` is used by Spring Security for authentication and to maintain the session for the user
- `tk.query.ctx_null` stores the current query term to track whether the query is refined
- `appkitRequestCookie` is used to prevent Cross-Site Request Forgery (CSRF) when using social features

Externalising Configuration

A commonly asked question is whether you can obtain configuration files from a location external to the application, that is, somewhere else on the file system. The answer is yes.

The inset section under **Configuration locations** on this page describes how to do this, but more detail is provided in this FAQ page.

You must set the `twigkit.conf` variable when running the application server (for example, Tomcat or Jetty). This is done via an extra JVM argument on startup.

For Tomcat (which is the choice for QA & PROD deployments most of our clients use), the simplest way to do this is modify the `catalina.sh` script, adding this to the end of the line starting `JAVA_OPTS=`:

```
-Dtwigkit.conf=file://{path-to-qa-or-prod-conf-directory}
```

The path in curly braces must point to the location of a valid `conf` directory structure, as found in most Appkit applications in `src/main/resources/conf`. If you have separate QA and PROD servers, the path can remain the same and the contents of the files can differ depending on the environment.

Alternatively, without setting `twigkit.conf`, the configuration files from the WAR are used instead (from `WEB-INF/classes/conf`). You can use a feature like Maven build profiles to customize the WAR files that are built, however this requires more work and is only recommended if you are used to working in this manner or if you require a high degree of customization for each environment.

Overlaying Configuration Values

To fine-tune your configuration hierarchy prior to production, Appkit has added the ability to overlay configuration hierarchies via the Jetty Maven plugin. While developing an application, you can experiment with a partial configuration and add/update only those properties that you actually must update without touching the application's main configuration. When the application starts these changes are merged with the application's main configuration hierarchy.

Example

To illustrate this, consider this snippet of (default) configuration for an application that in this case is using the Google Search Appliance (GSA):

```
/src /main /resources /conf /platforms /gsa gsa.conf platforms.conf /services
/suggestions static.conf services.conf
```

Typically, this configuration hierarchy will contain settings relating to a production environment. However, while the application is being developed, you might want to use different values for some of the configurations or introduce new values to test whether they can be moved over to the application's configuration hierarchy.

To facilitate this workflow, you can now configure your own hierarchy, which by default, can be placed in `src/dev/resources/conf`. Now, say, you want to use a different GSA host while developing your application but did not want to change the application's default configuration. To do this you could create this configuration hierarchy:

```
/src /dev /resources /conf /platforms /gsa gsa.conf
```

As stated, you can use the default location for your own configuration hierarchy. However, this location can be overridden at the command line, if needed, when launching the application, for example:

```
mvn clean jetty:run -Dtwigkit.conf.overlay="file://src/dev/my_resources/conf"
```

You do not must fully populate your configuration hierarchy because this partial configuration will be merged with the application's configuration on application startup.

If the path to the user's configuration hierarchy cannot be found (for example, there is a mismatch between what is stated at the command line and what is actually in the project) then the application's default configuration hierarchy will be used.

Troubleshooting

If you are having issues loading your configuration hierarchy, try changing the logging level (in `src/main/resources/logback.xml`) to TRACE. Then look for entries in the log relating to the configuration hierarchy. Entries include `Secondary fig .. not found. Falling back to primary.` and likewise `Primary fig .. not found. Falling back to secondary.` The figs here refer to the application (primary) configuration hierarchy and your own (secondary) configuration hierarchy.

Structure of a Web Application

Appkit web applications follow a standard Java web project structure. If using Maven for build and dependency management, the application structure is as follows:

```
project-name/ `pom.xml` src/ main/ java/ resources/ conf/ webapp/ assets/ styles/  
WEB-INF/ pages/ tags/
```

If deploying directly in a servlet container such as Tomcat rather than using Maven, however, the application structure is simply:

```
project-name/ assets/ styles/ WEB-INF/ classes/ conf/ pages/ tags/
```

Below, we take a closer look at four areas of the application: `pom.xml`, the assets and styles folders, the WEB-INF folder, and the resources or classes folder.

pom.xml

The build process is managed by the `pom.xml` file. It contains references to the Maven repositories and version information for the dependencies. When the application is built (using Maven) all dependencies are automatically retrieved, and a web application archive (WAR file) is generated. Configuration files are automatically placed in the correct folders within the WAR.

Assets and styles folders

The assets folder is where assets such as images are kept. The main logos in a standard Appkit application are stored and served from here.

The styles folder contains CSS and Less files that are specific to the application. The central stylesheet for configuring styling is `classic-settings.less`, which in turn includes several partial stylesheets that are stored in the `theme/partials/` folder.

Custom styling using Less can be added to `classic-settings.less` and the partial stylesheets, and custom CSS to the `custom.css` file.

WEB-INF folder

The WEB-INF folder includes pages, tags, and a handful of configuration files. It also includes the standard Java web application `web.xml` configuration file.

Pages folder

The pages folder is where the JSP files are kept. `search.jsp` or `content.jsp` is typically the main search page. Additional pages can be created here as needed.

Tags folder

The tags folder stores `.tag` files – commonly referred to as "widgets" – that are specific to the particular application. They can then be referenced in the JSPs using the client: prefix. Some of the default widgets include:

- **header.tag.** this widget contains the search header including the logo and search box

You can modify the existing `.tag` files as well as add your own.

Configuration files

In addition to the Appkit specific *configuration files* under 'conf' there are also these web application configuration files:

- **url-rules.xml**. This file is where URL routing can be configured. There is a default rule in place that says the `/${foo}/` URL path should use the `WEB-INF/pages/${foo}.jsp` JSP page, hence `/search/` would be routed to the page `search.jsp`.
- **wro.properties**. The application uses wro4j to optimize CSS and JavaScript resources for delivery. wro.properties can be used to specify which resources should be bundled together. Wro packages all the resources when the application starts, as well as recompiling on the fly if a CSS or JavaScript file is modified.

Resources or classes folder

The resources folder (if running with Maven/Jetty), or `WEB-INF/classes` if deploying with Tomcat, is the primary location for configuration files in the application. The `spring-security.xml` file contains security configuration. The `platform-cache.xml` file defines how responses from the search platform should be cached, and `log4j2.xml` controls the logging level.

Finally, there is the `conf` folder that provides a cascading property framework. Properties can be set at a general level (such as 'platforms.gsa') and then overwritten more specifically (for example, 'platforms.gsa.people'). These property trees are reported in the log at startup and can then be referenced throughout the application.

Navigation

The articles in this section describe Appkit tags used for navigation.

- *Breadcrumbs*
The Breadcrumbs tag outputs the components of the Query for example, the query string and filters. If the original Query is provided to the tag via an attribute then HTML will automatically be generated to display and allow the removal of individual filters.
- *Faceted Navigation*
Faceted navigation is one of the most powerful features of search. Appkit provides simple tags for rendering lists of facets and filters in your application, while also providing a single widget that handles many of the complexities of faceted navigation for you.
- *Tabs*
The Tabs tag displays user interface components for tab-based navigation or filtering. The tag can be used to either output path-based tabs or filter tabs.

Breadcrumbs tag

The Breadcrumbs tag outputs the components of the Query for example, the query string and filters. If the original Query is provided to the tag via an attribute then HTML will automatically be generated to display and allow the removal of individual filters.

Usage

This example shows how to add breadcrumbs to the page. When no body is specified for the tag, Appkit will generate appropriate HTML.

```
<search:breadcrumbs response="response"></search:breadcrumbs>
```

For a complete list of attributes, see the `search:breadcrumbs` tag doc.

Faceted Navigation

Faceted navigation is one of the most powerful features of search. Appkit provides simple tags for rendering lists of facets and filters in your application, while also providing a single widget that handles many of the complexities of faceted navigation for you.

Listing facets

Self-closing the Facet List tag will output a list of all the facets and filter links with the default look-and-feel. *Find out more about the facetList tag.*

```
<search:facet-list response="response" facet-names="*"></search:facet-list>
```

You could then open the Facet List tag and place an individual Facet tag within it. *Find out more about the facet tag.*

```
<search:facet-list response="response" facet-names="*"> <search:facet facet-  
name="Author" collapsible="true"></search:facet> </search:facet-list>
```

This tag is also customisable to allow different display values and behavior. For a complete list of attributes, see the `search:facet-list` tag doc.

Tabs tag

The Tabs tag displays user interface components for tab-based navigation or filtering. The tag can be used to either output path-based tabs or filter tabs.

With path-based tabs, the tag outputs a fixed set of tabs that link to different URLs in the search application. This can be used to direct the user's search query between different views or collections, for example.

When displaying filter tabs, specific filters from a given facet are output even when the filter has no hits. This is often used for tab-based filtering, such as when the chosen facet partitions the search collections into different types of content. Be aware that when displaying filter tabs, this tag performs a second query to the search platform.

The Tabs tag subsumes the functionality of the older Filter Tabs tag.

Usage for path-based tabs

This example shows how to display a fixed set of tabs, each linking to a different search URL. In this case, the tag outputs three tabs, labelled 'Content', 'People' and 'All content', which link to `<contextPath>/content`, `<contextPath>/people` and `<contextPath>`, respectively. When clicking on each of the three tabs, the current search query (the `query` object) is taken into account and appended to the URL that is generated.

```
<search:tabs query="query" tabs="content=Content,people=People" all="true" all-label="All content"></search:tabs>
```

Usage for tab-based filtering

This example shows how to display filter tabs for a particular field. Filter based tabs are enabled by the addition of a `field-name="mediaType"` attribute, `mediaType` being the chosen field to filter on.

In this case both 'Videos' and 'Podcasts' filter will be grouped under the 'Multimedia' tab (using an OR query).

```
<search:tabs field-name="mediaType"
tabs="Articles,Blogs,Videos|Podcasts=Multimedia" all-label="Everything"
platform="platform" response="response" query="query" count-number="rounded">
</search:tabs>
```

For a complete list of attributes, see the `search:tabs` tag doc.

Rendering Search Results

Search results are the focus of every search interface, and Appkit makes it easy to forge the results precisely to your liking.

Result Lists

To add a list of results containing every available field, use a Result List tag on its own:

```
<search:result-list response="response"></search:result-list>
```

To choose which fields are displayed, add the Result tag inside the body of the Result List tag:

```
<search:result-list response="response"> <search:result> ... </search:result>
</search:result-list>
```

Working with fields

Inside the Result tag, use the Field tag to specify and configure each field:

```
<search:result-list response="response"> <search:result> <search:field
name="title" /> <search:field name="description" /> </search:result>
</search:result-list>
```

Results Table

The Table tag library is a collection of JSP tags for rendering search results in table format.

To display a list of search results in a table, add the `table:result-list` tag on your search page, passing in a reference to the `twigkit.model.Response` object that contains your search results:

```
<table:result-list response="response"> <table:header> <table:header-cell
name="full_name" label="Name" sort="true"></table:cell> <table:header-cell
name="address" label="Address" sort="true"></table:cell> <table:header-cell
name="dob" label="Date of birth" sort="true"></table:cell> </table:header>
<table:body> <table:row> <table:cell name="full_name" query="query"></table:cell>
<table:cell name="address"></table:cell> <table:cell name="dob" date-
format="dd/MM/yyyy"></table:cell> </table:row> </table:body> </table:result-list>
```


Cell tag

The Cell tag displays the value of a given field from a search result as a table cell. The Cell tag supports a number of features in displaying the value, such as date formatting, adding a suffix or prefix, and default values.

Usage

Place the Cell tag inside the body of a Row tag. The order of the cell tags within a row should match the order of the Header Cell tags within the table's Header tag. In this example, a table contains three columns and displays the "full_name", "address", and "dob" fields for each document in the given search response.

```
<table:result-list response="response"> <table:header> <table:header-cell  
name="full_name" label="Name" sort="true"></table:header-cell> <table:header-cell  
name="address" label="Address" sort="true"></table:header-cell> <table:header-  
cell name="dob" label="Date of birth" sort="true"></table:header-cell>  
</table:header> <table:body> <table:row> <table:cell name="full_name"  
query="query"></table:cell> <table:cell name="address"></table:cell> <table:cell  
name="dob" date-format="dd/MM/yyyy"></table:cell> </table:row> </table:body>  
</table:result-list>
```

For a complete list of attributes, see the `table:cell` tag doc.

Header Cell tag

The Header Cell tag outputs a single header cell in a results table. The tag must be nested within a Header tag.

Usage

Place the Header Cell tag inside the body of a Header tag. The order of the cell tags within a header should match the order of the Cell tags within the table's Row tag. In this example, a table contains three columns and displays the "full_name", "address", and "dob" fields for each document in the given search response.

```
<table:result-list response="response"> <table:header> <table:header-cell  
name="full_name" label="Name" sort="true"></table:header-cell> <table:header-cell  
name="address" label="Address" sort="true"></table:header-cell> <table:header-  
cell name="dob" label="Date of birth" sort="true"></table:header-cell>  
</table:header> ... </table:result-list>
```

For a complete list of attributes, see the `table:header-cell` tag doc.

Header tag

The Table Header tag outputs the header of the results table. The tag must be nested within a *Table Result List tag*.

Usage

Place the Header tag inside the body of a `/app-studio/4.0/3103/table-result-list-tag`[Table Result List tag]. It contains nested *Header Cell tags*, one for every table column.

```
<table:result-list response="response"> <table:header> <table:header-cell  
name="full_name" label="Name" sort="true"></table:header-cell> <table:header-cell  
name="address" label="Address" sort="true"></table:header-cell> <table:header-  
cell name="dob" label="Date of birth" sort="true"></table:header-cell>  
</table:header> ... </table:result-list>
```

Row tag

The Table Row tag specifies how rows of the results table should be formatted. The tag must be nested within a *Table Body tag*.

Usage

Place the Row tag inside the body of a *Table Body tag*. It contains nested *Cell tags*, one for every table column.

```
<table:result-list response="response"> ... <table:body> <table:row> <table:cell  
name="full_name" query="query"></table:cell> <table:cell name="address">  
</table:cell> <table:cell name="dob" date-format="dd/MM/yyyy"></table:cell>  
</table:row> </table:body> </table:result-list>
```

Table Body tag

The Table Body tag contains the main body of the results table. The tag must be nested within a *Result List tag*.

Usage

Place the Body tag inside the body of a *Result List tag*. It contains a *Row tag*, to specify the format of the table rows.

```
<table:result-list response="response"> ... <table:body> <table:row> ...  
</table:row> </table:body> </table:result-list>
```

For a complete list of attributes, see the `table:body` tag doc.

Table Result List tag

The Table Result List tag displays the results for a given search response in a table. The body of the Table Result List tag contains a *Header* tag which outputs the top header of the table, and a *Body* tag, which specifies the format of the rows of the table.

Usage

This example shows how to display all results in a search response in a table.

```
<table:result-list response="response"> <table:header> ... </table:header>
<table:body> ... </table:body> </table:result-list>
```

For a complete list of attributes, see the `table:result-list` tag doc.

Result List tag

The Result List tag displays the results for a given Response. While it can be used as a standalone tag to output every field for each result, it is most often contains a Result tag with numerous Field tags.

Usage

This example shows how to display all results returned by the Platform. It shows two alternative methods of referencing the fields of each item:

- Using angular syntax to use the value directly
- Using the Field tag to produce strictly valid XHTML with comprehensive CSS selectors.

```
<search:result-list response="response"> <search:result> <h2><a ng-href="{{result | field:'url' | actual}}" target="_top">{{result | field:'title' | actual}}</a>
</h2> <search:field name="body" label=""></search:field> <search:field
name="concepts" label="concepts" query="query" separator=";"></search:field>
</search:result> </search:result-list>
```

While not strictly required, the nested `<search:result>` tag is recommended. For more details, refer to the *Result tag*.

Nested Result Lists

Some search platforms, such as Solr (when using Grouping) will create Results with further nested Results. In Appkit terminology these are named 'related results'. A JOIN will produce a secondary set of hits for each result in the response. This secondary set of hits can be accessed and output using a nested Result List tag:

```
<search:result-list response="response"> <search:result> <search:field
name="country" label="" value-element="h2"></search:field> <search:field
name="text" label=""></search:field> <search:field name="climate" label="Climate"
query="query"></search:field> <h3>Cities in {{result | field:'title' | actual}}
</h3> <!-- Nested list of related Results --> <search:result-list
results="result.related"> <search:result> <search:field name="city" label="">
</search:field> </search:result> </search:result-list> </search:result>
</search:result-list>
```

See the `search:result-list` tag doc for all available attributes.

Tag body variables

The body of the tag will be repeated for every result in the Response.

`${result}` (twigkit.model.Result)

The result being output. See the Result for available properties.

- To get a String representation of a Field use: `{{result | field:'name' | actual}}` or `{{result | field:'title' | display}}`.

Result tags

The Result tag renders if results are available. It outputs a single result, and must be nested within a Result List tag. The No Results tag renders if no results are available.

Result tag

The Result tag renders if results are available. It outputs a single result, and must be nested within a Result List tag.

```
<search:result-list response="response"> <search:result> <search:field
name="title"></search:field> <search:field name="description"></search:field>
</search:result> </search:result-list>
```

See the `search:result` tag doc for all available attributes.

