



APACHE WICKET

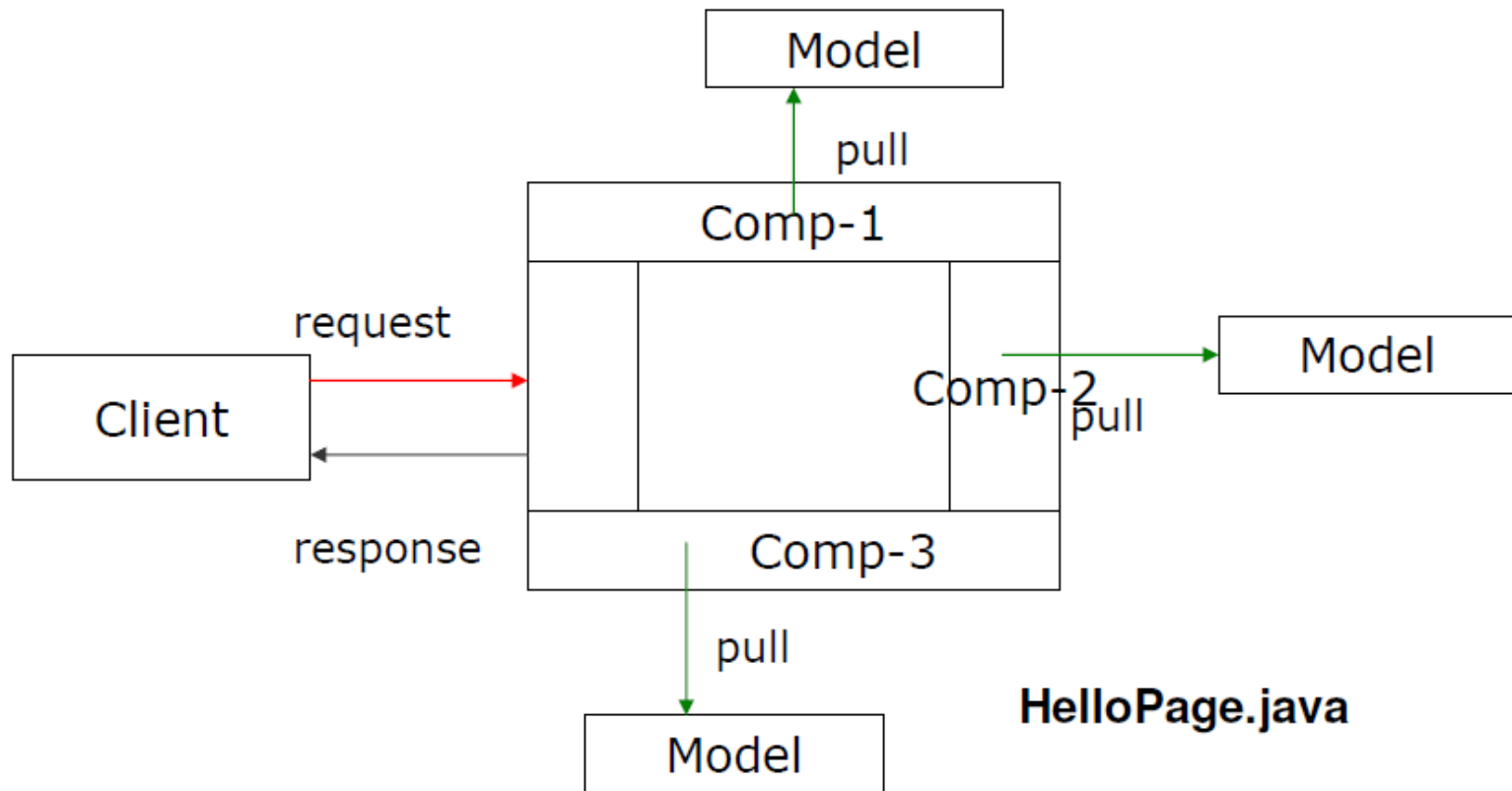
Agenda

- Que es Wicket?
- Core concepts of Wicket
- Desarrollando un custom component

Apache Wicket

- Framework de aplicaciones web
- Open Source (ASF)
- Orientado a componentes
- Clara separacion entre el Markup y el Codigo
- Solo se necesita saber Java y HTML 😊

