Arenadata™ Analytic Workspace

Версия - master

Руководство по работе с Apache Zeppelin

Оглавление

В документе приведено общее описание Apache Zeppelin, руководство по установке, конфигурации, использовании интерфейса и интерпретаторах.

Документ может быть полезен администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих внедрение и сопровождение кластеров Arenadata Hadoop и Arenadata DB.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Общий обзор

Открытый веб-блокнот для интерактивной аналитики данных

Apache Zeppelin – это новый многофункциональный веб-блокнот, обеспечивающий считывание, анализ и визуализацию данных, их обмен и взаимодействие с **Hadoop** и **Spark** (Рис.1.1.).

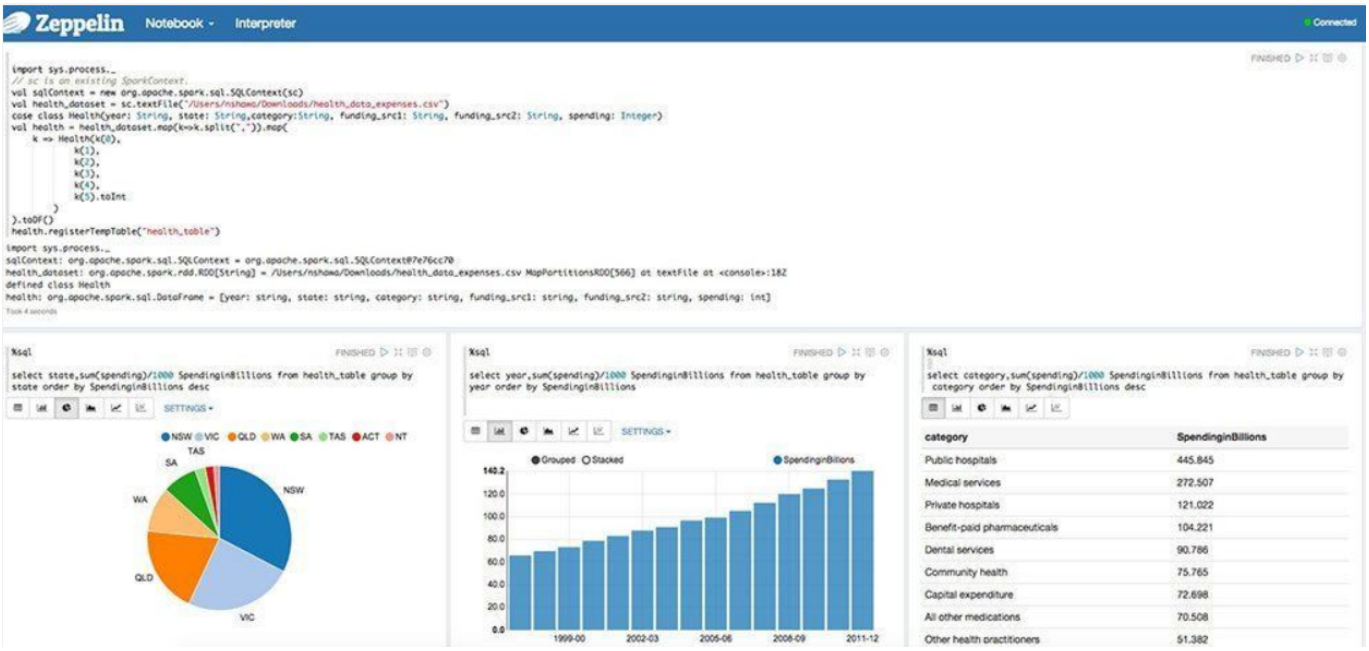

Рис.1.1.: Apache Zeppelin

Интерактивные браузерные блокноты позволяют инженерам данных, аналитикам и ученым в области данных более продуктивно выполнять работу, благодаря совместному использованию кода данных, его разработке, организации и выполнению, а также благодаря визуализации результатов без необходимости обращения к командной строке или к компонентам кластера. Блокноты обеспечивают пользователям не только выполнение задач, но и интерактивную работу с долго выполняющимися потоками операций.

Apache Zeppelin – это новый веб-блокнот, который предоставляет функции поиска, визуализации, совместного использования и функциональное взаимодействие с **Apache Spark**. В него встроена интеграция со **Spark**, что избавляет от необходимости создания отдельного модуля, плагина или библиотеки, и это дает следующие преимущества:

- Автоматическое создание *SparkContext* и *sqlcontext*;
- Загрузка jar-зависимостей из локальной файловой системы или репозитория *maven* во время выполнения задачи;
- Возможность отмены задания и отображение хода его выполнения.