Conditional result tag

It is also possible to use the result tag to conditionally display a set of fields based on the value of a given field. This is useful when each result fit into a specific *category*.

Usage

The syntax is as follows:

```
<search:result-list response="response"> <search:result a-field-
name="fieldValue"> <search:field name="title"></search:field> <search:field
name="description"></search:field> </search:result> <search:result a-field-
name="otherFieldValue"> <search:field name="otherTitle"></search:field>
<search:field name="otherDescription"></search:field> </search:result>
</search:result-list>
```

For example, if you had 2 types of results, documents and people, with different fields, you could display them in a single result list as follows:

```
<search:result-list response="response"> <search:result type="people">
<search:field name="name"></search:field> <search:field name="date-of-birth">
</search:field> </search:result> <search:result type="document"> <search:field
name="title"></search:field> <search:field name="author"></search:field>
</search:result> </search:result-list>
```

It is also possible to combine multiple conditions. All have to be met for the specific `<search:result>` tag to be selected. For example:

```
<search:result-list response="response"> <search:result type="people"
country="Vietnam"> <h1>Vietnam</h1> <search:field name="name"></search:field>
<search:field name="date-of-birth"></search:field> </search:result>
<search:result type="people"> <h1>Rest of the World</h1> <search:field
name="name"></search:field> <search:field name="date-of-birth"></search:field>
</search:result> </search:result-list>
```

Extending Appkit

The Appkit application framework adheres to standard Java web application methodologies making custom development using generally accepted principles perfectly acceptable. In addition to that Appkit provides a number integration hooks for any aspect of the lifecycle and extension points for adding custom capabilities and features to Appkit application.

Fundamentals

The Appkit software stack is a 100% Java framework and most commonly deployed as a standard Java Web Application using best practice (and non-prescriptive) conventions. The framework is stateless with (in the case of web applications) all parameters of a Query generated from HTTP request parameters.

Appkit uses Google Guice for dependency injection and plugin architectures and Maven for dependency and build management. The Appkit software components are available via our private Maven repository: <http://repo.twigkit.com/>. Use the settings.xml (<https://dev.twigkit.net/downloads/settings.xml>) file to correctly resolve dependencies to the Appkit libraries.

Appkit SDK

The Appkit SDK is a fully configured, deployable software project with placeholders for the major components described above. For further information submit questions via the issue tracking system.

Collaboration Events

All social and collaboration events relating to annotations (likes, comments etc.) are published to a local event bus. This means that these events can be subscribed to and custom behavior can be implemented when the event is fired. For example, the custom behavior might be invoking some code that submits information to additional systems or that triggers a message such as an email notification.

To do this, create a listener for these events. Here is an example to do this for 'likes'. Add this code to `src/main/java/twigkit/events/AnnotationInterceptorModule.java`:

```
package twigkit.events; import com.google.common.eventbus.Subscribe; import
org.slf4j.Logger; import org.slf4j.LoggerFactory; import
twigkit.AbstractTwigKitModule; import twigkit.event.InjectorReady; public class
AnnotationInterceptorModule extends AbstractTwigKitModule { private static final
Logger logger = LoggerFactory.getLogger(AnnotationInterceptorModule.class);
public AnnotationInterceptorModule() { super(Priority.LOWEST); } @Override
protected void configure() { EventBus.register(new Object() { @Subscribe public
void injectorReady(InjectorReady event) { try { EventBus.register(new
AnnotationInterceptor()); } catch (Exception e) { } } }); logger.info("Annotation
Interceptor application module loaded."); } }
```

And also create this file in `src/main/java/twigkit/events/AnnotationInterceptor.java`:

```
package twigkit.events; import com.google.common.eventbus.Subscribe; import
org.slf4j.Logger; import org.slf4j.LoggerFactory; import
twigkit.TwigKitApplication; import twigkit.social.event.AnnotationCreated; import
twigkit.social.model.Annotation; import twigkit.social.model.Link; public class
AnnotationInterceptor { private static final Logger logger =
LoggerFactory.getLogger(AnnotationInterceptor.class); public
AnnotationInterceptor(){ } /** * * @param event social event picked from the
event bus */ @Subscribe public void annotate(AnnotationCreated event) {
Annotation annotation = event.getAnnotation(); if (annotation instanceof Link) {
logger.debug("This is a like annotation); //DO SOMETHING } } }
```

You must then register your new module by adding `twigkit.events.AnnotationInterceptorModule` to the file `src/main/resources/META-INF/services/twigkit.TwigKitModule`. If this file does not already exist simply create it and add the text `twigkit.events.AnnotationInterceptorModule`.

This is all that is required to implement custom behavior. Add the code you must the `annotate` method using the information provided in the `AnnotationCreated` event object. As with other custom modules registered in Appkit you can also inject references to platforms and other core Appkit components and interact with them here.

Developing Platforms

Platforms wrap the underlying search platform; creating a unified abstraction layer for major platform operations such as search, load, delete and store. You can add your own Platform implementation that wrap arbitrary data providers with a few simple lines of code and configurations.

Process Flow

At a high level the process flow for search operations is simple: A Query is submitted to a Platform, which it turns into a request to the underlying data repository. The Platform implementation then turns that into an Appkit Response and returns.

```
public class SomePlatform implements Platform { public Response search(Query query) { ... } }
```

Configurability

Typically the platform will require configuration and initialisation parameters, which can be passed on for example, using the Generic Platform tag. A simple way of achieving this with your platform is to extend the AbstractPlatform class:

```
package acme; public class SomePlatform extends AbstractPlatform { public Response search(Query query) { if (hasParameter("foo")) { String foo = getParameterStringValue("foo"); ... } } }
```

This means that the platform can be instantiated using the configuration framework or in the user interface JSP page with:

```
<search:platform className="acme.SomePlatform" foo="bar" />
```

For a more complex or flexible implementation, the process of adapting source queries and responses is typically encapsulated in distinct components, using an Adapter Pattern.

In this case a Platform implementation is injected with a Query Adapter and a Response Adapter which will handle the mapping of Appkit queries to underlying search platform queries (for example, a SolrQuery) and source responses (for example, a Solr QueryResponse) to an Appkit Response.

```
public class SomePlatform extends AbstractPlatform { @Inject public SomePlatform(@SomePlatform QueryAdapter<SomeQuery> queryAdapter, @SomePlatform ResponseAdapter<SomeResponse> responseAdapter) { ... } public Response search(Query query) { Response response = new Response(); SomeQuery sq = queryAdapter.map(this, query); ... Response response = responseAdapter.map(this, query, response, someResponse); return validate(query, response); } }
```


The `AbstractPlatform` also provides a comprehensive set of validation procedures for dealing with spelling suggestions, auto-corrections, and other aspects of the query/response lifecycle. Furthermore, the `twigkit.platform.AbstractQueryAdapter` and `twigkit.platform.AbstractResponseAdapter` classes help you to ensure that you map all of the critical components, so that the Appkit widgets can effectively work with the new data source.

After you have done this, your platform is a fully fledged member of the framework which will act and behave like any other supported platform, meaning that it will support the full range of features and can be plugged into any Appkit application, federated to, etc.

Deploying your Platform

You can refer to your platform by fully qualified class name in the Platform tag, as long as it is present on the application's classpath.

A more efficient method would be to package it so that it will be automatically instantiated and provided by Guice. To do that use for example, Maven to create a jar file for the platform, making sure this folder and file structure is included:

```
/META-INF/ /services/ twigkit.platform.Platform <-- this is a file
```

The `twigkit.platform.Platform` file should contain a line delimited list of fully qualified class names of all the platforms in your jar. Doing this means that on start-up Appkit will dynamically load all available platforms and make them available for use via Guice injection.

Developing Processors

The Appkit Workflow module can be used to apply custom business logic during the search/response/render lifecycle. Workflow Processors are designed to intercept any Appkit model object, and for example, modify a Query before it is submitted to a Platform or augment a Response before it is rendered.

Developing Processors

All processors implement the Processor interface, parameterized with the object they are designed to work with:

```
public class AcmeProcessor implements Processor<Query> { public void init() { ... } public void process(Query query) { ... } public void done() { ... } }
```

Or:

```
public class AcmeProcessor implements Processor<Response> { public void process(Response response) { ... } }
```

Processors also support the `change(T obj)` method which will let you substitute model objects: `public Response change(Response response);`

If you extend an `AbstractProcessor` class then `change()` is automatically named by the `process()` method making it sufficient to implement the former.

A number of abstract helper classes make it easier to implement processors for a given purpose, such as:

twigkit.processor.AbstractProcessor<T>

Provides abstract implementations of `init()` and `done()` and handles setting parameters.

twigkit.search.processors.AbstractResultProcessor

Makes it easy to implement processors for Results by taking care of the iterations

```
public void processResult(int index, Result result) {...}
```

twigkit.search.processors.AbstractNamedFieldProcessor

Makes it easy to implement processors that target specific fields. Requires the `fields` parameter (use `*` for all fields). Only the named fields will be passed on to the `process` method.

```
public void process(Field field) {...}
```

twigkit.search.processors.AbstractNamedFieldValueProcessor<T>

Use this abstract class to implement a Processor that will pass the named field values (where the value is an instance of `T`). Requires the `fields` parameter (use `*` for all fields). Validation of the field value being of the correct type is done by the abstract class.

```
public Value<String> processValue(Value<String> value) {...}
```

twigkit.search.processors.AbstractNamedFacetProcessor

Use this abstract class to implement Processor that will process specific named facets. Requires the `facetNames` parameter (use `*` for all facets).

```
public void processFacet(Facet facet) {...}
```

twigkit.search.processors.AbstractNamedFacetFiltersProcessor

Use this abstract class to process all the FacetFilters in the named facets. Requires the `facetNames` parameter (use `*` for all facets).

```
public void processFilter(FacetFilter filter) {...}
```

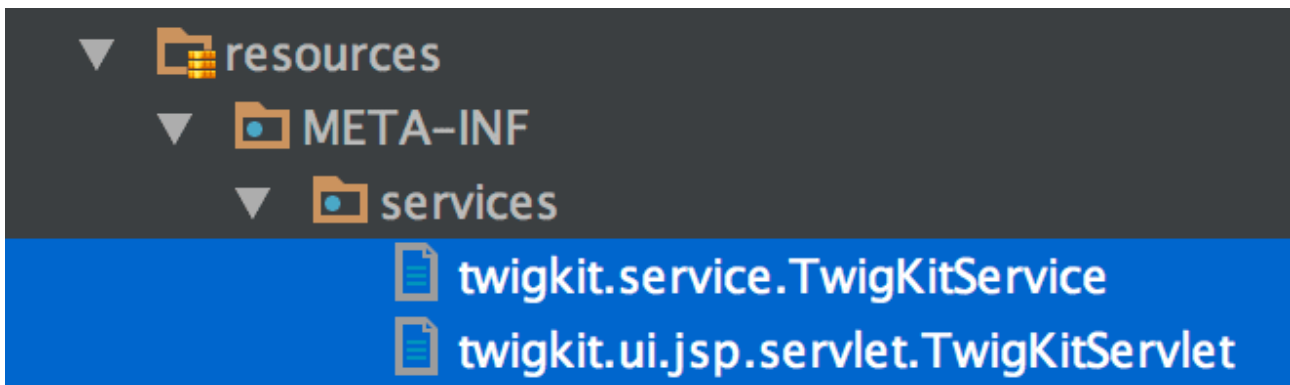
Developing Services and Servlets

You can develop services and servlets:

- Services encapsulate reusable processes that can be injected into providers such as Servlets. For example, different implementations of Query Suggestions are implement the `twigkit.service.AppkitService` interface and can be injected into the servlet responsible for returning the list of suggestions.
- Besides using standard JSP to generate output; Servlets are common way of doing custom HTTP responses with Java. You can deploy standard Java Servlets in Appkit application but in addition to that Appkit provides a convenient framework for developing and deploying servlets for search based applications.

Services

Services can be instantiated programmatically or using the Appkit configuration framework, making them available to other processes or programmatically via the `ConfiguredServiceProvider` class. To declare a service via the configuration framework, you must add a configuration file named `src/main/resources/META-INF/services/TwigKitService` that contains the fully-qualified Java class name of your services, one per line. See the screenshot for an example.

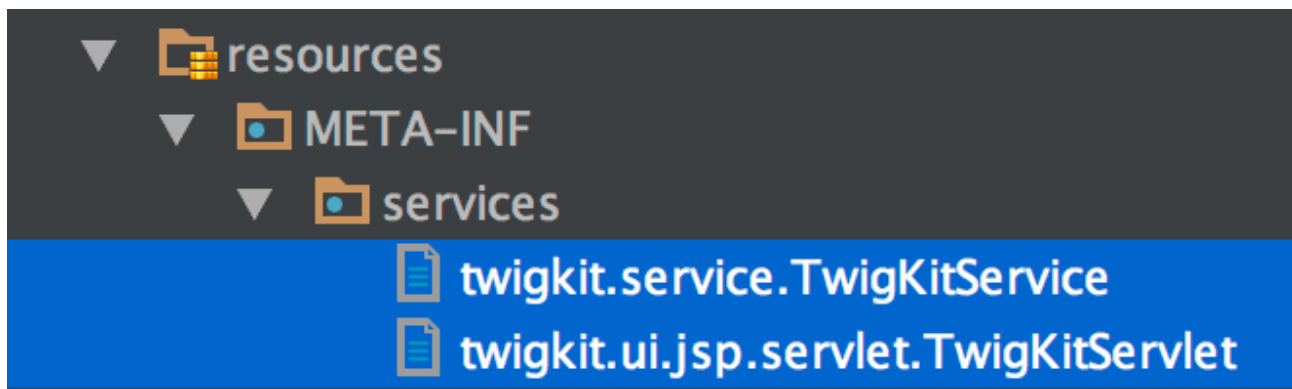


For legacy reasons, the service file must be named `TwigKitService`, with a capital "K". Below is a sample signature for a Query Suggestion Service:

```
public class StaticQuerySuggestionService extends Parameters implements
QuerySuggestionService, TwigKitService { public List<QuerySuggestion>
suggest(final Query query) { // Do whatever here to generate a List of
QuerySuggestions for the Query supplied! } }
```

Servlets and custom output formats

An instance of an `AppkitServlet` is automatically discovered and deployed by the Appkit application on startup, as long as your class is annotated with the path you would like it published at, implements or extends a `AppkitServlet`, and is declared via the Appkit configuration framework. To declare a servlet via the configuration framework, you must add a configuration file named `src/main/resources/META-INF/services/TwigKitServlet` that contains the fully-qualified Java class name of your servlets, one per line. See the screenshot for an example.



For legacy reasons, the service file must be named `TwigKitServlet`, with a capital "K". Below is an example of a `TwigKitServlet`.

```
@ServletPath(path = "/custom/excel/{platform}") public class SomeServlet extends
TwigKitServlet { protected void doGet(HttpServletRequest req, HttpServletResponse
resp, Query query, Config conf) throws ServletException, IOException { // Query
instances and Appkit configurations are automatically provided to your method! }
}
```

This means that the process of changing stateless HTTP requests to queries is handled for you, and the Servlet is automatically declared and published, in this case to respond at this URL: `/twigkit/services/custom/excel/`. Using the `ServletPath` annotation will guarantee that the HTTP Request/Response is correctly filtered by Appkit for example, for security and URL mapping.

Layout and Styling

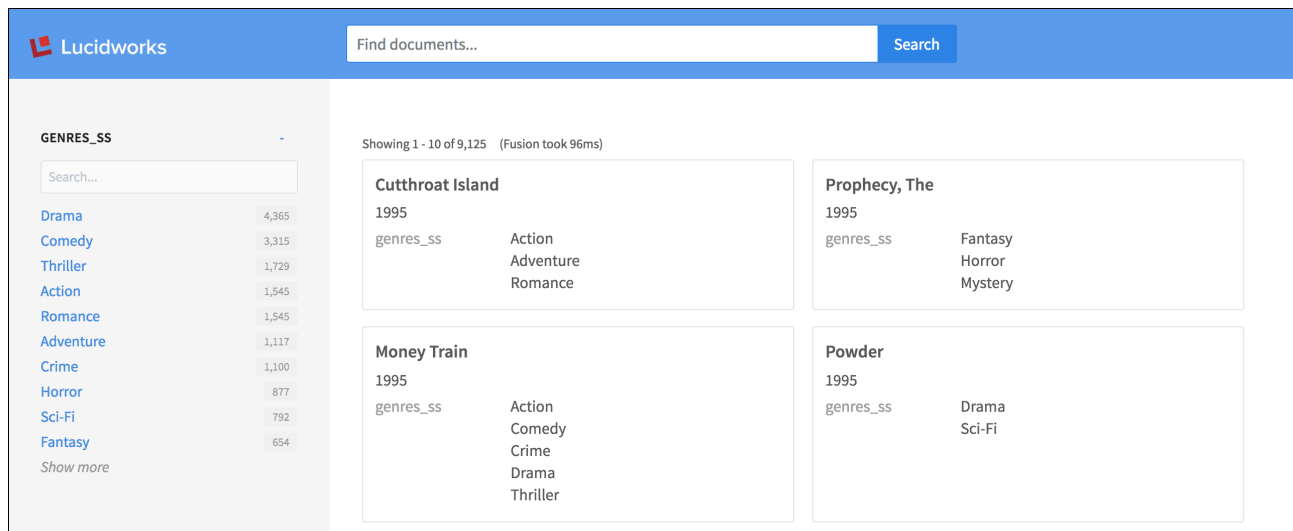
The articles in this section describe the basics of layouts and styling.