Orientado a componentes



Features

- Page Composition
- Localización (Internacionalización)
- Integración
 - Spring
 - Hibernate
 - Jasper Reports
 - ...
- Variedad de componentes
 - sortable, filterable, pageable, data aware tables
 - date picker, rich text editor, Google Maps
 - tabbed panel, navigation, tree, wizard

Features

- State management
 - type safe sessions
- Clustering
- Transparente ante “back button problem”
- Double submit strategies
- Testing
 - JUnit testing
- Ajax
 - Podemos usar Ajax sin tener que escribir JavaScript, Dojo, Scriptaculous, ...
- Seguridad a nivel componente

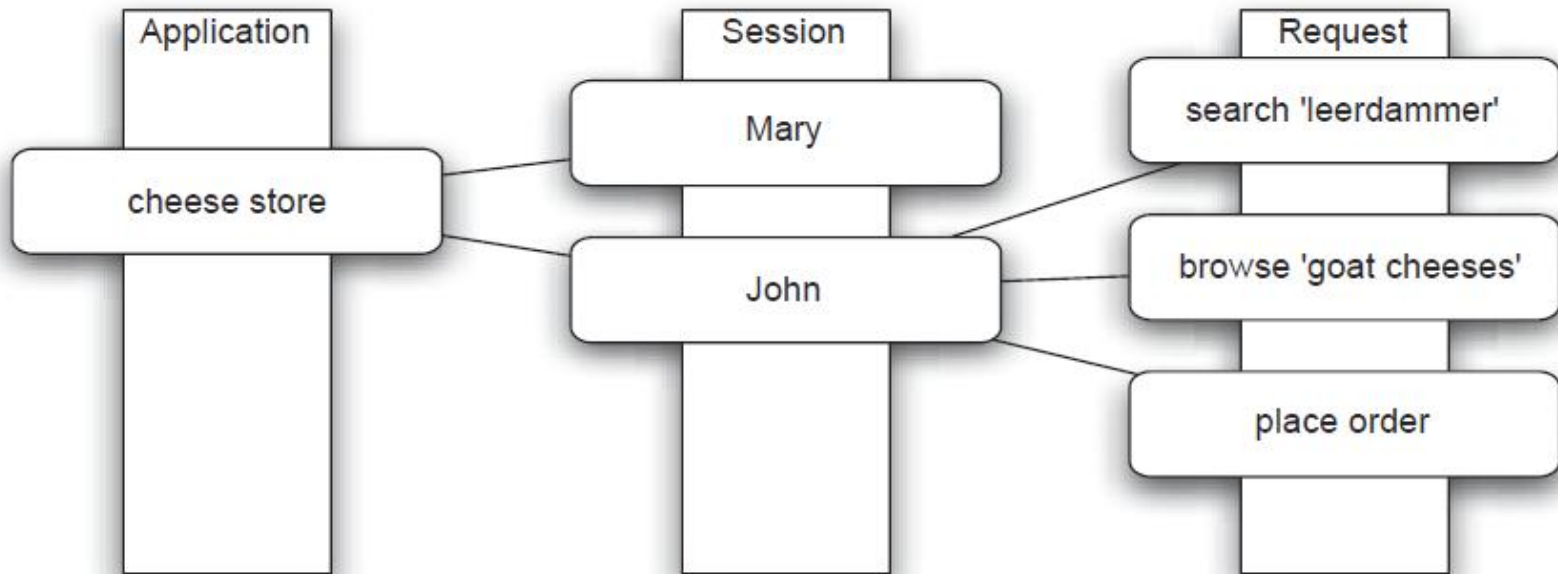
Agenda

- Que es Wicket?
- Core concepts of Wicket
- Desarrollando un custom component

Core concepts of Wicket