Apache Zeppelin поддерживает **Python**, но при этом концепция интерпретатора блокнота позволяет подключать любой язык/фрейворк обработки данных в **Zeppelin**. В настоящее время **Zeppelin** поддерживает множество интерпретаторов, например, такие как **Scala**, **Hive**, **SparkSQL**, **Shell** и **Markdown** (Рис.1.2.).

Рис.1.2.: Интерпретаторы

Некоторые базовые диаграммы уже включены в **Apache Zeppelin**, но визуализация не ограничивается запросом **Spark SQL** и любой результат с любого языка может быть распознан и визуализирован (Рис.1.3.).

Apache Zeppelin агрегирует значения и отображает их в сводной диаграмме с простым перемещением drag-and-drop. Можно легко создать диаграмму с несколькими агрегированными значениями, в том числе: сумма, количество, среднее, минимальное, максимальное (Рис.1.4.).

Также **Apache Zeppelin** может динамически создавать некоторые формы ввода в блокноте пользователя (Рис.1.5.).

Поиск данных, их анализ, отчетность и визуализация являются ключевыми компонентами рабочего процесса в области данных. **Zeppelin** предоставляет “Modern Data Science Studio” (“Современную научную студию данных”), которая поддерживает **Spark** и **Hive** из коробки. Фактически **Zeppelin** поддерживает несколько языков, которые в свою очередь имеют поддержку растущей экосистемы источников данных. Блокноты **Zeppelin** позволяют ученым в области данных в реальном времени создавать и выполнять небольшие фрагменты кода.