- *Layout of the Default View*
This topic explores the elements that compose the default App Studio view.
- *Themes and Styling*
Appkit is designed so that themes can be used to apply a different look and feel to an application. These encompass all visual aspects of a search user interface and currently there is one main *classic* theme which every project starts out with.
- *Topic Pages*
Appkit has advanced support for specifying custom URL patterns for filtering and grouping search results.

Layout of the Default View

This topic explores the elements that compose the default App Studio view.

The default view looks like this:

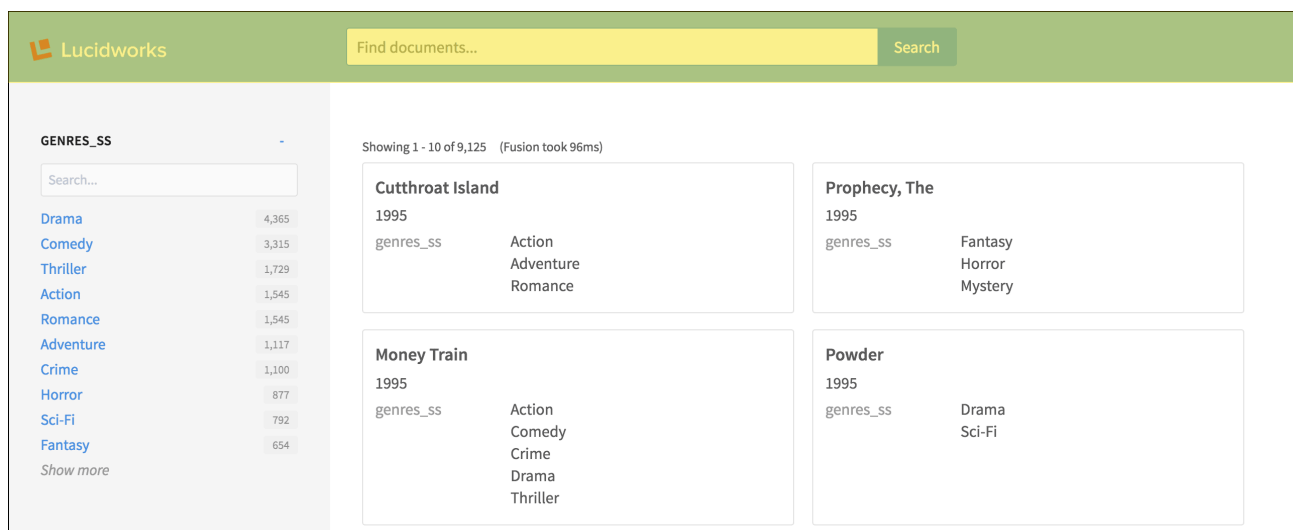


As we explore the components of the page, we will highlight each area in turn.

Include a partial file

The header of the default view is a partial file that is included with the `layout:include` tag:

```
<layout:include file="views/partials/header.tpl.html" action="search">
</layout:include>
```



The `header.tpl.html` file, in turn, contains a number of `layout` tags, plus the important `search:box` tag.

Sidebar with facet list

The sidebar is rendered using the simple `layout:sidebar` tag. Within the sidebar, the `search:facet-list` and `search:facet` tags contain the parameters that configure the facet

fields and their display.

Lucidworks

Find documents... Search

GENRES_SS

Search...

Drama 4,365
Comedy 3,315
Thriller 1,729
Action 1,545
Romance 1,545
Adventure 1,117
Crime 1,100
Horror 877
Sci-Fi 792
Fantasy 654
[Show more](#)

Showing 1 - 10 of 9,125 (Fusion took 96ms)

Cutthroat Island
1995
genres_ss Action
Adventure
Romance

Prophecy, The
1995
genres_ss Fantasy
Horror
Mystery

Money Train
1995
genres_ss Action
Comedy
Crime
Drama
Thriller

Powder
1995
genres_ss Drama
Sci-Fi

```
<layout:block md="1-3" lg="1-4" drawer="left" id="sidebar" styling="blocksidebar-light"> <layout:sidebar> <layout:box> <!-- List facets and give them intuitive labels --> <search:facet-list facet-names="genres_ss" response="response" platform="platform" query="query" styling="facet-list facet-list-wrappedheader"> <search:facet when-field="genres_ss" show="10" search-enabled="true" platform="platform" query="query" max-characters="40" show="12" show-more="24" collapsible="true"></search:facet> <search:facet show="10" max-characters="40" show="12" show-more="24" collapsible="true"></search:facet> </search:facet-list> </layout:box> </layout:sidebar> </layout:block>
```

Search statistics and results list

In the content area, `layout` tags are used again to construct the grid layout. For brevity, we will omit those from the code samples below.

Lucidworks

Find documents... Search

GENRES_SS

Search...

Drama 4,365
Comedy 3,315
Thriller 1,729
Action 1,545
Romance 1,545
Adventure 1,117
Crime 1,100
Horror 877
Sci-Fi 792
Fantasy 654
[Show more](#)

Showing 1 - 10 of 9,125 (Fusion took 96ms)

Cutthroat Island
1995
genres_ss Action
Adventure
Romance

Prophecy, The
1995
genres_ss Fantasy
Horror
Mystery

Money Train
1995
genres_ss Action
Comedy
Crime
Drama
Thriller

Powder
1995
genres_ss Drama
Sci-Fi

The `search:response-statistics` tag displays the number of search results, and a `{{ response.time }}` variable displays the number of milliseconds Fusion took to respond:

```
<search:response-statistics response="response"></search:response-statistics> <p class="response-statistics">(Fusion took {{ response.time }}ms)</p>
```

Then, the `search:result-list` tag lays out the grid of search results:

```
<search:result-list response="response" styling="cards-sm-1 cards-lg-2 cards-xl-2" platform="platform" query="query" instant-results="true"> ... </search:result-list>
```

Individual results

The `search:result` and `search:field` tags configure which fields are displayed with each individual search result. A field can be styled as the title or description, and any additional fields can have a variety of styles.

```
From Dusk Till Dawn
1996
genres_ss      Action
                Comedy
                Horror
                Thriller
```

```
<search:result> <search:field name="title_txt" styling="title" urlfield>
</search:field> <search:field name="year_i" styling="description" max-
characters="150"></search:field> <search:field name="genres_ss" styling="label-
left"></search:field> </search:result>
```

Null search results

In the case of no search results, the `search:no-results` tag displays a default message:

```
<search:no-results response="response"></search:no-results>
```

Showing all 0 (Fusion took 104ms)

xnxxnxxn ×

Sorry, there are no results that match your search criteria.

If you've checked your spelling, try using more general keywords.

Themes and Styling

Appkit is designed so that themes can be used to apply a different look and feel to an application. These encompass all visual aspects of a search user interface and currently there is one main *classic* theme which every project starts out with.

Themes can be controlled via the dependencies specified in the maven `pom.xml` file and will be available to select when a new project is generated through Launchpad.

Styling

Each of the themes uses a main Less file in `src/main/webapp/styles` named `themeName-settings.less`. This provides a basis for modifying the parameters of a theme at a high level such as the color scheme. If you are not familiar with Less or want to customize the UI by applying some standard CSS styling, this can be done in the `custom.css` file in the same location.

When to use Less and when to use CSS

In some cases, it is easier to customize particular components of the UI through the Less settings, rather than `custom.css`. In our themes and styling, Appkit defines a number of Less “mixins” which are re-usable code snippets that can be invoked using some number of parameters. The Less code is never exposed to the web client directly, but rather is compiled into CSS code at runtime (or even at compile time). When compiled, a single line of a `.less` file can result in multiple lines of CSS.

This means that in order to customize the default styling of a theme, you can:

- Modify a few lines in the `classic-settings.less` file
- Override the compiled CSS settings in the `custom.css` file.

This is best explained with an example. Consider a Less file with this “makeRed” re-usable code mixin:

```
.makeRed () { color: red; }
```

You can apply this mixin later by invoking it through a CSS rule; for example:

```
p { makeRed(); border: 1px black; }
```

which would output this in the compiled CSS file

```
p { color: red; border: 1px black; }
```

Looking at a particular Appkit example, consider for example, the way in which we specify the “title” field styling in the `classic-settings.less` file:

```
.tk-stl-title { .field-title(); } // @field-value-color, @field-label-color
```

Here, we use a `.field-title` mixin, which is defined in the classic theme. The comment at the end of the line lists the arguments that can be specified in this particular mixin. When compiled, the output CSS includes these rules:

```
.tk-stl-title .label { color: #000000; } .tk-stl-title .value { font-weight: bold; margin: 0 0 2px 0; font-size: 1.23em; line-height: 1.1667em; color: #52cc95; } .tk-stl-title .value a { color: #52cc95; }
```

To override the field title styling (`.tk-stl-title`), you could either change that one line in the Less file or include rules in `custom.css` overriding one or more of the compiled CSS rules, above.

So to conclude, it makes sense to override the Less settings when you are customising a part of the UI that has a well-defined mixin to match. For ad-hoc customization, it might be easiest to use `custom.css` directly.

Topic Pages

Appkit has advanced support for specifying custom URL patterns for filtering and grouping search results.

The Topic Pages module includes specialized versions of tags that work together to:

- Output links with a particular path structure according to a pattern
- Interpret the path of the current URL to apply filters to a search.

This allows for a rich user interface which can help a user make sense of highly 'categorized' data and map different views of the data to readable URLs.

See also *Add a Detail Page*.

Fusion App Studio Release Notes

Latest Release: 4.4.0 (27 September 2019)

Read the full release notes for version 4.4.0.

Component Versions

Fusion comes bundled with everything you need to get started (<https://lucidworks.com/try>).

Appkit SDK version in this release: 4.6.0

New features

There are no new App Studio features in this release.

Previous releases

4.2.1 (5 April 2019)

Read the full release notes for version 4.2.1.

Appkit SDK version in this release: 4.2.1

No new features are included in this release.

4.2.0 (28 February 2019)

Read the full release notes for version 4.2.0.

Appkit SDK version in this release: 4.2.0

New features in this release of App Studio come in through *the corresponding embedded release of the Appkit SDK*. * **Fusion collection for social data.** A new *Fusion auxiliary collection* called `_user_prefs` is now created automatically with each new Fusion app for the purpose of storing social data such as user tags, bookmarks, and so on.

- **Single package file.** A search app's two `package.json` files have been combined in a single file.

Previous files (pre-4.2.0):

```
/package.json /src/main/webapp/search/package.json
```

Current file (4.2.0):

```
/package.json
```

Older releases

- *App Studio 4.1.2*
- *App Studio 4.1.1*
- *App Studio 4.1.0*
- *App Studio 4.0.0*

App Studio 4.4.0 Release Notes

Release date: 27 September 2019

Appkit SDK version in this release: 4.6.0

See the *Appkit 4.6.0 release notes* for other changes.

In apps created with this version of Fusion App Studio, you can *upgrade the Appkit version to the latest version*. For a list of Appkit versions, see *the Appkit Release Notes*.

New features

There are no new App Studio features in this release.

Bug Fixes

- **Security configuration.** The Fusion security configuration (`/conf/security/fusion.conf`) is now generated using the correct realm parameter, `realm` instead of `realm-name`.
- **Compatibility with IE11.** Webapps created with App Studio are now compatible with Internet Explorer 11.

App Studio 4.2.1 Release Notes

Release date: 5 April 2019

Appkit SDK version in this release: 4.2.1

See the *Appkit 4.2.1 release notes* for other changes.

In apps created with this version of Fusion App Studio, you can *upgrade the Appkit version to the latest version*. For a list of Appkit versions, see *the Appkit Release Notes*.

New features

No new features are included in this release.

App Studio 4.2.0 Release Notes

Release date: 28 February 2019

Appkit SDK version in this release: 4.2.0

See the *Appkit 4.2.0 release notes* for other changes.

In apps created with this version of Fusion App Studio, you can *upgrade the Appkit version to the latest version*. For a list of Appkit versions, see *the Appkit Release Notes*.

New features

New features in this release of App Studio come in through *the corresponding embedded release of the Appkit SDK*.

Improvements

- **Fusion collection for social data.** A new *Fusion auxiliary collection* called `_user_prefs` is now created automatically with each new Fusion app for the purpose of storing social data such as user tags, bookmarks, and so on.
- **Single package file.** A search app's two `package.json` files have been combined in a single file.
Previous files (pre-4.2.0):

```
/package.json /src/main/webapp/search/package.json
```

Current file (4.2.0):

```
/package.json
```

App Studio 4.1.2 Release Notes

Release date: 17 December 2018

See also the *Fusion Server 4.1.2 release notes* and the *Fusion AI 4.1.2 release notes*.

New features

No new features were introduced in this release.

Improvements

- The `facet:dropdown` tag now supports multi-selection with a new `select` attribute that takes one of these values:
 - `toggle` (default)
Allow exactly one filter to be selected at a time, but show unselected filters and counts.
 - `multiselect`
Allow selection of multiple filters that are combined with a logical disjunction (OR). Show an "Apply" button at the bottom of the dropdown, which is enabled if and only if one or more checkboxes have been ticked.
 - `multiselect-instant`
Select multiple filters that are combined with a logical disjunction (OR). Filters are applied immediately when a checkbox is ticked.
- The `geo:highmaps` tag now support an `options` attribute that you can use to specify configuration options.
For a complete list of available options, see the Highmaps JS Options Reference (<https://api.highcharts.com/highmaps/>).

App Studio 4.1.1 Release Notes

Release date: 7 November 2018

New features

- New tags:
 - `advanced:text` creates an advanced search input field.
 - `collaborate:comment-form` renders a comment form.
 - `collaborate:comment-list` displays a list of comments.
- New service: `CollaborationService` accesses the Collaboration APIs.

Improvements

- `geo:map` tag improvements:
 - A new `drag` value for `events` changes the search range when the user drags the map; this works only with Google Maps.
 - The `draw` value for `events` now works with ESRI maps in addition to Google Maps.
 - The longitude filter is now created with the correct values.
- The `search:facet` tag can now be configured to display custom icons for expanded/collapsed facet lists using new `.collapsed` and `.facet-toggle` CSS classes. See the *CSS example*.
- The `instant:result-list` tag has a new `children-selector` attribute for selecting the child elements that should be used for the arrow key selection mechanism.

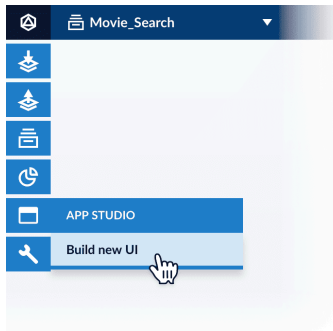
Other changes

- The `media:image` tag now respects the value of `image-service-url`, if defined.

App Studio 4.1.0 Release Notes

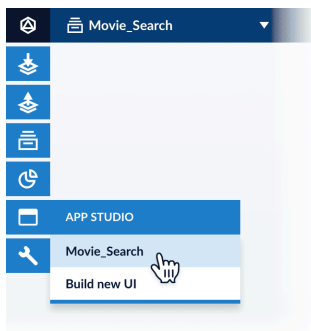
Release date: 17 July 2018

- The App Studio interface is now integrated with the Fusion Server interface. From the Fusion workspace, navigate to **App Studio > Build new UI**:



You will find a handy setup wizard, plus the embedded code editor, just as in App Studio 4.0.

After you have built one or more search interfaces with App Studio, you can access them from the same menu:



- An App Studio project can be *downloaded* for standalone development using your own tools.
- A search interface can now be deployed as a *self-contained application* or a *self-executing (.jar) application*, in addition to deployment as a Fusion Web app.

App Studio 4.0.0 Release Notes

Release date: 28 June 2018

This is the first public release of App Studio, compatible with Fusion Server 4.0.

Known issues

- Due to a known issue in Fusion Server 4.0.2, not all potential facet fields are not listed on the "Set Facets and Additional Fields" screen of the App Studio setup wizard.

Workaround:

In the *Query Workbench*, use the *Field Facet query pipeline stage* to add the desired facet field to the pipeline.

Appkit SDK Release Notes

Latest Release: 4.14.0 (31 October 2022)

Read the full release notes for version 4.14.0.

Improvements

- Swapped out dependencies on Log4j 1.2.x with SL4J in the Appkit code base.
- Fix for a bug that would prevent interval facets from toggling. Fix requires the `facet-list` tag to not use the `query=""` attribute.
- Fix for an issue when using the `<widget:spelling-suggestions>` component that would allow a cross-site scripting attack if the query term included a `<script>` tag and the Fusion `spellcheck` component returned a spell correction.