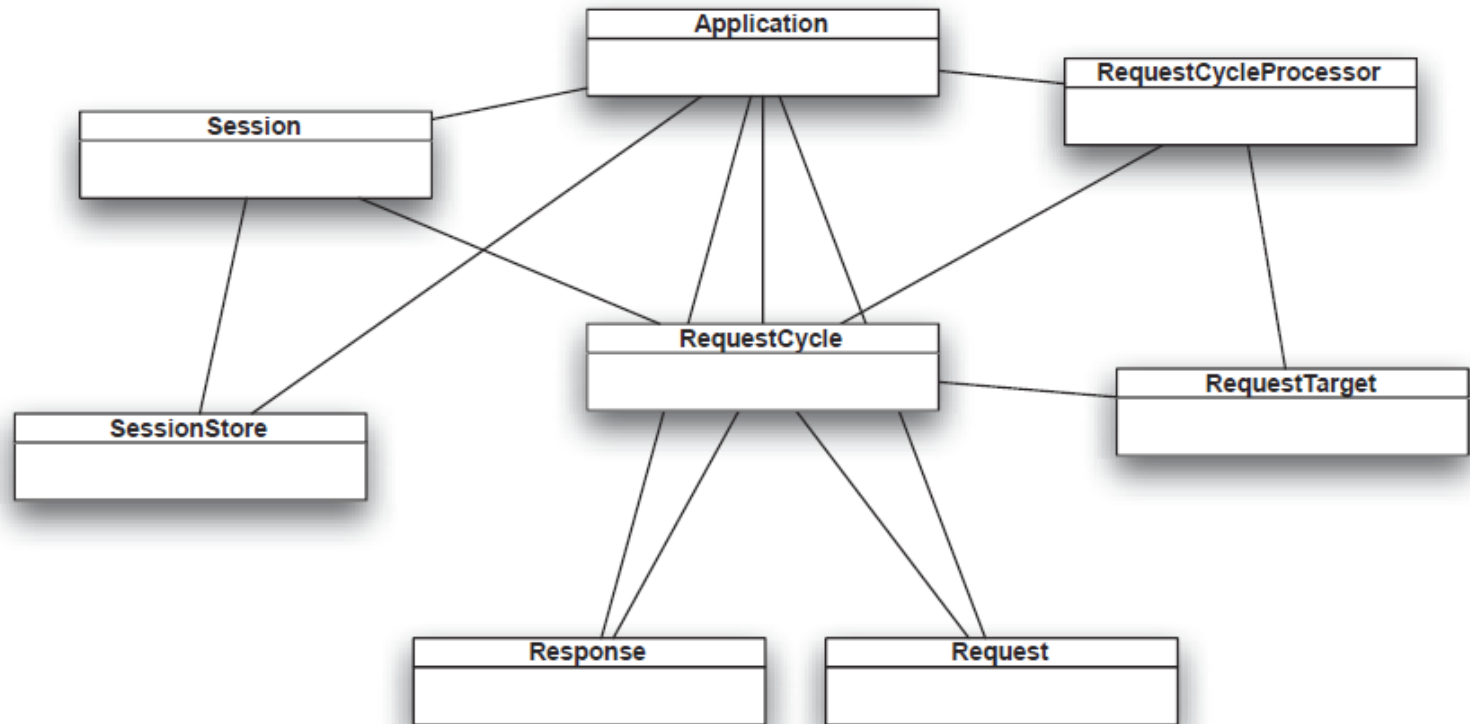
- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Core concepts of Wicket



One application handles multiple sessions, each of which handles multiple requests over its lifetime.

Core concepts of Wicket



Important classes for handling requests. The **Application** class is responsible for the instantiation of most objects.

Application

- Punto de inicio de la aplicación web
- Configuración
 - web.xml
- Factories para:
 - Session
 - RequestCycle
 - Security
 - ...

Core concepts of Wicket

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Session

- Es persistida en HttpSession
- ¿Qué info hay en la session?
 - Locale, Client info (browser vendor and version)
 - Datos propios
 - Usuario logueado
 - ..
- Custom session