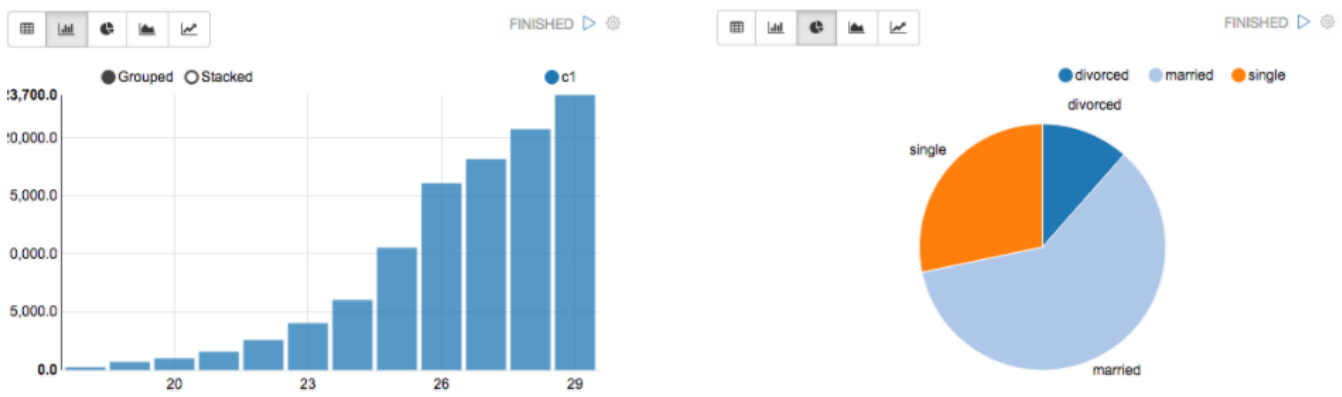

Рис.1.3.: Визуализация данных

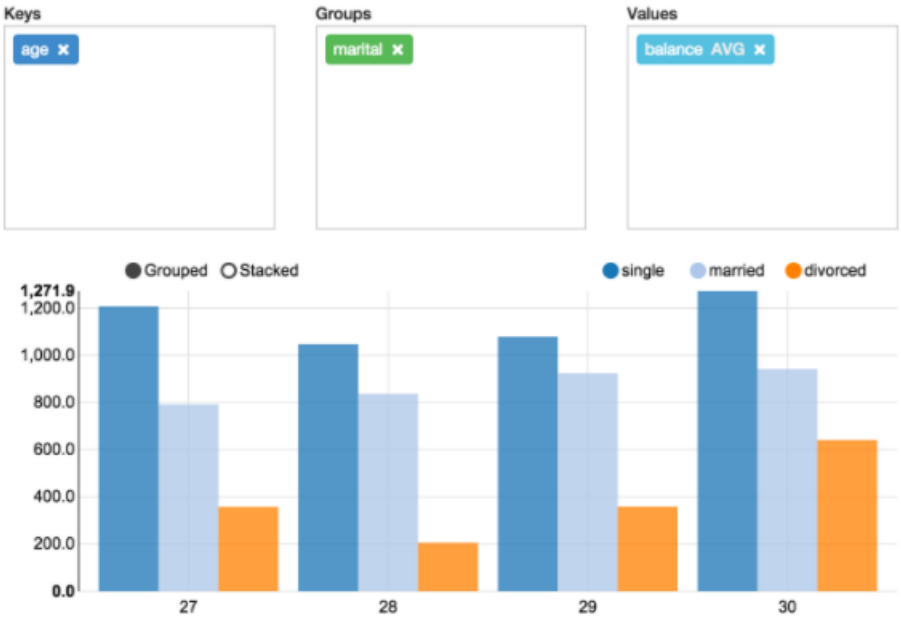

Рис.1.4.: Сводная диаграмма

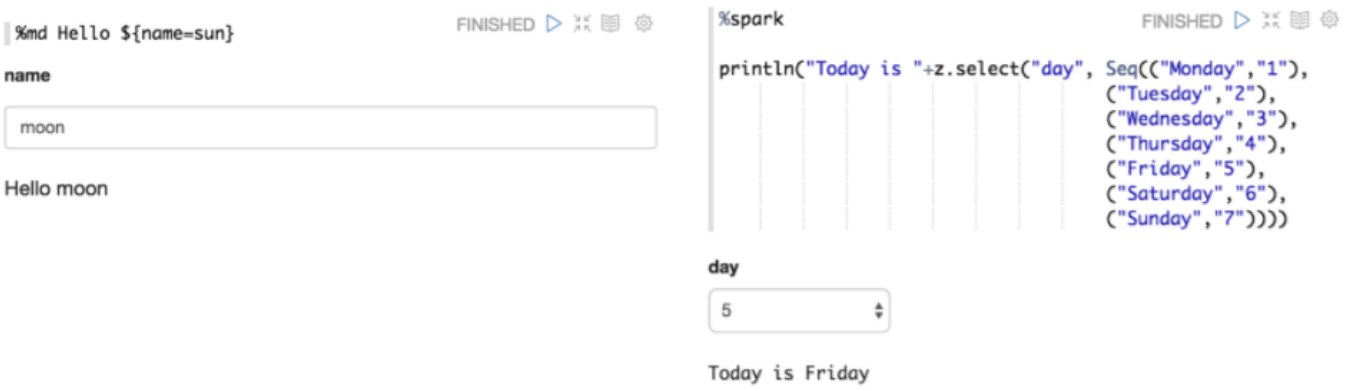

Рис.1.5.: Формы ввода в блокноте

URL-адресом блокнота можно поделиться между сотрудниками. В таком случае **Apache Zeppelin** транслирует любые изменения в реальном времени точно так же, как при работе в **Google docs**. Но данный URL-адрес отображает только результат, страница не содержит никаких меню и кнопок для редактирования. Кроме того, при завершении работы с блокнотом можно создать отчет и при необходимости распечатать его или экспортировать (Рис.1.6.).