Previous releases

4.13.0 (15 September 2021)

Read the full release notes for version 4.13.0.

- Upgraded AngularJS framework to 1.8.2
 - Version 1.8.2 meteoric-mining corrects `$sceDelegate` to ensure that `resourceUrlWhitelist()` is identical to `trustedResourceUrlList()`.
- Upgraded JQuery to version 3.60.
 - Version 3.60 corrects security issues during a pen test.
- Fixed a bug where the selection of a `query:suggestion` component added the following to the URL: an `mq` parameter and an `mq` value surrounded by quotes.
- Fixed a bug where the facet filter for a facet already applied did not clear when `select=toggle`.
- Fixed a bug in the `search:box` with the `action` attribute where the queries on the search submission page were re-triggered *before* the page was redirected to the value listed in the `action` attribute.
- Fixed an issue where extra requests were made when a new query was submitted *if* a fixed header and infinite scroll were used.
- Fixed an issue to ensure a facet filter that contains special characters cannot be deselected.

4.11.0 (21 July 2020)

Read the full release notes for version 4.11.0.

- Ability to limit the results displayed by the `result-list` to a subset of the array
 - For example, the following: `<result-list response="response" first="2" last="5">` will display the 3rd, 4th and 5th results in the response.
- ES7 platform can now be configured whether or not to track total hits.
 - In the ES7 platform config (for example, `resources/conf/platforms/elastic/elastic.conf`) you can now add this config to track total hits: `track-total-hits: true`
 - By default, this will be **false** and will max out at **10000** as stated in the ES docs.
- It is now possible to specify whether to include or exclude the range filter boundaries by specifying the `fromExcluded` and `toExcluded` attributes to the `<query:filter>` tag.
 - For example:


```
<search:query parameters="*" var="query"> <query:filter field="age"
from="25" to="50" display="25 to 50" fromExcluded="true" toExcluded="true">
</query:filter> </search:query>
```

- Appkit no longer adds the apollo path prefix to Fusion API URLs. If still required, the path prefix may be set via configuration.
- Fixed a bug where `<search:response-statistics>` are only updated when its specified response object is updated, not on any others.
- Added support for Angular expressions as a value for the Bookmark-list `collection` parameter.
- `HighlightingUtils` adds highlighting terms support when closing braces are absent.
- Fusion pre-authentication filter now uses the JWT token subject field for the user name.
- In `collaborate:saved-query-list`, the `show-more` user click option now specifies the number of additional list items to display.

4.10.0 (27 April 2020)

Read the full release notes for version 4.10.0.

- A new attribute, `auto-populate`, is added to the `<collaborate:saved-query-list>` tag. When set to `true`, the saved search name is automatically pre-populated with the query term in the URL. The default value for this attribute is `false`.
- Fixed a bug that sometimes caused selected filters to be removed from a query after applying new filters on the same facet.
- Fixed a bug that sometimes returned invalid annotation IDs when using the social response processors, which prevented the end-user from further deleting the annotation (like a bookmark or saved-search).
- Specifying how many annotations, such as bookmarks or saved-searches, to retrieve using the Social API would not result in the expected behavior. This has now been fixed.

4.9.0 (27 September 2019)

Read the full release notes for version 4.9.0.

- The `FusionProxy` endpoint `/twigkit/api/fusion/*` now accepts nested URI paths as expected. For example:

```
/insights/twigkit/api/fusion/collections/<collection_name>
/insights/twigkit/api/fusion/apps/<app_name>/experiments
```

Previously, an HTTP 400 error occurred.

- Removing filters using the `<query:filter action="remove">` tag will no longer remove filters added with the `<query:filter value="bar">` tag. For example, the following will leave the filter on `value="bar"` intact:

```
<search:query var="query"> <query:filter field="foo" value="bar"> <query:filter
field="foo" action="remove"> </search:query>
```

- Previously, using the Solr platform in compatibility mode forced exclusive boundaries on date range filters to become inclusive. This affected date range filters in the Fusion platform.
This behavior is fixed. Exclusive boundaries on range filters now remain exclusive in compatibility mode.
- Toggle events now fire as expected when using toggle controls.

4.8.0 (10 January 2020)

Read the full release notes for version 4.8.0.

- It is now possible to map a user's LDAP roles to the application user's group. This differs from the previous implementation to the AD pre-authorization filter, in that it reads the groups directly from the user object instead of querying for AD groups that the user belongs to.
To enable this filter, add a configuration file under `conf/security/ldap/provider.conf` with the following parameters:

```
enabled: true user: #Service account used to connect to LDAP. password:
#Password for the service account. This can be encrypted using the twigcrypt
binary. url: #Full URL to the LDAP server. user-search-base: #LDAP search base
to restricted the search to. user-search-filter: #LDAP search filter to
restrict the LDAP search.
```

- It is now possible to use basic authentication against a Solr cloud platform using the parameters `userName` and `password` in the platform configuration.
- The value of a highlighted field is now stored as the "display" field value, rather than replacing the original "actual" field value.
- Appkit 4.7.0 adds support for Elastic 7.4.
- A new filter, `find`, is added which searches the scope hierarchy to find a specified variable. For example: `{{ 'myVariable' | find:this }}`.
- The `twigkit.http.filter.DevelopmentErrorFilter` is replaced by the `twigkit.http.filter.ErrorFilter`. If you are using the `twigkit.http.filter.ErrorFilter` parameter in your `web.xml` file and an AJAX request fails, a JSON response is received identifying the status code of the REST call and any underlying causes. Previously, an HTML response was received.
Example configuration:

```
<filter> <filter-name>ErrorFilter</filter-name> <filter-
class>twigkit.http.filter.ErrorFilter</filter-class> <init-param> <param-
name>responseCode</param-name> <param-value>400,405,406,500,501,503</param-
value> </init-param> <init-param> <param-name>excludedPaths</param-name>
<param-value>/,/login,/login/</param-value> </init-param> </filter>
```

- Accessibility improvements are made to the *search breadcrumbs*. The text, `x`, that removed the breadcrumb is replaced by an icon with the alt-text "Remove this term from the query" for use by screen readers.

type ×

man ×

date ×

8/5/16 - 8/10/16 ×

numbers ×

100 - 200 ×

Clear all

Older releases

- *Appkit SDK 4.7.0.*
- *Appkit SDK 4.6.0*
- *Appkit SDK 4.5.0*
- *Appkit SDK 4.4.0*
- *Appkit SDK 4.3.0*
- *Appkit SDK 4.2.5*
- *Appkit SDK 4.2.4*
- *Appkit SDK 4.2.3*
- *Appkit SDK 4.2.2*
- *Appkit SDK 4.2.1*
- *Appkit SDK 4.2.0*

Appkit SDK 4.14.0 Release Notes

Release date: 31 October 2022

Additional upgrade steps

Upon upgrading, users should remove `src/main/resources/logback.xml` and create a new file called `simplelogger.properties` in the same directory with the following content:

```
# org.slf4j.simpleLogger.defaultLogLevel=debug
org.slf4j.simpleLogger.showDateTime=true
org.slf4j.simpleLogger.dateTimeFormat=yyyy-MM-dd HH:mm:ss:SSS Z
#org.slf4j.simpleLogger.log.my.class.path=DEBUG
org.slf4j.simpleLogger.log.twigkit=DEBUG
```

Going forward, the `simplelogger.properties` file will be used to configure additional logging. The default logging level is `INFO`, but can be changed using the `defaultLogLevel` property. Logging on a particular class can also be increased by adding a new line, such as `org.slf4j.simpleLogger.log.my.class.path=DEBUG`, changing `my.class.path` to the actual path, and setting the logging level as needed.

If your `pom.xml` file contains the following entry, remove it:

```
<dependency> <groupId>ch.qos.logback</groupId> <artifactId>logback-
classic</artifactId> <version>1.1.1</version> </dependency>
```

Improvements:

- Swapped out dependencies on Log4J 1.2.x with SL4J in the Appkit code base.

Bug Fixes

- Fix for a bug that would prevent interval facets from toggling. Fix requires the `facet-list` tag to not use the `query=""` attribute.
- Fix for an issue when using the `<widget:spelling-suggestions>` component that would allow a cross-site scripting attack if the query term included a `script` tag and the Fusion `spellcheck` component returned a spell correction.

Appkit SDK 4.13.0 Release Notes

Release date: 15 September 2021

The 4.13.0 release includes items from Appkit SDK 4.12.0, which is not a separate release.

Improvements:

- Upgraded AngularJS framework to 1.8.2
 - Version 1.8.2 meteoric-mining corrects `$sceDelegate` to ensure that `resourceUrlWhitelist()` is identical to `trustedResourceUrlList()`.
- Upgraded JQuery to version 3.60.
 - Version 3.60 corrects security issues during a pen test.

Bug Fixes

- Fixed a bug where the selection of a query:suggestion component added the following to the URL: an `mq` parameter and an `mq value` surrounded by quotes.
- Fixed a bug where the facet filter for a facet already applied did not clear when `select=toggle`.
- Fixed a bug in the search:box with the action attribute where the queries on the search submission page were re-triggered *before* the page was redirected to the value listed in the action attribute.
- Fixed an issue where extra requests were made when a new query was submitted *if* a fixed header and infinite scroll were used.
- Fixed an issue to ensure a facet filter that contains special characters cannot be deselected.

Appkit SDK 4.11.0 Release Notes

Release date: 21 July 2020

Improvements:

- Ability to limit the results displayed by the result-list to a subset of the array
 - For example, the following: `<result-list response="response" first="2" last="5">` will display the 3rd, 4th and 5th results in the response.
- ES7 platform can now be configured whether or not to track total hits.
 - In the ES7 platform config (for example, `resources/conf/platforms/elastic/elastic.conf`) you can now add this config to track total hits: `track-total-hits: true`
 - By default, this will be **false** and will max out at **10000** as stated in the ES docs.
- It is now possible to specify whether to include or exclude the range filter boundaries by specifying the `fromExcluded` and `toExcluded` attributes to the `<query:filter>` tag.
 - For example:

```
<search:query parameters="*" var="query"> <query:filter field="age"
from="25" to="50" display="25 to 50" fromExcluded="true" toExcluded="true">
</query:filter> </search:query>
```

- Appkit no longer adds the `apollo` path prefix to Fusion API URLs. If still required, the path prefix may be set via configuration.

Bug Fixes

- Fixed a bug where `<search:response-statistics>` are only updated when its specified response object is updated, not on any others.
- Added support for Angular expressions as a value for the Bookmark-list `collection` parameter.
- `HighlightingUtils` adds highlighting terms support when closing braces are absent.
- Fusion pre-authentication filter now uses the JWT token subject field for the user name.
- In `collaborate:saved-query-list`, the `show-more` user click option now specifies the number of additional list items to display.

Appkit SDK 4.10.0 Release Notes

Release date: 27 April 2020

Improvements

- A new attribute, `auto-populate`, is added to the `<collaborate:saved-query-list>` tag. When set to `true`, the saved search name is automatically pre-populated with the query term in the URL. The default value for this attribute is `false`.

Bug Fixes

- Fixed a bug that sometimes caused selected filters to be removed from a query after applying new filters on the same facet.
- Fixed a bug that sometimes returned invalid annotation IDs when using the social response processors, which prevented the end-user from further deleting the annotation (like a bookmark or saved-search).
- Specifying how many annotations, such as bookmarks or saved-searches, to retrieve using the Social API would not result in the expected behavior. This has now been fixed.
- Previously, applying the dynamic Solr date range query processor to multiple facets simultaneously in the same configuration resulted in the same gap being applied to all configured facets. This occurred independently of the preexisting filters applied to individual facets, which could lead to inconsistent date range bucketing. This is now fixed.

Appkit SDK 4.9.0 Release Notes

Release date: 10 February 2020

Bug Fixes

- The `FusionProxy` endpoint `/twigkit/api/fusion/*` now accepts nested URI paths as expected. For example:

```
/insights/twigkit/api/fusion/collections/<collection_name>  
/insights/twigkit/api/fusion/apps/<app_name>/experiments
```

Previously, an `HTTP 400` error occurred.

- Removing filters using the `<query:filter action="remove">` tag will no longer remove filters added with the `<query:filter value="bar">` tag. For example, the following will leave the filter on `value="bar"` intact:

```
<search:query var="query"> <query:filter field="foo" value="bar"> <query:filter  
field="foo" action="remove"> </search:query>
```

- Previously, using the Solr platform in compatibility mode forced exclusive boundaries on date range filters to become inclusive. This affected date range filters in the Fusion platform. This behavior is fixed. Exclusive boundaries on range filters now remain exclusive in compatibility mode.
- Toggle events now fire as expected when using toggle controls.

Appkit SDK 4.8.0 Release Notes

Release date: 10 January 2020

New features

- It is now possible to map a user's LDAP roles to the application user's group. This differs from the previous implementation to the AD pre-authorization filter, in that it reads the groups directly from the user object instead of querying for AD groups that the user belongs to.
To enable this filter, add a configuration file under `conf/security/ldap/provider.conf` with the following parameters:

```
enabled: true user: #Service account used to connect to LDAP. password:
#Password for the service account. This can be encrypted using the twigcrypt
binary. url: #Full URL to the LDAP server. user-search-base: #LDAP search base
to restricted the search to. user-search-filter: #LDAP search filter to
restrict the LDAP search.
```

Improvements

- It is now possible to use basic authentication against a Solr cloud platform using the parameters `userName` and `password` in the platform configuration.
- The value of a highlighted field is now stored as the "display" field value, rather than replacing the original "actual" field value.
- UTC dates sent from the browser are now retained as UTC when passed to Solr.

Bug Fixes

- All Solr Stats components are now mapped to the Appkit response object. See The Stats Components (https://lucene.apache.org/solr/guide/7_5/the-stats-component.html#statistics-supported) for more information.
- In provided templates, some of the CSS and JavaScript files under `/twigkit/api/static/**` had the wrong content type. This prevented some browsers from executing the files properly. This is fixed.
- Fixed an issue that sometimes caused the session cookie used for activity tracking to reset and be incorrectly assigned.
- Fixed an issue in the `RoleBasedAuthorizationFilter` that sometimes resulted in a Null Pointer Exception.
- Fixed an issue that sometimes prevented highlighting of multivalue fields.
- Previously, request signals generated by App Studio did not specify whether a filter was excluded or not (a negative filter). Now, filter metadata represents excluded filters with a `-` symbol in front of the filter value.
- Fixed an issue with the `<search:box>` tag that caused the query term used in the search box to differ from the value of the query associated with the tag.

Appkit SDK 4.7.0 Release Notes

Release date: 22 November 2019

New features

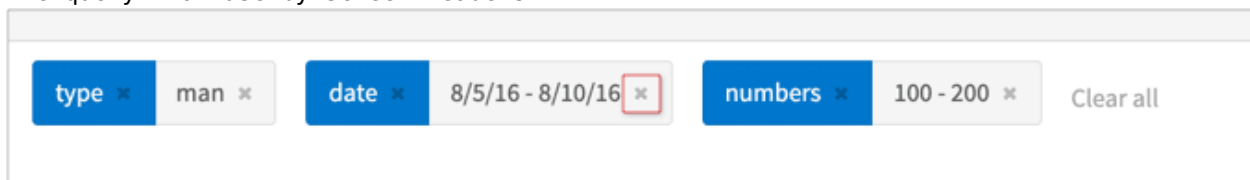
- Appkit 4.7.0 adds support for Elastic 7.4.
- A new filter, `find`, is added which searches the scope hierarchy to find a specified variable. For example: `{{ 'myVariable' | find:this }}`.

Improvements

- The `twigkit.http.filter.DevelopmentErrorFilter` is replaced by the `twigkit.http.filter.ErrorFilter`. If you are using the `twigkit.http.filter.ErrorFilter` parameter in your `web.xml` file and an AJAX request fails, a JSON response is received identifying the status code of the REST call and any underlying causes. Previously, an HTML response was received.
Example configuration:

```
<filter> <filter-name>ErrorFilter</filter-name> <filter-  
class>twigkit.http.filter.ErrorFilter</filter-class> <init-param> <param-  
name>responseCode</param-name> <param-value>400,405,406,500,501,503</param-  
value> </init-param> <init-param> <param-name>excludedPaths</param-name>  
<param-value>/,/login,/login/</param-value> </init-param> </filter>
```

- Accessibility improvements are made to the *search breadcrumbs*. The text, `x`, that removed the breadcrumb is replaced by an icon with the alt-text "Remove this term from the query" for use by screen readers.



- For security purposes, any existing incoming `userID` value is removed and replaced when impersonation is enabled.
- When a user enters incorrect login credentials, the username is preserved in the form and the URL for the user to verify.
- The UI now sends a `X-Requested-With=XMLHttpRequest` HTTP header with every API request. This makes it easier for the middleware to detect XHR requests and handle errors appropriately.
- Previously with the `ApplyDynamicSolrDateFormattingResponseProcessor`, the same time format was applied to days, hours, minutes, and seconds. Now, these values have their own formats.
- When using the `twigkit.search.processors.response.ApplyDynamicSolrDateFormattingResponseProcessor` and `twigkit.search.processors.query.DynamicSolrDateRangeQueryProcessor`, a new facet value, `gap`, is added to the date facet. This value of this facet is the unit of the gap size that was applied.