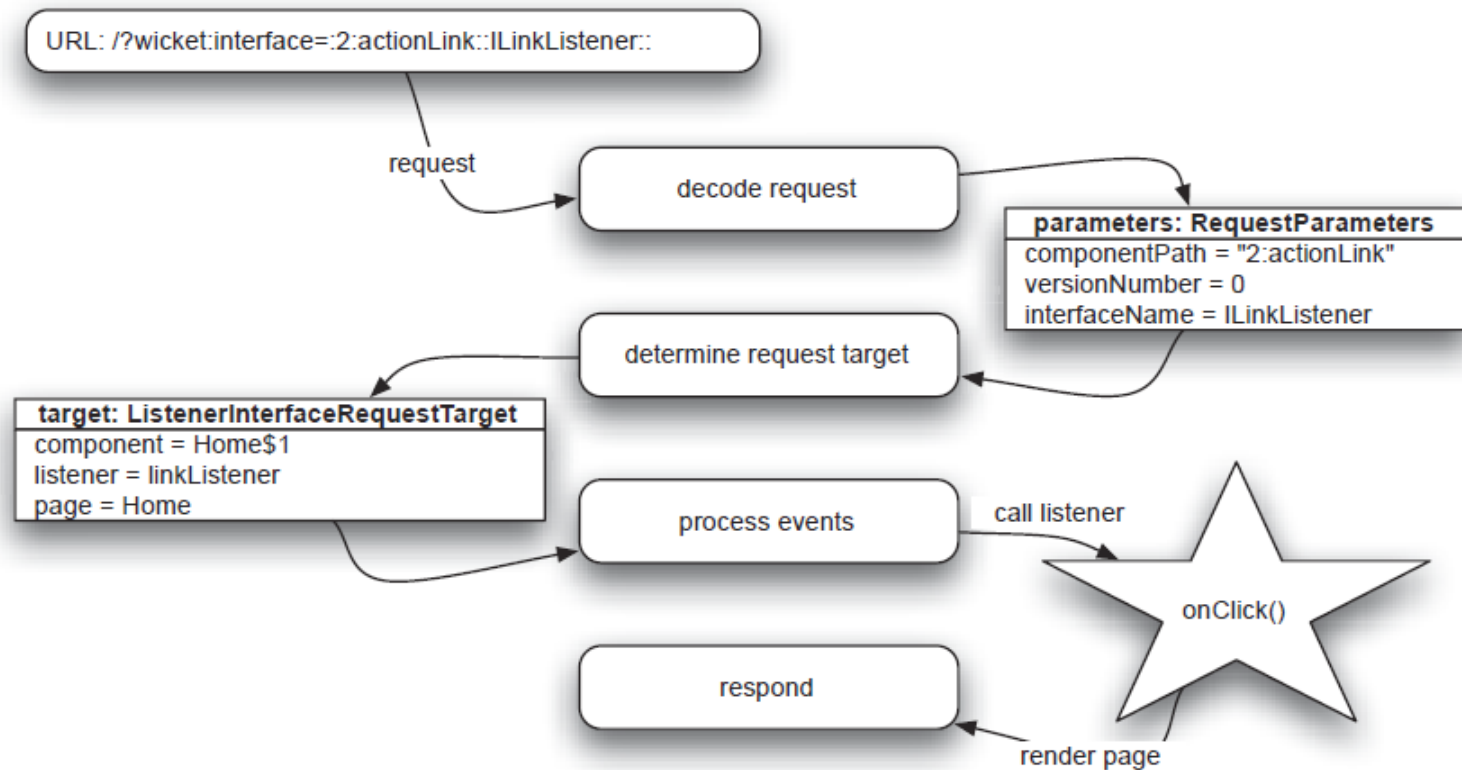
Session - custom session

```
class ApplicationSession extends WebSession {  
    private User loggedInUser;  
    public User getLoggedInUser() { ... }  
    public void setLoggedInUser (User cart) { ... }  
}  
appSession.setLoggedInUser(new User());  
...  
User loggedInUser = mysession. getLoggedInUser();  
Permission[] permissions = loggedInUser.getPermissions();
```

Core concepts of Wicket

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

RequestCycle



Request processing is performed in four steps: decode request, determine request target, process events, and respond.