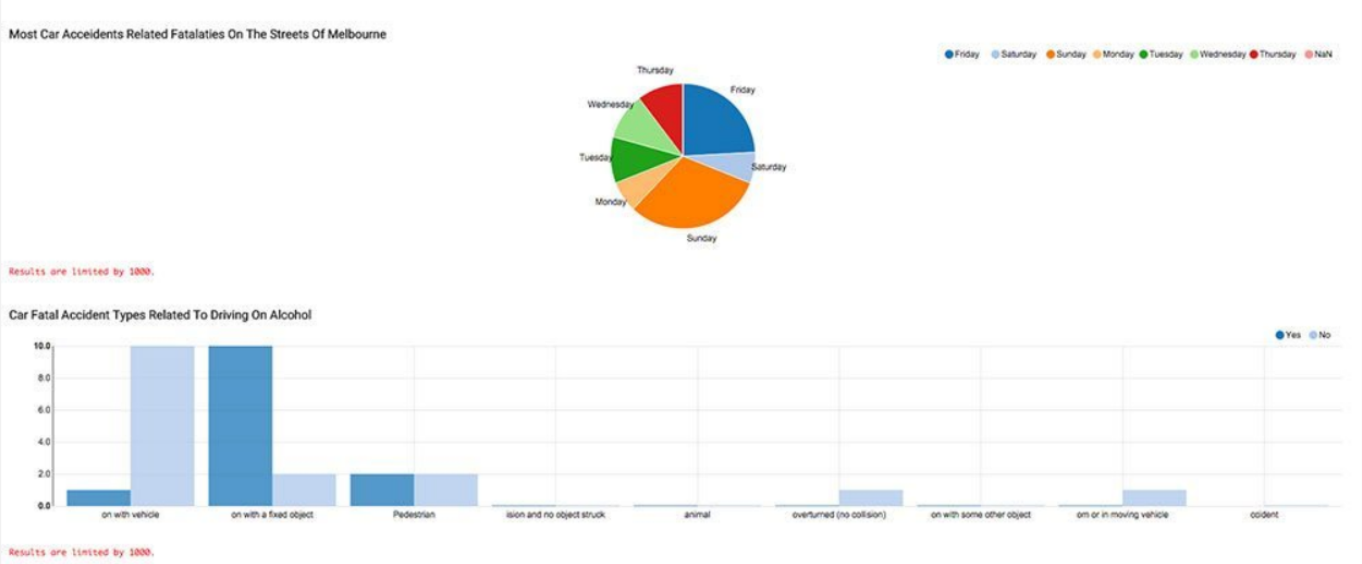

Рис.1.6.: Отчет о работе в Apache Zeppelin

В **Arenadata** мы считаем, что **Spark** и **Hadoop** идеально сочетаются. И что **Zeppelin** является ключевым компонентом для ускорения решений в области науки о данных.

Глава 2

Установка интерпретатора

Apache Zeppelin обеспечивает механизм установки интерпретатора, благодаря загруженному с **Zeppelin** бинарному пакету *netinst*, также можно установить другие сторонние интерпретаторы.

- *Установка интерпретаторов, разрабатываемых сообществом*
- *Сторонние интерпретаторы*
- *Доступные интерпретаторы, разрабатываемые сообществом*

Important: После установки интерпретаторов необходимо перезапустить Apache Zeppelin, выполнить настройку интерпретатора и привязать его к блокноту (Интерпретаторы в Apache Zeppelin)

2.1 Установка интерпретаторов, разрабатываемых сообществом

Apache Zeppelin позволяет управлять несколькими интерпретаторами одновременно, объединяя их в группы, перечень которых представлен в разделе *Доступные интерпретаторы, разрабатываемые сообществом*. Если бинарный пакет *netinst* уже загружен, следует в зависимости от необходимых интерпретаторов выполнить соответствующие команды.

- **Установка предоставляемых сообществом интерпретаторов**

Для установки всех интерпретаторов, которые предоставляются сообществом, необходимо выполнить следующую команду:

```
./bin/install-interpreter.sh --all
```

- **Установка выборочных интерпретаторов**

Для установки отдельно выбранных интерпретаторов необходимо воспользоваться следующей командой:

```
./bin/install-interpreter.sh --name md,shell,jdbc,python
```

Для получения полного списка интерпретаторов, разрабатываемых сообществом, следует выполнить команду:

```
./bin/install-interpreter.sh --list
```

- **Установка интерпретатора с версией языка Scala 2.10**

Zeppelin поддерживает **Scala 2.10** и **2.11** для нескольких интерпретаторов, параметры которых приведены в таблице.

Таблица2.1.: Параметры интерпретаторов для Scala

Параметр –name	Параметр –artifact для Scala 2.10	Параметр –artifact для Scala 2.11
cassandra	org.apache.zeppelin:zeppelin-cassandra_2.10:0.7.3	org.apache.zeppelin:zeppelin-cassandra_2.11:0.7.3
flink	org.apache.zeppelin:zeppelin-flink_2.10:0.7.3	org.apache.zeppelin:zeppelin-flink_2.11:0.7.3
ignite	org.apache.zeppelin:zeppelin-ignite_2.10:0.7.3	org.apache.zeppelin:zeppelin-ignite_2.11:0.7.3
scio	org.apache.zeppelin:zeppelin-scio_2.10:0.7.3	org.apache.zeppelin:zeppelin-scio_2.11:0.7.3
spark	org.apache.zeppelin:zeppelin-spark_2.10:0.7.3	org.apache.zeppelin:zeppelin-spark_2.11:0.7.3

При установке интерпретатора только с параметром *–name*, программа установки загружает по умолчанию интерпретатор с поддержкой версии языка **Scala 2.11**. Для указания иной версии **Scala** следует добавить параметр *–artifact*. Далее приведен пример установки интерпретатора *flink* с версией языка **Scala 2.10**:

```
./bin/install-interpreter.sh --name flink --artifact org.apache.zeppelin:zeppelin-flink_2.10:0.7.3
```

- **Установка интерпретатора Spark, поддерживающего версию языка Scala 2.10**

Дистрибутив **Spark** до версии *1.6.2* поддерживает **Scala 2.10**. Если *SPARK_HOME* указывает на версию **Spark** ниже *2.0.0*, необходимо скачать интерпретатор **Spark** с версией языка **Scala 2.10**. Для этого следует выполнить команду:

```
rm -rf ./interpreter/spark
./bin/install-interpreter.sh --name spark --artifact org.apache.zeppelin:zeppelin-spark_2.10:0.7.3
```

2.2 Сторонние интерпретаторы

Сторонние интерпретаторы из репозитория **maven** можно установить при помощи следующей команды:

```
./bin/install-interpreter.sh --name interpreter1 --artifact groupId1:artifact1:version1
```

Данная команда загружает артефакт **maven** *groupId1:artifact1:version1* и все его зависимости в каталог *interpreter/interpreter1*.