Other changes

- The default session timeout is updated from 10 minutes to 1 hour, to align with Fusion 4.2.x.

Addressed Security Vulnerabilities

- Upgrades to Jackson 2.9.10 (<https://github.com/FasterXML/jackson/wiki/Jackson-Release-2.9.10>) and Apache Commons BeanUtils 1.9.4 (<http://commons.apache.org/proper/commons-beanutils/javadocs/v1.9.4/apidocs/index.html>) are made to address CVE vulnerabilities.

Bug Fixes

- Fixed a bug that occasionally left a filter styled as being selected after another filter was chosen, despite the query results changing to reflect the new selection.
- Fixed an issue that would occasionally let a response object from one view be used by another view.
- Upgraded the JFrog protocol to `https` to prevent a `[FATAL] Non-resolvable parent POM` error.
- Fixed a bug that occasionally caused "like" signals to generate the wrong annotation type metadata.
- Fixed a bug with `FacetFilterValueMappingProcessor` mappings which prevented commas from being used with the value. Additionally, two parameters are introduced, `separator` and `key-value-separator`. For example:

```
separator: ; // Defaults to "," key-value-separator: _ // Defaults to "="
```

Using `key-value-separator: _` will separate the following key value pairs properly: `alpha,beta=greek_a,b=english`.

- Fixed a bug that rarely caused a Null Pointer Exception (NPE) error when a workflow platform was reading an underlying platform parameter.

Appkit SDK 4.6.0 Release Notes

Release date: 27 September 2019

Upgrading to this version

In apps created with earlier versions of Fusion App Studio, you can *upgrade the Appkit version to 4.6.0*.

Additional upgrade steps

If you have previously specified `requestHandler: /` in the platform suggestions configuration files of your apps, you can now remove this parameter.

In earlier versions, `requestHandler: /` was used to allow search term suggestions for apps using a query profile, but this parameter is no longer needed.

For example:

```
# Configure a Platform for Solr Query Suggestions query-profile:
YOUR_QUERY_PROFILE requestHandler: /
```

Upgrading from Appkit versions prior to 4.4

If you are upgrading from a version of Appkit prior to 4.4 then you may need to follow additional steps described in previous Release Notes.

New features

No new features are included in this release.

Improvements

Easier key navigation in result lists

To support easier keyboard navigation in result lists, for example by using the down arrow to move from result to result, Appkit result titles now use HTML heading tags.

Specifically, `<search:field styling="title">` tags now use `<h3>` tags instead of `<p>`.

Bug Fixes

- **Fusion 5 authentication.** Apps created using Appkit 4 and App Studio 4 can now authenticate against Fusion 5.0.
- **Fusion sessions.** Existing Fusion sessions are now correctly utilized when authorizing users against a service account.
- **Term suggestions.** Search term suggestions no longer fail when using a Fusion query profile.
If you previously used `requestHandler: /` as a workaround for this functionality, carry out the upgrade step described in *Additional upgrade steps* above.
- **Click tracking table results.** The `<track:clicks>` tag now works correctly with `<table:result-list>`.
- **Click tracking fields.** In some conditions, additional fields were incorrectly tracked by the `<track:clicks>` tag. They are now tracked correctly.

Appkit SDK 4.5.0 Release Notes

Release date: 19 August 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.5.0*.

Also see the additional required and recommended upgrade steps described in the Release Notes for *Appkit 4.4.0* and *Appkit 4.3.0*.

New features

No new features are included in this release.

Improvements

Role-based authorization now supports multiple roles

Appkit now supports role-based authorization with multiple roles. Previously, it was possible to define only a single, global role-based authorization filter.

Create authorization filters

Create multiple configuration files for role-based access in the `security/access` folder. Each file defines a single authorization filter. When Appkit considers whether a specific user can access a specific resource, Appkit evaluates all of the filters.

A best practice is to create authorization filters *that do not overlap*, that is, that do not refer to the same resources.

You must give users access to resources explicitly by referencing one or more of their roles. A user who has none of the roles mentioned in the authorization-filter files cannot access any resources.

If a user has multiple roles that specify different access to a resource, then the rule that denies access wins.

Properties

Property	Required	Notes
<code>allow</code>	optional	Comma-separated list of roles that allow access to a resource
<code>deny</code>	optional	Comma-separated list of roles that deny access to a resource
<code>pattern</code>	required	Bar-separated list of paths to which this role-based authorization filter applies

You must supply either an `allow` or a `deny` property. You can supply both, though that is not necessary, because roles that are not allowed are denied. If you do supply both properties and a role is mentioned in both `allow` and `deny`, then `deny` wins.

Example

In this example, users with the roles `EMPLOYEES` or `CONTRACTORS` can access the resources specified in the `pattern` property, whereas users with the role `INTERNS` cannot access the resources.

```
allow: EMPLOYEES,CONTRACTORS deny: INTERNS pattern: (/twigkit/api)|(/dashboard/*)
```

If authorization filters overlap

Although multiple authorization filters can overlap (reference the same resources in the pattern), this is not a good practice.

Appkit figures out and applies the most restrictive filter, but results might be counterintuitive. For example, filter 1 might allow users with the role `CONTRACTOR` access to a specific resource, whereas filter 4 denies contractors access to the same resource. Filter 4 wins, but you wouldn't see that if you were looking at filter 1.

Timezone handling

UI components and Appkit server components have been improved to ensure that they handle time zones consistently. Previously, some UI components were aware of local time zones and others were not (instead working with UTC times). The result was that some UI components displayed incorrect times.

Improvements are:

- In the `<search:field>` tag, you can now specify a `timezone` parameter, which tells Appkit the timezone to use when formatting date fields in responses. For example:

```
<search:field name="date" styling="label-left" label="Created Date" date-format="dd/MM/yy hh:mm" timezone="PST"></search:field>
```

Use these time zone abbreviations

(https://en.wikipedia.org/wiki/List_of_time_zone_abbreviations).

- Every HTTP request to Appkit's API services includes an `X-Client-Timezone` header, which specifies the browser client's current timezone as a UTC offset (https://en.wikipedia.org/wiki/List_of_time_zone_abbreviations).

Fusion security provider stores JWT to authorize requests to Fusion

Applications that use the Fusion security provider (`twigkit.security.provider.fusion`) can now check whether an incoming request has a JSON web token (JWT) associated with it (in an `Authorization` header that begins with `Bearer`). Appkit extracts the JWT from the header and stores the JWT so that subsequent requests can use the JWT to authorize the requests to Fusion.

The JWT support added in Release 4.5.0 is in addition to the support that was present in prior Appkit releases for pre-authentication with the Fusion security provider.

Now, instead of just checking for a `fusion-user-name` header in the incoming request, Appkit also checks for a JWT header, if it does not find the `fusion-user-name` header.

If a JWT is present, Appkit uses the token to authenticate the user. This is typically used when an application (e.g. App Insights) is running behind a Fusion proxy.

After Appkit retrieves and stores the JWT, Appkit can retrieve the JWT to authorize requests to Fusion. By default, Appkit, handles all of this for you.

Explicit configuration

The means by which a user can authorize a request to Fusion can be made explicit by adding an `auth: auto | serviceaccount | jwt | native` property to the app's Fusion

services resources under `resources/services/api/fusion.conf`. The default value of this property is `native`, which means that Appkit uses the pre-authentication provided by the Fusion security provider.

If you use the `auto` value of the `auth` property, then Appkit works its way through `serviceaccount`, `jwt`, and finally `native`. With the `serviceaccount` option, Appkit tries to find a `userName` and `password` in `resources/services/api/fusion.conf`. With the `jwt` option, Appkit tries to generate a JWT for the user. (This is not the same as the JWT passthrough.)

Geo-filtering and sorting is supported for Elastic 5

In Appkit 4.4.0, support for geo-filtering and sorting was added to the Elasticsearch search platform. The support in Appkit 4.4.0 was solely for Elastic 6.

In AppKit 4.5.0, geo-filtering and sorting support has been extended to include Elastic 5.

Accessibility is improved for assistive technologies

Appkit's Angular JS now incorporates ngAria (<https://docs.angularjs.org/guide/accessibility>), which improves accessibility by including ARIA (<https://www.w3.org/TR/wai-aria/>) attributes in the HTML of Appkit UI components. ARIA attributes help assistive technologies such as screen readers to convey state and semantic information to users.

Bug Fixes

- **Hierarchical facets.** Hierarchical facet requests now always return hierarchical facets. Previously, a `<search:facet-list>` request with multiple hierarchical facets would sometimes return a flattened structure (parent only) for some of the facets.
- **dateFormat and time zone.** The `dateFormat` filter now applies the `timezone` parameter correctly to both date-time values in a date-time range.
- **Responses from multiple suggesters.** When there are multiple suggesters, the `<query:suggestions>` element now includes responses from all suggesters in their respective sections. Previously, only the responses of one suggester were included.

Appkit SDK 4.4.0 Release Notes

Release date: 22 July 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.4.0*.

If you are upgrading to 4.4.0 from a version prior to 4.3.0, *additional upgrade steps* are required or recommended.

Recreate topics that contain non-URL-safe characters

An issue in the Social Module in prior releases prevented topics that contained non-URL-safe characters from loading correctly in browsers.

The fix in this release allows *future* topics to contain URL-unsafe characters; the topics load correctly. But this release *does not* repair existing topics.

To repair existing topics, you must delete and recreate the topics.

New features

Elasticsearch search platform supports geo-filtering and sorting

The Elasticsearch search platform now supports geo-filtering and sorting based on distance.

Filter syntaxes

These are the filter syntaxes. The `unit` property is optional. If not present, the default unit `kilometers` is used.

```
f=geo-point-field[{"lat":latitude-value,"lon":longitude-value,"distance":distance-value}][display-value] f=geo-point-field[{"lat":latitude-value,"lon":longitude-value,"distance":distance-value,"unit":"unit-value"}][display-value]
```

Filter properties

Property	Required	Notes
lat	required	Latitude value (decimal, for example, 37.773972)
lon	required	Longitude value (decimal, for example, -122.431297)
distance	required	Distance from a latitude/longitude point (a positive real number, for example, 10 or 2.5) Do not specify the distance units as a part of distance (as you would in Elasticsearch). Use unit instead.
unit	optional	Units of distance. Possible values are the long form distance units (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/common-options.html#distance-units), that is, miles, yards, feet, inch, kilometers, meters, centimeters, millimeters, and nauticalmiles. The default is kilometers.

Sorting

The syntax of the URL query parameter for sorting on a geo_point field is the same as for other sorting (+ for ascending, - for descending):

```
s=+geo-point-field s=-geo-point-field
```

Example queries with geo-filters

Explicit query, default units (kilometers):

```
http://localhost:8080/#/search?  
q=*&f=pin.location[{"lat":40,"lon":-70,"distance":200}]["Within 200 km"]
```

Implicit query, non-default units (miles):

```
http://localhost:8080/#/search?  
f=pin.location[{"lat":40,"lon":-70,"distance":200,"unit":"miles"}]["Within 100 miles"]
```


Implicit query with sorting (descending), default units (kilometers):

```
http://localhost:8080/#/search?
f=pin.location[{"lat":40,"lon":-70,"distance":1000}][{"Within 1000
km"]&s=+pin.location
```

End-to-end example

In Elasticsearch

1. Create an index. For example:

```
curl -X PUT "localhost:9200/map_locations" -H 'Content-Type: application/json'
-d' { "settings" : { "number_of_shards" : 3, "number_of_replicas" : 2 } } '
```

2. Apply a geo-point mapping to a field in the index. For example:

```
curl -X PUT "localhost:9200/map_locations" -H 'Content-Type: application/json'
-d' { "mappings": { "_doc": { "properties": { "pin": { "properties": {
"location": { "type": "geo_point" } } } } } } } '
```

3. Index documents that contain one or more fields of type `geo_point` (<https://www.elastic.co/guide/en/elasticsearch/reference/6.8/geo-point.html>) (any of the four formats). For example:

```
curl -X PUT "localhost:9200/my_locations/_doc/1" -H 'Content-Type:
application/json' -d' { "pin" : { "location" : { "lat" : 40.12, "lon" : -71.34
} } } ' curl -X PUT "localhost:9200/my_locations/_doc/2" -H 'Content-Type:
application/json' -d' { "pin" : { "location" : { "lat" : 42.12, "lon" : -74.34
} } } ' curl -X PUT "localhost:9200/my_locations/_doc/3" -H 'Content-Type:
application/json' -d' { "pin" : { "location" : { "lat" : 0.12, "lon" : -1.34 }
} } '
```

4. In the Elasticsearch configuration file (<https://www.elastic.co/guide/en/elasticsearch/reference/6.8/settings.html>) `elasticsearch.yml`, identify the field or fields that are of type `geo_point`. For example:

```
geo-point-fields: pin.location
```

In Appkit

1. Perform a search with a filter that identifies a `geo_point` field and specifies a latitude, a longitude, and a distance (and optionally units). For example:

```
http://localhost:8080/{hash}/search?
q=:*&f=pin.location[{"lat":40,"lon":-70,"distance":200}]["Within 200 km"]
```

2. Optionally, include an `s` (sort) query parameter to sort search results on a `geo_point` field that was used in a filter; for example, `s=+pin.location` (ascending) or `s=-pin.location` (descending). For example:

```
http://localhost:8080/{hash}/search?
q=:*&f=pin.location[{"lat":40,"lon":-70,"distance":200}]["Within 200
km"]&s=+pin.location
```

3. The Elasticsearch platform performs an Elasticsearch `geo_distance` query, which finds documents with at least one geo-point that falls within the specified distance of the specified point (and that satisfy any other query and filter criteria), and returns a response that lists the documents. If the `s` query parameter was included, the documents listed in the response are sorted in order of distance from the specified point.

Improvements

Appkit filter syntax supports complex filter values

Previously, the Appkit filter syntax supported a small number of simple data types (`string`, `boolean`, `integer`, `double`, and `date`) and ranges over simple data types.

With this release, the Appkit filter syntax adds support for an `object` data type that can specify filters of arbitrary complexity. The `object` data type uses JSON syntax (<https://json.org/>) to describe the data. For example, a filter can specify unordered lists of name/value pairs (objects) and ordered collections of values (arrays).

For example:

```
data[{"property1":"some_text","property2":34,"property3":true,"property4":9223372
036854775807,"property5":53.4678,"property6":["a","b"],"property7":
{"key1":"value","key2":56}}]["A filter object"]
```

Security providers return 401 Unauthorized for unauthenticated API requests

All Appkit security providers that use Spring security now return a `401 Unauthorized` status for unauthenticated API requests. Previously, they returned a `403 Forbidden` status.

Additionally, for the SAML security provider, the standard AJAX request handling mechanism has been included, which means that `302 Found` redirects do not occur for unauthenticated API requests.

The security providers affected are:

- Active Directory
- Twigkit Cookie Cracker
- Lucidworks Fusion Sessions API
- Kerberos
- LDAP
- ADFS OAuth
- Facebook OAuth
- Google OAuth
- Office 365 OAuth
- SAML
- Spring Security

Histograms and sliders work with small timeframes

Histograms and sliders now work correctly with date ranges that have small timeframes, for example hours or minutes.

The filter-optional attribute has been added to <search:field>

When using `<search:field>` to make a field clickable in order to refine the current query, you can now define the `filter-optional` tag attribute. This tag attribute tells Appkit how to combine filters using connective logical operators.

For example, when you have a multi-select (checkboxes) facet defined on the same field and displayed on the same page as a clickable field for refining a query, you must specify `filter-optional` to be `value` in the filter for the facet directive.

For more information, see *Optional logical operators*.

Bug Fixes

- Click tracking for instant results (`<instant:result-list>`) now returns the correct `query` object in the tracked-click object. These are examples of markup and a response object:

```
<search:box placeholder="Find documents..." action="search" social="true"
query="query"> <track:clicks query="query"> <!-- Instant Results --> <div>
<instant:result-list platform="platform" query="query"> <search:result>
<search:field name="title" styling="title" urlfield="url"></search:field>
</search:result> </instant:result-list> </div> </track:clicks> </search:box>
```

```
{ id: "quickstart/Revolution-Session-Data.csv#36", label: "Passage Search: the
Answer to Your User Questions!", offset: 0, pos: 3, query: {rpp: "14", fa:
"keywords_s", p: 1, q: "sa*"}, type: "result", url:
"http://localhost:8090/quickstart/Revolution-Session-Data.csv#36" }
```

- When using the Social module, topics that contain non-URL-safe characters (for example, #) now load in browsers correctly, *if the comments were created using this software version, 4.4.0.*

This release does not repair existing topics that contain non-URL-safe characters. You must *recreate those comments*.

- When using the `<search:field>` tag, fields with no value are no longer clickable.
- The localization key `collaborate.saved.placeholder` was added for the hint text in the search box for the `collaborate:saved-query-list` tag. The search box lets the user

search through a list of saved queries. The default English hint text is `Search name`.