Core concepts of Wicket

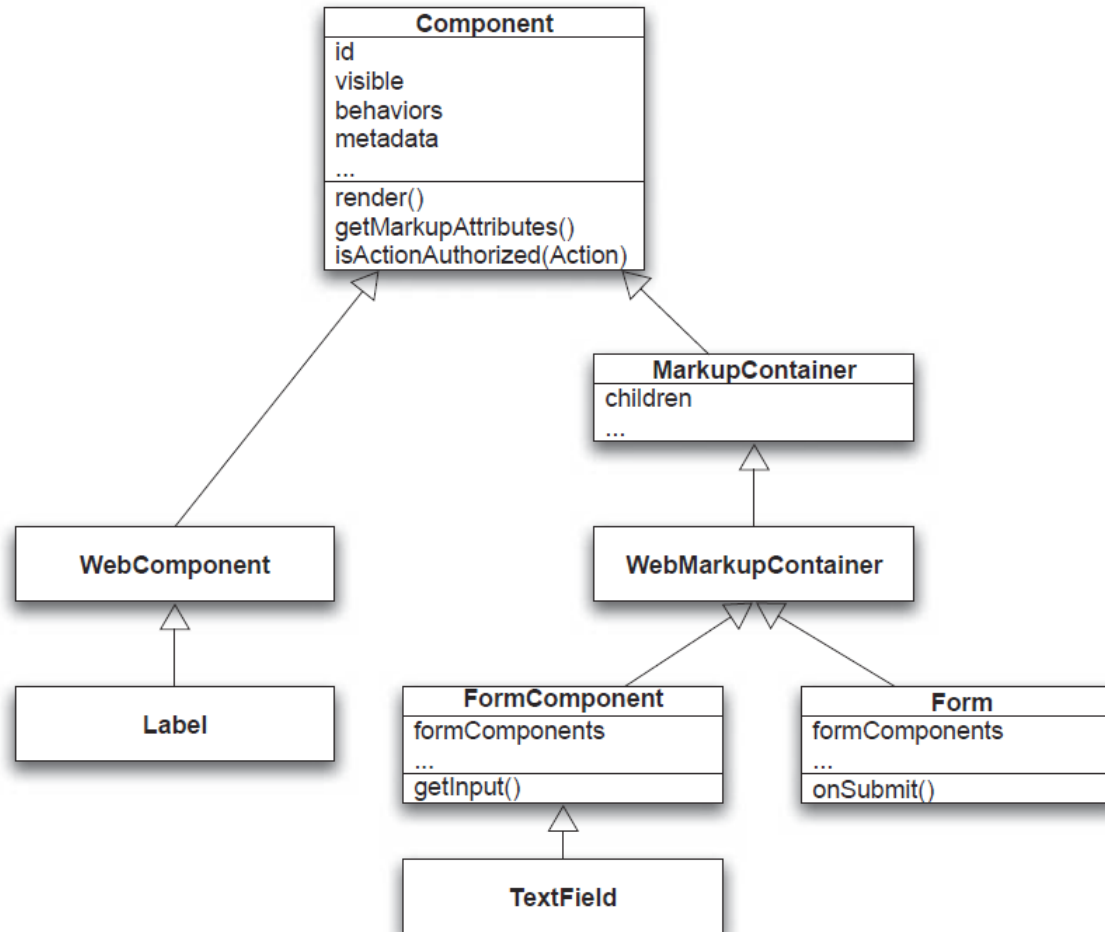
- Application
- Session
- RequestCycle
- **Components**
- Behaviors
- Models

Component

- org.apache.wicket.Component
 - *Label*
 - *MultiLineLabel*
 - *TextField*
 - *PasswordTextField*
 - *Image*
 - *Link*
 - *Tree*
 - *BookmarkablePageLink*
 - *Panel*
 - *ListView*
 - *Loop*
 - *PagingNavigator*
 - *ImageMap*
 - *Button*
 - *Ajax...*
 - *Sorting, paging repeaters*
 - *Wizard*
 - *DatePicker*



Component



A sample of Wicket's component hierarchy

Component

- Un componente es identificado en el markup con `wicket:id`
 - Html:

```
<h1 wicket:id="msg">Gets replaced</h1>
```

- Java:

```
new Label("msg", "Hello, World!");
```

- Final (wicket tags puede se el eliminados):

```
<h1>Hello, World!</h1>
```



Component

- Componentes que pueden tener su propio markup file
 - Page
 - Panel
 - Border
- Los archivos java, markup deben pertenecer a mismo package y tener el mismo nombre