Установка нескольких сторонних интерпретаторов осуществляется командой, где аргументы *–name* и *–artifact* указываются списком через запятую:

```
./bin/install-interpreter.sh --name interpreter1,interpreter2 --artifact␣
↪groupId1:artifact1:version1,groupId2:artifact2:version2
```

2.3 Доступные интерпретаторы, разрабатываемые сообществом

Список интерпретаторов, предоставляемых сообществом, приведен в таблице. Также данную информацию можно найти в файле *conf/interpreter-list*.

Таблица2.2.: Предоставляемые сообществом интерпретаторы

Параметр –name	Maven Artifact	Описание
alluxio	org.apache.zeppelin:zeppelin-alluxio:0.7.3	Интерпретатор Alluxio
angular	org.apache.zeppelin:zeppelin-angular:0.7.3	Просмотр HTML и AngularJS
beam	org.apache.zeppelin:zeppelin-beam:0.7.3	Интерпретатор Beam
bigquery	org.apache.zeppelin:zeppelin-bigquery:0.7.3	Интерпретатор BigQuery
cassandra	org.apache.zeppelin:zeppelin-cassandra_2.11:0.7.3	Интерпретатор Cassandra, построенный с помощью Scala 2.11
elasticsearch	org.apache.zeppelin:zeppelin-elasticsearch:0.7.3	Интерпретатор Elasticsearch
file	org.apache.zeppelin:zeppelin-file:0.7.3	Интерпретатор файлов HDFS
flink	org.apache.zeppelin:zeppelin-flink_2.11:0.7.3	Интерпретатор Flink, построенный с помощью Scala 2.11
hbase	org.apache.zeppelin:zeppelin-hbase:0.7.3	Интерпретатор Hbase
ignite	org.apache.zeppelin:zeppelin-ignite_2.11:0.7.3	Интерпретатор Ignite, построенный с помощью Scala 2.11
jdbc	org.apache.zeppelin:zeppelin-jdbc:0.7.3	Интерпретатор Jdbc
kylin	org.apache.zeppelin:zeppelin-kylin:0.7.3	Интерпретатор Kylin
lens	org.apache.zeppelin:zeppelin-lens:0.7.3	Интерпретатор Lens
livy	org.apache.zeppelin:zeppelin-livy:0.7.3	Интерпретатор Livy
md	org.apache.zeppelin:zeppelin-markdown:0.7.3	Поддержка Markdown
pig	org.apache.zeppelin:zeppelin-pig:0.7.3	Интерпретатор Pig
postgresql	org.apache.zeppelin:zeppelin-postgresql:0.7.3	Интерпретатор Postgresql
python	org.apache.zeppelin:zeppelin-python:0.7.3	Интерпретатор Python
scio	org.apache.zeppelin:zeppelin-scio_2.11:0.7.3	Интерпретатор Scio, построенный с помощью Scala 2.11
shell	org.apache.zeppelin:zeppelin-shell:0.7.3	Команда Shell

Глава 3

Конфигурация Apache Zeppelin

3.1 Свойства Apache Zeppelin

Существует две локации, в которых можно настроить **Apache Zeppelin**:

- Переменные окружения могут быть заданы в *conf/zeppelin-env.sh* (*confzeppelin-env.cmd* для ОС Windows);
- Свойства Java могут быть определены в *conf/zeppelin-site.xml*.

В случае если настроены обе локации, переменные окружения будут приоритетными.