- The root context page is now an allowed login location.
- When grouping results, the first related result no longer shows values for all related results.
- When grouping results using the platform `groupField` parameter, the first result in a group of results now contains only its own metadata. Previously, the first result contained the metadata of all of the results in the group.
- The `<widget:date-range>` tag now completely clears dates when the date filter is cleared.

Appkit SDK 4.3.0 Release Notes

Release date: 29 May 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.3.0*.

This section explains *additional steps* that are required or recommended (as indicated) for upgrading to this version of Appkit.

Spring Security XML schema

Update your apps that use Spring security to use version 4.2 of the Spring Security XML schema. In any application that has its own `spring-security.xml` file:

1. Replace this Spring Security XML schema declaration:

```
http://www.springframework.org/schema/security/spring-security-4.1.xsd
```

With this one:

```
http://www.springframework.org/schema/security/spring-security-4.2.xsd
```

Using a Fusion service account for authentication and authorization

If you used a service account to authenticate against Fusion in a prior Appkit version, you must move the `userName` and `password` configuration values from your Fusion platform configuration to `services/api/fusion.conf` in this version.

Service accounts no longer require the `impersonate` configuration option to be set to `true`. If the option is set to `true` in `services/api/fusion.conf`, it is ignored.

For more information, see *Fusion authorization*.

Authorization and pre-authorization filters

For existing applications that implement custom `AuthorisationFilter` logic, you can and should re-implement those filters as instances of `PreAuthorisationFilter`, if the intention of those filters is not to authorize a request, but instead to augment the user object with pre-authorization information. However, this is not a necessary requirement for upgrading, because existing authorization filters work exactly as before. (They run after the pre-authorization filters.)

Changing how to send signals to Fusion

This change was introduced in Appkit 4.0.0. If you have not already made this change, it is necessary now because the old approach to generating signals is *deprecated and has been removed*.

How to change how signals are sent to Fusion

1. Remove the following from your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId> <artifactId>twigkit.activity.fusion-  
signals</artifactId> <version>${project.parent.version}</version>  
</dependency>
```

2. Add the following to your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>  
<artifactId>twigkit.activity.tracker</artifactId>  
<version>${project.parent.version}</version> </dependency> <dependency>  
<groupId>twigkit</groupId>  
<artifactId>twigkit.message.service.fusion.producer</artifactId>  
<version>${project.parent.version}</version> </dependency>
```

3. Add configuration parameters in `conf/message/service/fusion.conf`. You can send signal events to Fusion using either the Signals API or the Query API. This is an example in which the `query-profile` parameter is used to send events with the Fusion Query API:

```
query-profile: my-profile commit: true async: false signals-index-pipeline: my-  
index-pipeline
```

The parameter `signals-index-pipeline` is an optional parameter that defines a specific index pipeline to use when indexing signals. If no index pipeline is specified, the pre-configured `_signals_ingest` pipeline is used.

The `signals-endpoint` configuration parameter is no longer supported.

4. By default, Appkit sends all events to the Signals endpoint. If you want to customize which events are sent, add a configuration file at `activity/tracking/tracker.conf` with the property `events`:

```
events: click, request, response, login, annotation # or any subset thereof
```

A query pipeline automatically generates response signals when the signals feature is enabled for a collection. The search application should not send response signals to Fusion directly, because those would conflict with the automatically generated signals.

New features

- **JWT authorization against Fusion is supported.** Appkit now supports JSON Web Token (JWT) authorization against a Fusion JWT realm. For information about options for authorization in Fusion, see *Fusion authorization*.
- **Discovery of Fusion proxy instances.** By default, Appkit now attempts to discover a Fusion Proxy service instance for querying. This removes the need to colocate a Fusion Proxy Service instance on every node running the Fusion Web Apps service.

- If Appkit discovers a proxy, Appkit uses the discovered proxy instead of the configured `host` and `port` in `services/api/fusion.conf`.
- If Appkit does not discover a proxy, Appkit uses the configured `host` and `port` in `services/api/fusion.conf`, or `localhost:8764` if the configuration values are not set.

Proxy discovery uses the `fusion-proxy` Java system property, which is usually only available when an Appkit app runs on the Fusion Web Apps service.

Configuration:

To specify how Appkit finds a Fusion Proxy Service instance, define the property `fusion-discovery-enabled` in `services/api/fusion.conf`:

- `fusion-discovery-enabled: true` (default if omitted). Appkit attempts to discover a Fusion proxy instance by reading the `fusion-proxy` Java system property. If it does not discover a proxy, Appkit uses the configured `host` and `port` in `services/api/fusion.conf`, or `localhost:8764` if the configuration values are not set.
- `fusion-discovery-enabled: false`. Appkit does not attempt to discover a proxy, and uses the configured `host` and `port` in `services/api/fusion.conf`, or `localhost:8764` if the configuration values are not set.

In an existing application that is deployed to Fusion, but that is not designed to query the Fusion server on which it is deployed for the location of the Fusion Proxy service, you must add the `fusion-discovery-enabled` parameter to `fusion.conf` with a value of `false` to retain the current behavior.

- **Role-based authorization.** Appkit adds an authorization filter that can authorize requests for protected resources based on the roles associated with the user who made the request. Security providers (such as SAML) or LDAP lookups can provide the roles. For information, see *Role-based authorization*.
- **Solr JSON range facets.** Appkit now supports Solr JSON range facets as follows:
 - Appkit can now parse facets that were generated using JSON syntax and leverage them for display purposes in Appkit UI components.
 - Appkit cannot generate JSON facet query syntax (which needs to be manually input in the query pipeline or as query custom parameters).
 - Appkit cannot use the JSON facet's filters to further refine query.
- **Localization.** Appkit replaces hard-coded labels (in English) with calls to an Appkit translation service. The service provides a label in the chosen language, with a fallback to English if no translation exists for the label in the chosen language. For more information, see *Localization service*.

This changes the attributes for a number of tags.

- **Stack traces in browser.** Appkit can send stack traces to the browser in responses (instead of HTML error pages) when run in development mode.

To enable full stack traces:

1. Add a new filter, `twigkit.http.filter.DevelopmentErrorFilter`:

```
<filter> <filter-name>ErrorFilter</filter-name> <filter-
class>twigkit.http.filter.DevelopmentErrorFilter</filter-class> <!-- <init-
param>--> <!-- <param-name>responseCode</param-name>--> <!-- <param-
value>500</param-value>--> <!-- </init-param>--> </filter> <filter-mapping>
<filter-name>ErrorFilter</filter-name> <url-pattern>*/url-pattern>
<dispatcher>ERROR</dispatcher> </filter-mapping>
```

1. To only send stack traces for a subset of error response codes, e.g. 500 and 404, uncomment the `<init-param>` block and specify the response codes as a list of comma-separated values in `<param-value>`. For example:

```
<init-param> <param-name>responseCode</param-name> <param-value>403,404,500</param-value> </init-param>
```

Improvements

- **Retrieval of properties by workflow platforms.** If a platform is configured to be a workflow platform, platform property retrieval proceeds as follows:
 - a. Appkit checks to see whether the workflow platform has a property.
 - b. If the workflow platform does not have the property, Appkit checks to see whether the underlying platform has the property.
 - c. If both the workflow platform and the underlying platform have the same property, the one defined for the workflow takes precedence.
- **Solr result clustering.** Appkit adds support for Solr result clustering (https://lucene.apache.org/solr/guide/7_1/result-clustering.html).
- **No longer necessary to configure wt=xml in Fusion.** Appkit's Solr Suggestions service that is used when backward compatibility is enabled (`SolrSuggestionsParserService`) now appends `wt=xml` to the Solr suggestions URL automatically. You no longer need to configure this query parameter in the Fusion query profile for suggestions.
- **Improved performance in Social module.** Performance of the Social module has been improved for annotation deletions.
- **Overwrite context path.** It is now possible to overwrite the context path. This is necessary when search app HTML files are hosted on one server and the Appkit APIs are on a different server.
- **No separate license restriction for Workflow module.** The separate license restriction for the Workflow module has been removed.
- **Lucene version.** The `twigkit.lucene` module now uses Lucene v7.2.1. You can use the `twigkit.lucene` module with other modules, for example, Elasticsearch 6, that use more recent versions of Lucene.
- **Default requestMethod for Solr search platform.** The default `requestMethod` for the Solr search platform has been changed from GET to POST.

Role-based authorization

In many cases, you might want to restrict access to your application, or parts of your application, to specific groups of users. To enable this option, you must add the file `conf/security/access.conf` to your application's configuration tree with these configuration parameters:

```
allow: role-with-access-1,role-with-access-2,... deny: role-without-access-1,role-without-access-2,... pattern: regex-of-uris-to-intercept
```

These configuration parameters are:

- `allow`: A comma-separated list of roles that should be granted access. When not specified or set to a wildcard ('*'), Appkit defaults to allowing access to all roles.
- `deny`: A comma-separated list of roles that should be denied access.
- `pattern`: A regular expression that defines the pattern of URIs that should be intercepted for checking authorization. This defaults to `.*` (all paths are checked).

Example

Let us assume that you have a user directory containing two users, `user` and `admin`, where only the latter has an `ADMIN` role. For example, you can define this using a simple `spring-security.xml` configuration with a static list of users, like this:

```
<authentication-manager> <authentication-provider> <user-service> <user
name="user" password="user" authorities="USER"/> <user name="admin"
password="admin" authorities="USER,ADMIN"/> </user-service> </authentication-
provider> </authentication-manager>
```

To configure role-based access, add `conf/security/access.conf` to the application with this configuration:

```
allow: ADMIN pattern: (/)|(\/twigkit/api/.*)
```

This says that, for all requests to `/` (the root page) and the API service, Appkit applies role-based authorization rules. Including the root page is not strictly necessary, but it leads to a better experience for users who are denied access.

To validate this setup, first log in as `user`. Appkit denies you access, as shown in the application logs:

```
TRACE t.s.SecurityFilterExecutionModule - Filtering with
[twigkit.security.filter.RoleBasedAuthorizationFilter] TRACE
t.s.f.InterceptAuthorizationFilter - Filtering request to / - comparing against
access pattern (/)|(\/twigkit/api/.*) TRACE t.s.f.RoleBasedAuthorizationFilter -
User 'user' is DENIED access to protected resource ERROR
t.s.SecurityFilterExecutionModule - Authorization chain failed - returning 403
```

Next, log in as `admin`. Appkit allows you to access the app, as shown in the application logs:

```
TRACE t.s.SecurityFilterExecutionModule - Filtering with
[twigkit.security.filter.RoleBasedAuthorisationFilter] TRACE
t.s.f.InterceptAuthorisationFilter - Filtering request to / - comparing against
access pattern (/)|(\/twigkit/api/.*) TRACE t.s.f.RoleBasedAuthorisationFilter -
User 'admin' is GRANTED access to protected resource
```

Bug Fixes

- The default `requestMethod` for the Solr search platform has been changed to POST to avoid URL length limitations. Previously, the default was GET.
- An Apache Xerces dependency conflict between the SAML and Fusion security providers has been resolved.

- An error was corrected in the Spring security file `spring-security.xml` for Active Directory authentication. The `<security:loginForm>` tag lacked this line, which has been added:

```
login-processing-url="/j_spring_security_check"
```

- When using the Social module with the Fusion platform, the index pipeline and query pipeline now default to `_system` if `social.conf` specifies the `collection`. Instead of specifying the `collection` in `social.conf`, you can specify the `index-profile` and `query-profile`.
- The `slider` widget now submits the correct query.
- The bookmark icon on a result document now correctly indicates whether the document is bookmarked. Previously, deleting a bookmark from the bookmark list or search-box dropdown did not update the icon on the document.
- Click signals are now posted correctly to Fusion when Fusion is configured for single sign-on (SSO).
- The CSRF profile filter occasionally returns the wrong request tokens for a user.
- Previously, some query strings were incorrectly URL-encoded when being persisted in the Social module. The query strings are now persisted correctly.
- The `<search:facet>` tag now displays breadcrumb names such as `date` that were erroneously omitted.
- In the Social module, likes are now deleted when the user deselects the Like (thumb) icon.
- The task framework now allows completed tasks to be deleted.
- When using Appkit Guice security filters (not Spring security) failed authentication requests now return 401 rather than 200.
- When `label-field` is not set, Appkit no longer displays an empty tooltip when hovering over a marker on a map.

New features

The following are new features in this release:

Deprecations

The following are deprecations in this release:

- **Old method of sending signals to Fusion.** How to send signals to Fusion has changed in this release. The `twigkit.activity.fusion-signals` module is deprecated and has been removed. Use the `twigkit.message.service.fusion.producer` module instead. For more information, see *Changing how to send signals to Fusion*.
- **Service account configuration option 'impersonate'.** Service accounts no longer require the `impersonate` configuration option to be set to `true`. If the option is set to `true` in `services/api/fusion.conf`, it is ignored.

An alternative means of authorizing with a service account is provided.

If you used a service account to authenticate against Fusion in a prior Appkit version, you must move the <code>userName</code> and <code>password</code> configuration values from your Fusion platform configuration to <code>services/api/fusion.conf</code> in this version.

Improvements

The following are improvements in this release:

Fusion authorization

Appkit now has three Fusion authorization modes:

Mode	Description
Fusion native realm	Authorization is against the <i>Fusion native security realm</i> directly, using Fusion session passthrough. This is most useful for prototyping apps with Fusion's Web Apps service.
Service account	Authorization uses a service account in the <i>Fusion native security realm</i> . This is useful with external security providers, but entails having a username and password in the Appkit configuration.
Fusion JWT realm	Authorization is of individual users against the <i>Fusion JWT security realm</i> . This is also useful with external security providers, and there is no username and password in the Appkit configuration that might allow an attacker to gain access to the Fusion UI.

Specify one of these modes or `auto` in the *Fusion authorization configuration*.

Service accounts no longer require the `impersonate` configuration option to be set to `true`. If the option is set to `true` in `services/api/fusion.conf`, it is ignored. An *alternative means of authorizing with a service account* is provided.

If a configured authorization mode is not available (that is, configuration keys are missing), then Appkit uses the `native` mode as a fallback. An application is never left in a state in which it cannot at least try to authorize.

Configuration

By default, Appkit reads `services/api/fusion.conf` for configuration.

Fusion configuration

Provide the `auth` configuration parameter in `services/api/fusion.conf` to configure Fusion authorization. The `auth` parameter can have these values.

Value	Description
<code>native</code>	Authorization is against the <i>Fusion native realm</i> directly, using Fusion session passthrough. This is most useful for prototyping apps with Fusion's Web Apps service.
<code>serviceaccount</code>	Authorization uses a service account in the <i>Fusion native security realm</i> . This is useful with external security providers, but entails having a username and password in the Appkit configuration.
<code>jwt</code>	Authorization is of individual users against the <i>Fusion JWT security realm</i> . This is also useful with external security providers, and there is no username and password in the Appkit configuration that might allow an attacker to gain access to the Fusion UI.
<code>auto</code> (default)	Appkit searches for an available authentication provider in the order (1) <code>jwt</code> , (2) <code>serviceaccount</code> , and (3) <code>native</code> . Appkit uses the first authorization provider that confirms it has the necessary configuration to attempt authorization.

Service account configuration

To configure Appkit for service account authorization against Fusion, include the following configuration options in `services/api/fusion.conf`:

Option	Description
<code>userName</code>	User name of the user for authorization (String, encryptable)
<code>password</code>	User's password (String, encryptable)

Service accounts no longer require the `impersonate` configuration option to be set to `true`. If the option is set to `true` in `services/api/fusion.conf`, it is ignored.

If you used a service account to authenticate against Fusion in a prior Appkit version, you must move the `userName` and `password` configuration values from your Fusion platform configuration to `services/api/fusion.conf` in this version.

JWT configuration

To configure Appkit for JWT authorization against Fusion, include the following configuration options in `services/api/fusion.conf`:

Option	Description
<code>issuer</code>	Issuer that is configured in the Fusion JWT security realm (String, encryptable)
<code>key</code>	Key that is configured in the Fusion JWT security realm (String, encryptable)
<code>groups</code>	Group or list of groups. Groups must match those configured in the Fusion JWT security realm (Comma-separated values, encryptable)
<code>subject</code>	Override the current user's ID with this value. Use this when all JWT requests should be sent as a single user (for impersonation or service account emulation). (String; optional, encryptable)

Native configuration

No configuration is required for `native` authorization.

App localization

You can localize the UI labels in an app in multiple languages of your choice.

Localization service

Appkit obtains the correct labels for a language by making calls to an Appkit translation service. The service provides a label in the chosen language, with a fallback to English if no translation exists for the label in the chosen language.

Loading translations

You can use the new `<translations:localize>` JSP tag to load a set of UI label translations into an app. The tag syntax is:

```
<translations:localize dictionary="*dictionary*" locale="*locale*" />
```

- **dictionary**. Name of the resource bundle from which to load translations. This is also the first part of the name of each property file that contains translations.
Default: `translations`
- **locale**. Locale
Default: `en` (English)

The **dictionary** and **locale** are used to find the file as follows:

1. In `src/main/resources`, the Appkit translation service looks for a resource bundle named **dictionary**, for example, `translations`.
2. Within the resource bundle, the translation service looks for property files named **dictionary_locale.properties**, for example, for the files `translations_en.properties` and `translations_fr.properties`.