Core concepts of Wicket

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Behaviors

- Behaviors son plug-ins para Componentes
- Pueden modificar el markup de los componentes
 - Atributos
 - Eventos de javascript
 - Ajax

Behaviors

```
item.add(new AbstractBehavior() {  
    public void onComponentTag(  
        Component component, ComponentTag tag) {  
        String css = (((Item)component).getIndex() % 2 == 0)  
            ? "even" : "odd";  
        tag.put("class", css);  
    }  
});
```

Output:

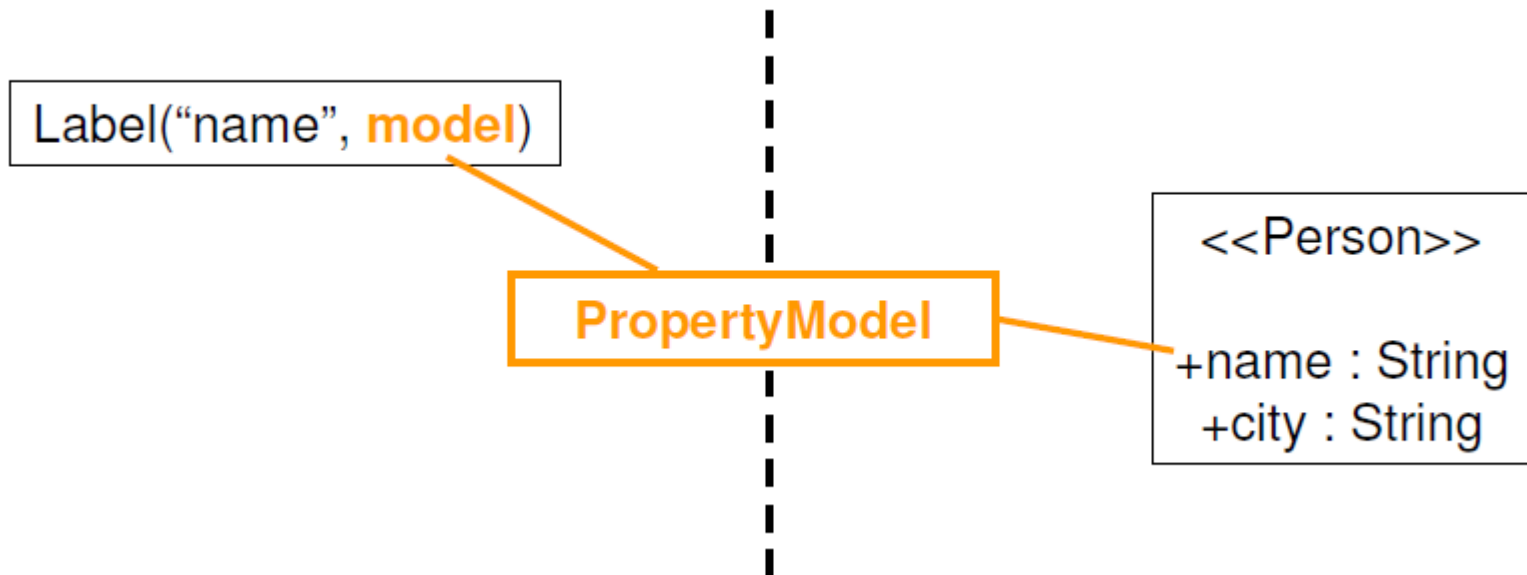
```
<tr class="odd">...</tr>  
<tr class="even">...</tr>
```

Core concepts of Wicket

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Models

- Los modelos asocian nuestros POJO's con los componentes Wicket



Models

- Lazy binding

- No actualiza modelo

```
new TextField("txt", new Model(person.getName()))
```

- Provoca null pointers:

```
new Label("street", new Model(person.getAddress().getStreet()))
```

- Solución:

- PropertyModel:

```
new TextField("txt", new PropertyModel(person, "name"))
```

```
new Label("street", new PropertyModel(person, "address.street"))
```

Agenda

- Que es Wicket?
- Core concepts of Wicket
- Desarrollando un custom component

Referencias

- <http://wicket.apache.org>