Таблица3.1.: Параметры Zeppelin, их значения и описание

Параметр Zeppelin	Значение и описание
• zeppelin-env.sh: ZEPPELIN_PORT • zeppelin-site.xml: zeppelin.server.port	*8080* Порт сервера Zeppelin. Примечание: необходимо убедиться, что не используется тот же порт, что для разработки веб-приложений Zeppelin (по умолчанию: 9000)
• zeppelin-env.sh: ZEPPELIN_SSL_PORT • zeppelin-site.xml: zeppelin.server.ssl.port	*8443* ssl порт сервера Zeppelin (используется, когда значение ssl свойства установлено true)
• zeppelin-env.sh: ZEPPELIN_MEM • zeppelin-site.xml: N/A	*-Xmx1024m -XX:MaxPermSize=512m* Параметры JVM mem

Параметр Zeppelin	Значение и описание
• zeppelin-env.sh: ZEPPELIN_INTP_MEM • zeppelin-site.xml: N/A	*ZEPPELIN_MEM* Параметры JVM mem для процесса интерпретатора
• zeppelin-env.sh: ZEPPELIN_JAVA_OPTS • zeppelin-site.xml: N/A	Параметры JVM
• zeppelin-env.sh: ZEPPELIN_ALLOWED_ORIGINS • zeppelin-site.xml: zeppelin.server.allowed.origins	*символ "*"* Позволяет разделяя знаком запятой перечислить разрешенные источники для REST и websockets подключений. Например, http://localhost:8080
• zeppelin-env.sh: N/A • zeppelin-site.xml: zeppelin.anonymous.allowed	*true* Вход для неавторизованного пользователя разрешен по умолчанию
• zeppelin-env.sh: ZEPPELIN_SERVER_CONTEXT_PATH • zeppelin-site.xml: zeppelin.server.context.path	*символ "/"* Относительный путь веб-приложения
• zeppelin-env.sh: ZEPPELIN_SSL • zeppelin-site.xml: zeppelin.ssl	*false*
• zeppelin-env.sh: ZEPPELIN_SSL_CLIENT_AUTH • zeppelin-site.xml: zeppelin.ssl.client.auth	*false*
• zeppelin-env.sh: ZEPPELIN_SSL_KEYSTORE_PATH • zeppelin-site.xml: zeppelin.ssl.keystore.path	*keystore*
• zeppelin-env.sh: ZEPPELIN_SSL_KEYSTORE_TYPE • zeppelin-site.xml: zeppelin.ssl.keystore.type	*JKS*
• zeppelin-env.sh: ZEPPELIN_SSL_KEYSTORE_PASSWORD • zeppelin-site.xml: zeppelin.ssl.keystore.password	

Параметр Zeppelin	Значение и описание
• zeppelin-env.sh: ZEPPELIN_SSL_KEY_MANAGER _PASSWORD • zeppelin-site.xml: zeppelin.ssl.key.manager.password	
• zeppelin-env.sh: ZEPPELIN_SSL_TRUSTSTORE_PATH • zeppelin-site.xml: zeppelin.ssl.truststore.path	
• zeppelin-env.sh: ZEPPELIN_SSL_TRUSTSTORE_TYPE • zeppelin-site.xml: zeppelin.ssl.truststore.type	
• zeppelin-env.sh: ZEPPELIN_SSL_TRUSTSTORE _PASSWORD • zeppelin-site.xml: zeppelin.ssl.truststore.password	
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_HOMESCREEN • zeppelin-site.xml: zeppelin.notebook.homescreen	Отображение идентификаторов блокнотов на рабочем столе Apache Zeppelin. Например, 2A94M5J1Z
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_HOMESCREEN _HIDE • zeppelin-site.xml: zeppelin.notebook.homescreen.hide	*false* Скрытие идентификатора блокнота, установленного в ZEPPELIN_NOTEBOOK_HOMESCREEN на рабочем столе Apache Zeppelin
• zeppelin-env.sh: ZEPPELIN_WAR_TEMPDIR • zeppelin-site.xml: zeppelin.war.tempdir	*webapps* Расположение временного каталога jetty
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_DIR • zeppelin-site.xml: zeppelin.notebook.dir	*notebook* Каталог root, в котором хранятся каталоги блокнотов
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_STORAGE • zeppelin-site.xml: zeppelin.notebook.storage	*org.apache.zeppelin.notebook.repo. GitNotebookRepo* Список мест хранения блокнотов, перечисленный через запятую
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_ONE_WAY_SYNC • zeppelin-site.xml: zeppelin.notebook.one.way.sync	*false* Если есть несколько мест для хранения блокнотов, следует ли рассматривать первое как единственное?