Enabling translations

1. Find label strings that need to be translated in the UI and identify the *key names for the labels*.
2. Create translation property files and add them to a resource bundle in `src/main/resources`. For example, add `translations_en.properties` and `translations_fr.properties` to the resource bundle `src/main/resources/translations`. Edit the property files and include the label keys and translations. For example:
In `translations_en.properties`:

```
# translations_en.properties ... components.pagination.next = Next  
components.pagination.previous = Previous ...
```

In `translations_fr.properties`:

```
# translations_fr.properties ... components.pagination.next = Suivant  
components.pagination.previous = Précédent ...
```

Some translations include variables. For example, `components.response-statistics.showing` is `Showing {first} – {last} of {total}` in English. Translations should retain the variables, though their order might differ. Labels can use `<em>` tags for emphasis, for example, `More results like <em>{result}</em>`.

3. Add this `taglib` directive at the top of your app's `index.jsp`:

```
<%@ taglib prefix="translations" uri="/twigkit/translations" %>
```

4. Add the Appkit `<translations:localize>` JSP tag inside the `<head>` element of your app's `index.jsp`:

```
<translations:localize locale="${param.locale}" />
```

5. Add this rule to the `url-rules.xml` file:

```
<!-- Language specific pages --> <rule> <from>^/lang/(.*)/</from> <to  
last="true">/index.jsp?locale=$1</to> </rule>
```

You can modify other URL rules to suit your application logic. The `<translations:localize>` JSP tag works the same.

6. Restart your app.

7. Verify that the translations are present by visiting pages in the app that should be in different languages.

Tag attribute changes

There are localization-related attribute changes for these tags:

- `<collaborate:bookmark>`: Attributes `tooltip`, `active-tooltip`, `singular-tooltip`, and `plural-tooltip` have been removed.
- `<collaborate:bookmark-list>`: Attribute `empty-text` has been removed.
- `<collaborate:comment-list>`: Attributes `delete-label`, `show-more-label`, `show-less-label`, and `default-delete-message` have been removed.
- `<collaborate:like>`: Attributes `tooltip`, `active-tooltip`, `singular-tooltip`, and `plural-tooltip` have been removed.
- `<collaborate:topic-list>`: Attribute `empty-text` has been removed.
- `<search:breadcrumbs>`: Attribute `exclude-label` has been removed.
- `<search:facet>`: Attributes `show-more-label` and `show-less-label` have been removed.
- `<search:pagination>`: Attributes `next-label` and `previous-label` have been removed.
- `<search:response-statistics>`: Attributes `showing-all-label`, `showing-label`, `of-label`, and `to-label` have been removed.
- `<widget:spelling-suggestions>`: Attributes `label`, `original-label`, and `corrected-label` have been removed.

Appkit SDK 4.2.5 Release Notes

Release date: 5 April 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.5*.

New features and improvements

There are no new features or improvements in this release.

Fixed issue

The following issue is fixed in this release:

- Click tracking result position does not work with nested results.

Appkit SDK 4.2.4 Release Notes

Release date: 5 April 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.4*.

New features and improvements

There are no new features or improvements in this release.

Fixed issues

The following issues are fixed in this release:

- Spring security configuration for Active Directory is missing `login-processing-url`.
- Breadcrumbs do not display the field name "date".
- Likes are not deleted when a user unselects the thumb icon.
- Highlighting in table cells displays emphasis as plain-text `<em></em>` tags.

Appkit SDK 4.2.3 Release Notes

Release date: 25 March 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.3*.

New features and improvements

You can add the optional `search-wildcard` attribute to the `search-facet` directive to specify whether Appkit performs a wildcard search within facet names and, if it does, how it attaches the wildcard character(s) to the terms that users enter.

A wildcard character matches zero or more characters where it is placed. For example, `computer*` would match the facets `computers` and `computer accessories`. This wildcard is often represented as an asterisk (*).

The `search-wildcard` attribute can have these values:

Value	Description
<code>left</code>	Adds a wildcard prefix to facet search terms, for example, <code>term</code> (entered) becomes <code>*term</code> (used to search).
<code>right</code>	Adds a wildcard suffix to facet search terms, for example, <code>term</code> (entered) becomes <code>term*</code> (used to search).
<code>both</code> (default)	Adds both a wildcard prefix and suffix to facet search terms, for example, <code>term</code> (entered) becomes <code>*term*</code> (used to search).
<code>none</code>	Adds no wildcard to facet search terms (exact match).

Fixed issues

This release contains these fixes:

- Previously, some click signals caused HTTP 401 errors. These are now sent correctly.
- Saved queries with page parameters now work correctly.

Appkit SDK 4.2.2 Release Notes

Release date: 14 March 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.2*.

New features and improvements

There are no new features or improvements in this release.

Fixed issues

The following issues are fixed in this release:

- The Cross-Site Request Forgery (CSRF) profile filter occasionally returned the wrong request tokens to the user.
- After deleting a bookmark from a bookmark list, the bookmark icon on the result document remained in the "bookmarked" state.

Appkit SDK 4.2.1 Release Notes

Release date: 8 March 2019

Upgrading to this version

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.1*.

New features and improvements

There are no new features or improvements in this release.

Fixed issues

The following issue is fixed in this release:

- Using a two-date time picker widget, a user could select a From time that was after the To time. The `widget:date-range` element now ensures that the user can only select a From time that precedes the To time.

Appkit SDK 4.2.0 Release Notes

Release date: 28 February 2019

Appkit SDK 4.2.0 is the version of Appkit that is included in Fusion App Studio 4.2.0.

Also see the *App Studio 4.2.0 release notes* and the *Fusion Server 4.2.0 release notes*.

Upgrading to this version

Appkit 4.2.0 is the version of Appkit that is included in Fusion App Studio 4.2.0.

In apps created with prior versions of Fusion App Studio, you can *upgrade the Appkit version to 4.2.0*.

New features

The following are new features in this release.

Social module

- **Single response processor for social module.** You can now configure a single response processor for the Social module, instead of having to configure separate ones for likes, comments, bookmarks, and so forth.
- **Persist social data in Fusion.** You can now *persist Social module data in a Fusion collection*.

Fusion

- **Query against the Fusion Analytics Catalog.** A Fusion Analytics Catalog platform has been added to query against the Fusion Analytics Catalog collection. This platform translates Appkit queries into SQL statements which are executed against a Fusion query pipeline that includes an *Analytics Catalog Stage*. The Fusion Analytics Catalog platform uses the *Fusion Catalog API*.
For more information, see the *Fusion Analytics Catalog platform*.
- **Fusion platform supports index profiles.** The Fusion platform adds the ability to use an index profile to index data into Fusion. When a read/write Fusion platform is present and an index profile is specified, then Appkit can store, update, and delete data in the Fusion index.
To use an index profile, put `index-profile: my-index-profile-name` in your Fusion platform configuration, replacing `my-index-profile-name` with the index profile Appkit will use. Index profiles take precedence over index pipelines.
- **Use Fusion preauthentication.** Fusion preauthentication is now available in the Fusion Security Provider module. By default, if an app is detected as having been uploaded to Fusion and the user has already logged into Fusion, then the user will not need to log into the app.

Miscellaneous

- **Render facets in tabular format.** A new web service lets you render facets in a tabular format.

A new API endpoint, `/core/services/platform/table`, lets you retrieve the output of a Fusion/Solr pivot query.

The endpoint renders the data as a set of headers and a set of rows. The output looks like this:

```
{headers: [h1,h2,h3,h4]; rows: [c11,c12,c13,c14], [c21,c22,c23,c24] }
```

You can leverage the endpoint in `pivot:table` elements.

- **Satellite platform can delegate search requests.** A Satellite Platform can delegate search requests to any underlying platform.
To use a satellite platform, place a new platform configuration file under the app's resources; for example, a `satellite.conf` configuration file in `conf/platforms/satellite/`. Set these parameters in your platform configuration file:
 - `name`: `twigkit.search.satellite.Satellite` (required)
 - `host`: URL of the host for the underlying platform. It must begin with `http` or `https`.
 - `platform-config`: Platform configuration being delegated to
 - `username`: Username for authentication on the underlying platform
 - `password`: Password for authentication on the underlying platform. It is the encoded output of the Appkit `twigcrypt` utility.

The `username` and `password` are required if Basic authentication is required to request search results from the underlying platform, for example, as is the case with Solr.
For example:

```
name: twigkit.search.satellite.Satellite platform-config: platforms.fusion.data
username: search-user password: Enc(hfuqeif894732)
```

- **Query parameters are included in tab links.** Query parameters are now included automatically in the links for tab names.
- **Query by drawing rectangles on ESRI maps.** Users of Appkit web apps that display ESRI maps can now make geospatial queries by drawing rectangles on maps. The control for drawing rectangles is between the plus and minus signs.
Google Maps has had this feature previously.
- **Specify date formats.** The Date Format processor `DateFormatProcessor` lets you specify the format for the display value of a date field.
For more information, see *DateFormatProcessor*.
- **Use JSP tag library utility functions.** A JSP tag library and utility functions within the library are available to JSP pages.
A dependency on `twigkit.fusion.ui-jsp` provides a JSP tag library. Adding `<%@ taglib prefix="app" uri="/lucidworks/app" %>` to the top of a JSP page exposes utility functions within that library. Adding `${app:contextPath(pageContext.request)}` sets the context path of the app on that page, taking into account the Fusion webapps proxy.
- **Use Elasticsearch 6 spell correction/did-you-mean.** Support has been added for Elasticsearch 6 spell correction/did-you-mean functionality via term and phrase suggesters:
<https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-suggesters-term.html> (<https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-suggesters-term.html>)
<https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-suggesters-phrase.html> (<https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-suggesters-phrase.html>)

Improvements

The following are improvements in existing features in this release.

- **Date formats.** Date formats are applied to date ranges in facet series in charts.
- **Range values.** The `numberFormat` and `dateFormat` value filters now handle range values automatically by formatting the endpoints of a range individually. With this change, you must remove any special handling of range values in individual widgets.
- **Multiselection of highmaps regions.** A new `select` attribute has been added to the `geo:highmaps` tag, which takes these values:
 - `click` (default) – User selects a filter by clicking a region.

- `multiselect.*` Select multiple filters that get combined with a logical disjunction (OR). Shows an Apply button that is enabled if and only if one or more checkboxes have been ticked.
- `multiselect-instant`. Select multiple filters that get combined with a logical disjunction (OR). Filters are applied immediately when a checkbox is ticked.
- **Tag `search:dropdown` renamed.** The `search:dropdown` tag has been replaced with the `facet:dropdown` tag. All functionality from the prior tag exists in the new one.
- **Control filter selection for `facet:dropdown`.** A new `select` parameter has been added to the `facet:dropdown` tag, which takes these values:
 - `toggle` (default) – Only allows one filter to be selected at a time, but shows unselected filters and counts.
 - `multiselect.*` Select multiple filters that get combined with a logical disjunction (OR). Shows an Apply button that is enabled if and only if one or more checkboxes have been ticked.
 - `multiselect-instant`. Select multiple filters that get combined with a logical disjunction (OR). Filters are applied immediately when a checkbox is ticked.
- **Pass options to `geo:highmaps`.** The new `options` parameter lets you pass other options to `geo:highmaps`.
- **Show more values for `search:filter`.** A new `show-more-values` parameter has been added to the `search:filter` tag.
 - When `show-more-values` is specified without an assignment (e.g. `<search-field show-more-values ...>`) or with the value `-1`, then Appkit shows all additional values for the field when the user clicks the "Show more" link.
 - When `show-more-values` is assigned a positive integer value (e.g. `<search-field show="5" show-more-values="10" ...>`), then Appkit shows that number of additional values of the field when the user clicks the "Show more" link (or all remaining values if there are fewer values than the number requested).
 - In both cases, Appkit does not show a "Show more" link when there are no additional values to be shown.

Example usage

```
<search:result-list results="results"> <search:result> <search:field
name="multivalued-1" show="2" show-more-values></search:field> <search:field
name="multivalued-2" show="2" show-more-values="10"></search:field>
</search:result> </search:result-list>
```

- **New parameters for `facet:dropdown`.** A number of additional parameters are now available in the `facet:dropdown` tag:

```
facet-name title show show-more max-characters select show-empty show-more-
label show-less-label show-count action order count-number date-format measure
```

These parameters were previously available in the `search:filter-list` tag. They have the same uses.

- **Track clicks for any clickable elements.** You can add click tracking to any clickable element on a page, including clicks on elements that are not anchor elements (and not even hyperlinks). To configure click tracking on custom elements, include your block of trackable elements in a `<track-clicks>` element, as before, and add one or more of the following attributes to the DOM elements for which you want to intercept click events:

- `trackable`.* Clicks are tracked for all DOM elements with a `trackable` attribute, in the scope of a `<track:clicks>` element.
- `data-click-type` (optional) – Specify a custom click type for a trackable DOM element. If `data-click-type` is not specified, the click type defaults to the `type` argument specified in the `<track:clicks>` tag.
- `data-click-url` (optional) – Specify a custom URL to track for a trackable DOM element.
- **Deleting annotations is tracked.** Annotation deletions are now tracked (they generate signals).
- **Restrict `tk.session` cookie to HTTPS channels.** You can now configure whether or not the `tk.session` cookie should be restricted to HTTPS channels over "SSL" (actually TLS). By default, the cookie is sent over all channels (the prior behavior). To restrict the cookie to HTTPS channels, include this entry in `conf/activity/activity.conf`:

```
secure-cookie: true
```

- **Set the hover timeout of a widget popover.** You can set the hover timeout of the widget popover. This is the amount of time in milliseconds that the popover is displayed *after the user leaves the popover trigger*. The optional parameter is `hover-timeout`. The default is 1000 (1 second). Here is an example that sets the hover timeout to 1/10th of a second:

```
<widget:popover mode="hover" hover-timeout="100"> <popover-trigger><button
styling="button">Hover with a short timeout</popover-trigger> <popover-
content> <h1>Hover</h1> <p>The hover timeout of the button you are hovering
over is set to 1/10th of a second.</p> </popover-content> </widget:popover>
```

- **Customize icons for facet lists.** You can customize the icons used to expand and collapse a facet list.
 - a. In the `project-name/assets/styles.css` file, define CSS class selectors for the collapsed (the first of these) and expanded (the second) icons, for example:

```
.custom-collapse-facet .collapsed a.facet-toggle:after { content: '\e817'; *
font-family: "twigkit-basic"; } .custom-collapse-facet a.facet-toggle:after
{ content: '\e815'; font-family: "twigkit-basic"; }
```

- b. Reference the CSS classes in the `search:facet` tag, for example:

```
<search:facet-list response="response" styling="facet-list facet-list-step
facet-list-wrappedheader facet-list-wrappedheader-noborder"> <search:facet
class="custom-collapse-facet" facet-name="test1" title="First Pass Files"
collapsible="true" show="1" show-more="10"> </search:facet> </search:facet-
list>
```

- **Press Enter to save a search query.** After entering a name for a saved search, a user can now press Enter to save the search query.
- **Number formatting for widget:slider.** The `widget:slider` element now supports number formatting. Use a `number-format` parameter to specify a format using the Java `DecimalFormat` (<https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html>). For example, for a Cost facet:

```
<widget:slider response="response" facet-name="Cost_i" number-
format="$#,###.00"></widget:slider>
```

- **Performance improvement for search-enabled facets.** Performance has been improved for search-enabled facets, that is, `search:facet` elements with `search-enabled` set to `true`.
- **Choose fonts to include in web apps.** You can choose among three Less stylesheet language files for including fonts in web apps.
 - `phoenix.all.less` (available prior to 4.2.0) – This stylesheet uses custom fonts fetched from a content delivery network by the browser. If a web app is hosted on a secure site with no Internet access, this stylesheet can have font loading errors that will manifest as "using different fonts" and as an eternally loading spinner in the browser somewhere.
 - `phoenix.all.local-fonts.less` (new in 4.2.0) – This stylesheet includes custom fonts that are hosted in the Appkit web app. However, you cannot use this stylesheet when a web app is accessed via the Fusion Proxy and Web Apps services.
 - `phoenix.all.no-fonts.less` (new in 4.2.0) – This stylesheet excludes all custom fonts.

Suggested usage:

- Normal client, not using the Fusion Web Apps and Proxy services – `phoenix.all.local-fonts.less`
- Normal client, using the Fusion Web Apps and Proxy services – `phoenix.all.less`
- Secure client.* `phoenix.all.no-fonts.less`

To change the Appkit Less file in an app:

Find the reference to the stylesheet in `wro.xml` or `twigkit.less`. After the `appkit-ui` Node module is installed, the Less stylesheets are found in `node_modules/appkit-ui/dist/less/`.

- **jQuery 3.3.1 support.** Appkit upgraded its use of jQuery to version 3.3.1
- **Performance improvement for loading comments.** Loading times for comments in app UIs have been reduced significantly.
- **Configuration property passthrough for Elastic X-Pack.** For the Elastic 6 platform, any configuration properties that start with `xpack` are passed through to the Elastic X-Pack client.
- **Modify path of app URL.** To make your app available at a different URL than `http://hostname:*port*/` (for example, `http://localhost:8080/`), use the `-Dtwigkit.http.context` command-line parameter when starting the app to specify a path component to append to the URL. This can make using a reverse proxy easier.

For example, `-Dtwigkit.http.context=search` would start the app at `http://hostname:*port*/search` (for example, `http://localhost:8080/search`)
Each app must use a different HTTP port.

- **License restriction in Fusion.** The Appkit Fusion platform is now license-restricted. Using the Fusion platform now requires this license property in the Fusion `license.properties` file:

```
property_social: *
```

- **Sort Solr facets by count.** Previously, in the `SolrHierarchicalFacetService` and `GreedySolrHierarchicalFacetService` services, you could only sort by `name`. Now, you can also sort by `count`.
- **Where to enable tracking.** The determination of whether tracking should be enabled is now made in `ActivityModule` instead of in individual trackers.
`Tracking is either enabled (the default) or disabled for all trackers.`
Upgrade information: To upgrade a project in which tracking is disabled to the 4.2.0 version of Appkit, migrate the `enable: false` configuration parameter from `activity/tracking/tracker.conf` to `activity/tracking.conf`.
- **Redirection to logout page when in HTML5 mode.** The `security:user-actions` tag has been improved to redirect to the logout page when the application is in HTML5 mode.
- **Specify hidden fields in search:fields.** In addition to specifying user selectable fields, the `search:fields` element now lets you use the new attribute `hidden-fields` to specify a set of fields that will always be requested, whether or not the user selects them.

Fixed issues

The following issues are fixed in this release:

- When CSRF security is enabled, the backend for Appkit Lightning social/collaboration widgets doesn't pass the request token to the frontend.
- Search query saved previously in the `search:box` component persists in the saved-search pane and must be cleared manually.
- The `widget:date-range` component lets a user select a To date that precedes a From date.
- Custom colors defined in `@highchart-colors` does not work.
- Clearing all breadcrumbs does not clear the breadcrumb for the search query term.
- Lucene index delete queue is not flushed after document deletion.
- The `chart:histogram` element does not display a histogram if there is only a single facet value.
- The indicator of the sorting order for search results is not set when a new sort is performed.
- For the `search:box` and `advanced:button` elements, pagination is not reset when a new search is performed.
- The `widget:date-range` component uses a previously selected To date when reselecting dates.
- When using both tabs and saved searches, sometimes selecting a saved search had no effect.
- When doing partial updates using an Appkit Solr platform, the `id` field is modified incorrectly.

- The `suggester` parameter in the `SolrSuggestionsRetriever` service is ignored, which prevents specification of multiple suggester dictionaries.
- An OAuth filter conflicts with a Spring Security filter if the filters have the same name.
- Logging about floating licenses is not correct.
- It is impossible to use arrows to select typeahead results when a `search:result-list` element is nested inside an `instant:result-list` element.
- A dependency conflict exists between the Elastic 6 adaptor and the Media module. It is not possible to use them together.
- Track Clicks cannot calculate the offset if RPP (results per page) is not included in the query.
- Hierarchical facet filters are not being fully cleared.
- Attempting to connect to a Fusion data source using a query profile returns a 404 error if the Fusion collection has more than one shard.
- The draw function in Google Maps creates the longitude filter incorrectly.
- When using the `facet:dropdown` element and making more than one facet selection, incorrect facets are selected, highlighted, indicated in breadcrumbs, and used in queries.
- Calling `store` on the Lucene platform does not add new documents.
- New bounds are not used for a Google Map when a user clicks a filter.
- Documentation for the `collaboration:comment-list` tag mentions bookmarks instead of comments.
- When no LDAP groups are found for a user, an invalid LDAP query cause an exception to be thrown.
- Clicking outside a list of selectable fields displayed by a `search:fields` element does not hide the list of fields.

Date Format processor

The Date Format processor `DateFormatProcessor` lets a user format the display value of a date field.

How to enable `DateFormatProcessor`:

1. Add the file `data-format.conf` under `conf/processors/response`. In the file add:

```
name: twigkit.search.processors.response.DateFormatProcessor
format: date-format
```

For example:

```
name: twigkit.search.processors.response.DateFormatProcessor
format: dd-MM-yyyy
```

The default format is `yyyy-MM-dd`.

2. Add the processor to the relevant platform configuration. For example, in `conf/platforms/fusion/data.conf` add:


```
workflow: processors.response.date-format
```

When this platform is used to request search results, the Date Format processor is used to format dates in the response.

Fusion Analytics Catalog platform

A Fusion Analytics Catalog platform has been added to query against the Fusion Analytics Catalog collection. This platform translates Appkit queries into SQL statements which are executed against a Fusion query pipeline that includes an *Analytics Catalog Stage*. The Fusion Analytics Catalog platform uses the *Fusion Catalog API*.

The Analytics Catalog platform expects standard Fusion platform configuration parameters (for example, `host` and `collection`).

The current implementation only supports fields and facets parameters in the `<search:query>` tag.

Get started with the Fusion Analytics Catalog platform

1. Start Fusion Server: `bin/fusion start`
2. Start the Fusion SQL service: `bin/sql start`
3. Add a new platform configuration file, for example `conf/platforms/fusion/analytics.conf`. Include these properties:

```
name: twigkit.fusion.analytics.FusionAnalytics
resultIDField: ID-field
userName: Fusion-user
password: password
host: host-name
collection: Fusion-collection
```

For example:

```
name: twigkit.fusion.analytics.FusionAnalytics
resultIDField: id
userName: admin
password: password123
host: search.my.org
collection: analytics
```

4. Edit the configuration file `conf/services/api/fusion.conf`. Specify the catalog to use with the platform. The default catalog is `fusion`:

```
catalog-project: fusion-catalog
```

5. Use the platform in HTML pages. For example:

```
<search:platform var="platform" conf="platforms.fusion.analytics">
</search:platform> <search:query var="query" parameters="*"
fields="id,pdf_url_s,summary_t,published_dt pubdate,categories_s"
facets="categories_s,pdf_url_s"></search:query> <search:response var="response"
platform="platform" query="query"> <widget:spinner></widget:spinner>
</search:response>
```

Fusion App Studio

App Studio

App Studio is a collection of software and resources that helps you *build* modern, user-friendly apps to search and visualize your Fusion data. This includes:

- **Appkit** – The framework that drives App Studio Wizard and App Studio Enterprise.
- **App Studio Enterprise** – The standalone web application which you connect to your Fusion collection.
- **App Studio Wizard** – The app development wizard built in the Fusion 4.x UI.

App Studio requires a running instance of *Fusion Server*, or access to one of our *supported datasources*, with one or more collections of indexed data.

Looking to get started?

Download

Appkit

The Appkit framework provides a library of search application components encapsulated as simple building blocks that can be combined to create complex applications for search and information retrieval.

See the *reference docs* for more information about the Appkit framework.

Modules

Appkit's suite of search components is modular, which means you are able to get a working solution into place quickly using tried-and-tested components.

Read more about modules.

Tags

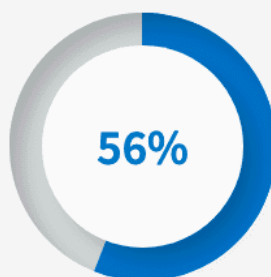
The App Studio markup language defines a set of HTML-like elements, or *tags*, for defining search queries to the backend and visualizing the search results and facets, handling all the complexities of connecting to Fusion.

For example, adding a radial progress bar is as simple including the following tag:

```
<progress:radial percentage="56"> </progress:radial>
```

This results in the following:

Default Progress Radial



Over 150 tags are provided in the Appkit framework.

Read more about tags.

Services

App Studio exposes a number of frontend JavaScript services which encapsulate reusable processes. These services are based on the AngularJS framework. There are also backend Java web services that perform the same functions as many of the frontend services, including platform search endpoints, platform series endpoint, query suggestion/typeahead services, csv/excel export, and facet search.

Read more about services.

Search Platforms

Appkit integrates with a number of search engines, data warehouses and web service endpoints, even surfacing information from multiple sources simultaneously. The platform abstraction covers both search and storage sides, meaning that Appkit will implement for each platform the necessary protocols to save or index content for any platform. A number of platform adapters are available out-of-the-box with Appkit, and *adding new ones* is a simple exercise.

Read more about search platforms.

Concepts

- *Appkit Concepts*
- *Appkit Configuration*
- *Extending Appkit*

Reference

- *App Studio Versions FAQ*
- *Accessing Data*
- *App Deployment*
- *FAQ*

How-to

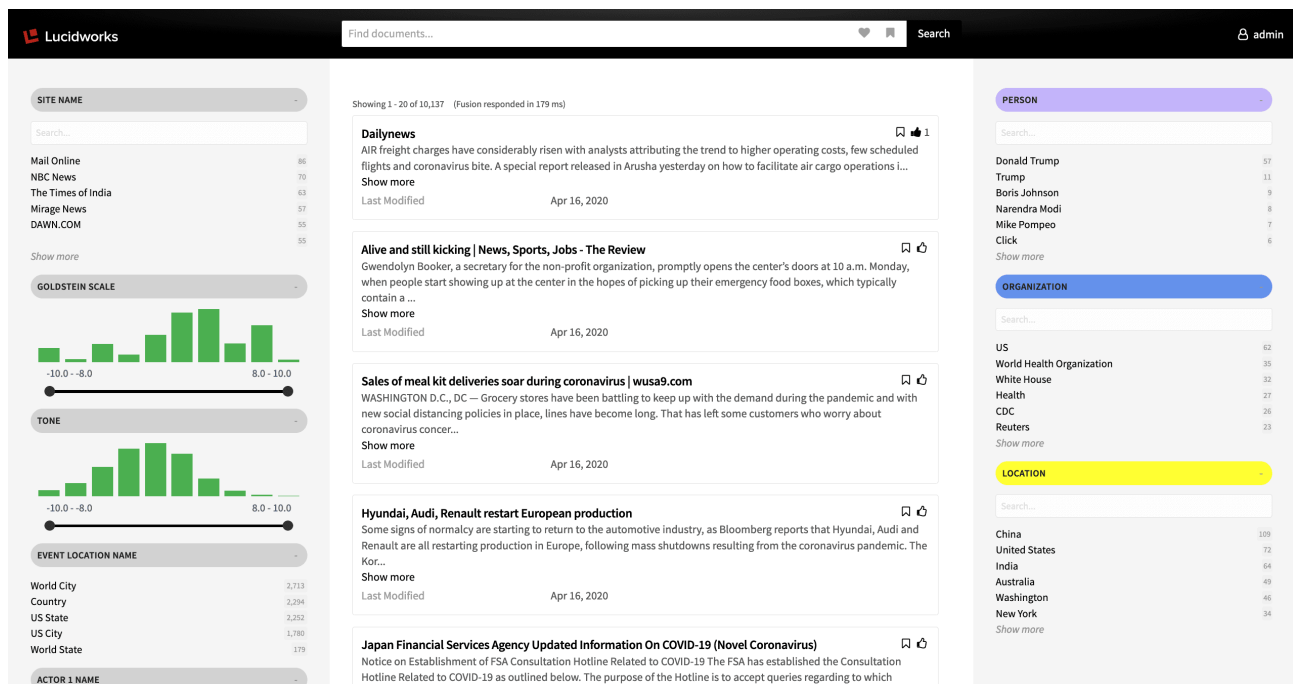
- *Upgrade Appkit in Existing Apps*

App Studio Enterprise

App Studio Enterprise is an Appkit based web application that connects your Fusion instance, or other *datasources*, to the *Appkit* design framework. It comes preloaded with the Appkit framework library and a starter search app template. This allows you to develop and preview your search application using your data before deploying it to a production environment.

Getting started only takes a few minutes and is as simple as *downloading App Studio Enterprise*, configuring the application to use your Fusion instance, and launching. See

the README file included in the download for full instructions.



How does this differ from App Studio Wizard?

Although both are powered by the Appkit framework, App Studio Enterprise is not integrated in the Fusion UI or packaged as a wizard. It is a standalone web application.

Concepts

- *App Studio Enterprise Concepts*
- *App Studio Enterprise Downloads*
- *Getting Started with App Studio Enterprise*

Reference

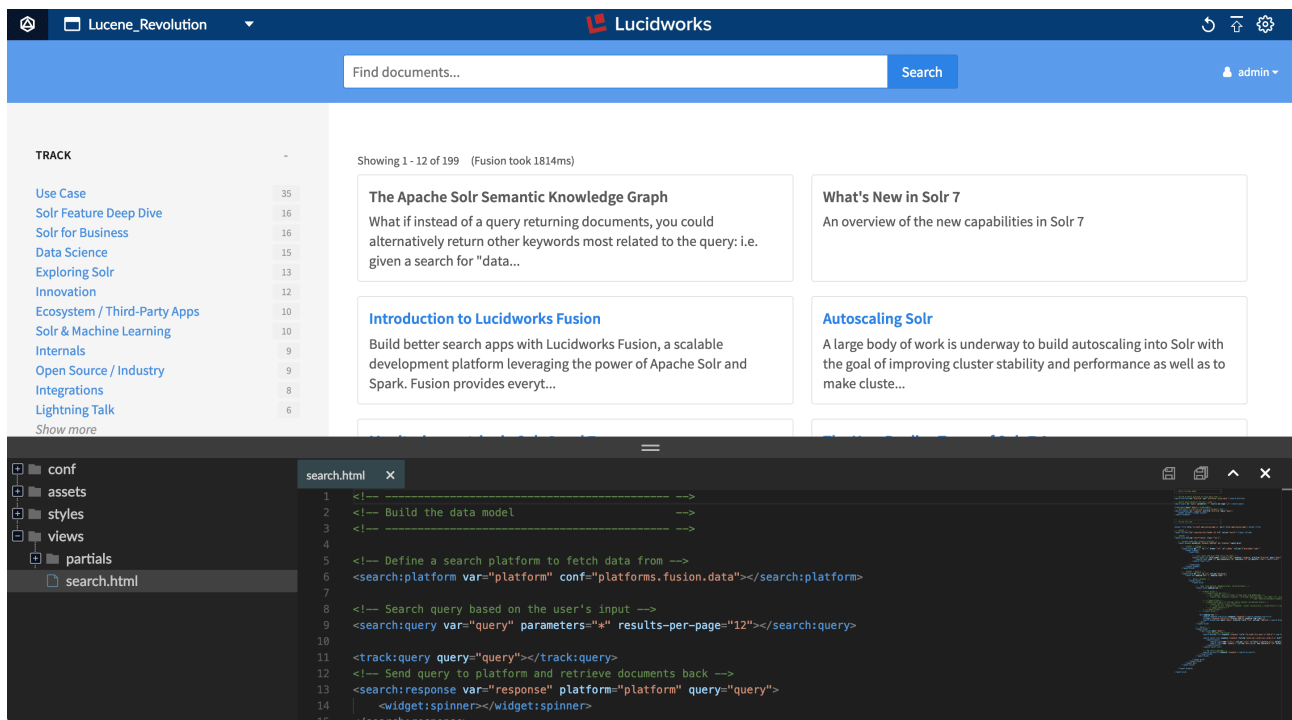
- *App Studio Versions FAQ*
- *FAQ*

How-to

- *Building from Source*
- *Deploy App Studio Enterprise*

App Studio Wizard

In Fusion 4.x, the App Studio Wizard is included in the Fusion UI. The wizard walks you through a step-by-step process for creating a basic search application using your data from Fusion. From there, you can continue customizing your search app with elements of the Appkit framework. A code editor is provided in the UI so you can preview your changes in real-time:



Starting in Fusion 5, App Studio is no longer included in the Fusion UI. However, you can use *App Studio Enterprise* to create apps for Fusion 5.

Concepts

- *Layout and Styling*
- *datasources*

Reference

- *App Studio Versions FAQ*
- *Licensing*
- *FAQ*

How-to

- *Develop and Deploy with App Studio*
- *Use the Code Editor*
- *Add a Sidebar with Facets*
- *Configure Autocomplete*

Versions FAQ

How are these products versioned?

- **Appkit** is versioned according to updates to the code base.
- **App Studio Enterprise** versions are loosely tied to the version of the Appkit framework it includes. An update from **4.8.0** to **4.9.0** may not include any changes to App Studio Enterprise.

What about compatibility?

- Any version of **Appkit** can be used with Fusion 4.x. However, only Appkit 4.6.0+ can be used with Fusion 5.y.
- **App Studio Enterprise** can be used with any version of Fusion or Appkit.

Can these products be updated independently?

- **Appkit** can be *updated* independently, assuming there are no version *compatibility* issues.
- **App Studio Enterprise** is released when a new version of Appkit is released. As a result, App Studio Enterprise itself does not need to be updated. To move to the latest code base, simply update the Appkit version within the project.

See the *FAQ* section for more frequently asked questions.