Параметр Zeppelin	Значение и описание
• zeppelin-env.sh: ZEPPELIN_NOTEBOOK_PUBLIC • zeppelin-site.xml: zeppelin.notebook.public	*true* Сделать блокнот общедоступным по умолчанию при создании или импортировании (при указании только владельцев). Если установлено значение false, необходимо добавить пользователей, для которых доступно чтение и редактирование, таким образом делая блокнот недоступным и невидимым для пользователей, не имующих прав на него
• zeppelin-env.sh: ZEPPELIN_INTERPRETERS • zeppelin-site.xml: zeppelin.interpreters	*org.apache.zeppelin.spark.SparkInterpreter, org.apache.zeppelin.spark.PySparkInterpreter, org.apache.zeppelin.spark.SparkSqlInterpreter, org.apache.zeppelin.spark.DepInterpreter, org.apache.zeppelin.markdown.Markdown, org.apache.zeppelin.shell.ShellInterpreter, ...* Конфигурации (классы) интерпретатора с разделителями-запятыми. Примечание: это свойство устарело с Zeppelin-0.6.0 и не будет поддерживаться Zeppelin-0.7.0
• zeppelin-env.sh: ZEPPELIN_INTERPRETER_DIR • zeppelin-site.xml: zeppelin.interpreter.dir	*interpreter* Каталог интерпретатора
• zeppelin-env.sh: ZEPPELIN_INTERPRETER_DEP _MVNREPO • zeppelin-site.xml: zeppelin.interpreter.dep.mvnRepo	*http://repo1.maven.org/maven2/* Удаленный основной репозиторий для загрузки дополнительных зависимостей интерпретатора
• zeppelin-env.sh: ZEPPELIN_DEP_LOCALREPO • zeppelin-site.xml: zeppelin.dep.localrepo	*local-repo* Локальный репозиторий для загрузки зависимостей. Модули npm
• zeppelin-env.sh: ZEPPELIN_HELIUM_NPM_REGISTRY • zeppelin-site.xml: zeppelin.helium.npm.registry	*http://registry.npmjs.org/* Удаленный реестр Npm для загрузки зависимостей Helium
• zeppelin-env.sh: ZEPPELIN_INTERPRETER_OUTPUT _LIMIT • zeppelin-site.xml: zeppelin.interpreter.output.limit	*102400* Размер выходного сообщения от интерпретатора. Если сообщение превышает заданный размер, то она урезается до заданного значения.
• zeppelin-env.sh: ZEPPELIN_WEBSOCKET_MAX_TEXT _MESSAGE_SIZE • zeppelin-site.xml: zeppelin.websocket.max.text.message.size	*1024000* Размер (в символах) максимального текстового сообщения, которое может быть получено от websocket

Параметр Zeppelin	Значение и описание
• zeppelin-env.sh: ZEPPELIN_SERVER_DEFAULT_DIR _ALLOWED • zeppelin-site.xml: zeppelin.server.default.dir.allowed	*false* Доступность каталогов на сервере

3.2 Конфигурация SSL

Включение **SSL** требует некоторых изменений конфигурации – следует создать сертификаты, а затем обновить необходимые настройки для подключения проверки подлинности **SSL** со стороны сервера и/или клиентской стороны.

3.2.1 Создание и настройка сертификатов

Информацию о создании сертификатов и хранилище ключей можно найти по ссылке. Краткий пример можно найти в верхнем ответе на вопрос StackOverflow.

Хранилище ключей **keystore** содержит закрытый ключ и сертификат на сервере, а хранилище **trustore** содержит сертификаты доверенных клиентов. Необходимо убедиться, что путь и пароль для этих двух хранилищ правильно настроены в полях пароля, приведенных ниже. Они могут быть скрыты с помощью **Jetty Password Tool**. **Maven** подтягивает все зависимости для сборки **Zeppelin**, один из jar-файлов **Jetty** содержит инструмент **Jetty Password Tool**. Необходимо вызвать команду из каталога сборки *Zeppelin Home* с соответствующей версией, пользователем и паролем:

```
java -cp ./zeppelin-server/target/lib/jetty-all-server-<version>.jar org.eclipse.jetty.util.
↪security.Password <user> <password>
```

Если используется самоподписанный сертификат, сертификат, подписанный недоверенным центром сертификации, или если включена аутентификация клиента, то у клиента в браузере должно появиться исключение как в случае https-порта, так и websocket-порта. Это можно проверить, установив HTTPS соединение по обоими портами в браузере (например, если порты *443* и *8443*, при переходе на *https://127.0.0.1:443* и *https://127.0.0.1:8443*). Данный шаг может быть пропущен, если сертификат сервера подписан доверенным центром сертификации и аутентификация клиента отключена.

3.2.2 Настройка SSL на стороне сервера

Для включения **SSL** на стороне сервера необходимо отредактировать в *zeppelin-site.xml* следующие свойства:

```
<property>
  <name>zeppelin.server.ssl.port</name>
  <value>8443</value>
  <description>Server ssl port. (used when ssl property is set to true)</description>
</property>

<property>
  <name>zeppelin.ssl</name>
  <value>true</value>
  <description>Should SSL be used by the servers?</description>
</property>

<property>
  <name>zeppelin.ssl.keystore.path</name>
```

```
  <value>keystore</value>
  <description>Path to keystore relative to Zeppelin configuration directory</description>
</property>

<property>
  <name>zeppelin.ssl.keystore.type</name>
  <value>JKS</value>
  <description>The format of the given keystore (e.g. JKS or PKCS12)</description>
</property>

<property>
  <name>zeppelin.ssl.keystore.password</name>
  <value>change me</value>
  <description>Keystore password. Can be obfuscated by the Jetty Password tool</description>
</property>

<property>
  <name>zeppelin.ssl.key.manager.password</name>
  <value>change me</value>
  <description>Key Manager password. Defaults to keystore password. Can be obfuscated.</
↪description>
</property>
```

3.2.3 Включение проверки подлинности сертификата на стороне клиента

Для включения аутентификации по сертификату на стороне клиента необходимо отредактировать в *zeppelin-site.xml* следующие свойства:

```
<property>
  <name>zeppelin.server.ssl.port</name>
  <value>8443</value>
  <description>Server ssl port. (used when ssl property is set to true)</description>
</property>

<property>
  <name>zeppelin.ssl.client.auth</name>
  <value>true</value>
  <description>Should client authentication be used for SSL connections?</description>
</property>

<property>
  <name>zeppelin.ssl.truststore.path</name>
  <value>truststore</value>
  <description>Path to truststore relative to Zeppelin configuration directory. Defaults to the␣
↪keystore path</description>
</property>

<property>
  <name>zeppelin.ssl.truststore.type</name>
  <value>JKS</value>
  <description>The format of the given truststore (e.g. JKS or PKCS12). Defaults to the same type␣
↪as the keystore type</description>
</property>

<property>
  <name>zeppelin.ssl.truststore.password</name>
  <value>change me</value>
```

```
  <description>Truststore password. Can be obfuscated by the Jetty Password tool. Defaults to the␣
↪keystore password</description>
</property>
```

3.2.4 Скрытие паролей с помощью Jetty Password Tool

Лучшие практики по безопасности рекомендуют не использовать текстовые пароли, а с помощью утилиты **Jetty Password Tool** (см. документацию) можно усложнить пароли, используемые для доступа к **keystore** и **trustore**.

После установки **Jetty Password Tool**:

```
java -cp $ZEPPELIN_HOME/zeppelin-server/target/lib/jetty-util-9.2.15.v20160210.jar \
         org.eclipse.jetty.util.security.Password  \
         password

2016-12-15 10:46:47.931:INFO::main: Logging initialized @101ms
password
OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v
MD5:5f4dcc3b5aa765d61d8327deb882cf99
```

Затем необходимо обновить конфигурацию с паролем:

```
<property>
  <name>zeppelin.ssl.keystore.password</name>
  <value>OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v</value>
  <description>Keystore password. Can be obfuscated by the Jetty Password tool</description>
</property>
```

Important: После обновления настроек сервер Zeppelin необходимо перезапустить

Глава 4

Аутентификация Apache Shiro для Apache Zeppelin

4.1 Обзор

Apache Shiro – это мощный и простой в использовании **Java** фреймворк безопасности, плозволяющий выполнить аутентификацию, авторизацию, криптографию и управление сессиями. В данном разделе объясняется, как **Shiro** работает для аутентификации на сервер **Zeppelin**.

При подключении к **Apache Zeppelin** необходимо ввести учетные данные. После входа в систему появляется доступ ко всем блокнотам, включая блокноты других пользователей.

4.2 Настройка безопасности

Настройка аутентификации на сервер **Zeppelin** осуществляется несколькими простыми шагами:

1. Права Shiro

По умолчанию в *conf* находится файл *shiro.ini.template*. Данный файл используется в качестве примера, и настоятельно рекомендуется создать файл *shiro.ini*, выполнив следующую команду:

```
cp conf/shiro.ini.template conf/shiro.ini
```

Дополнительная информация о формате файла *shiro.ini* находится по ссылке Shiro Configuration.

2. Безопасность канала Websocket

Для свойства *zeppelin.anonymous.allowed* необходимо установить значение *false* в *conf/zeppelin-site.xml*. В случае если данного файла нет, следует просто скопировать файл *conf/zeppelin-site.xml.template* в *conf/zeppelin-site.xml*.

3. Запуск Zeppelin

Для запуска **Zeppelin** необходимо выполнить команду:

```
bin/zeppelin-daemon.sh start (or restart)
```

После чего можно обратиться к **Zeppelin** по адресу *http://localhost:8080*.

4. Авторизация

Теперь можно войти в систему, используя комбинацию имени и пароля пользователя (Рис.4.1.).

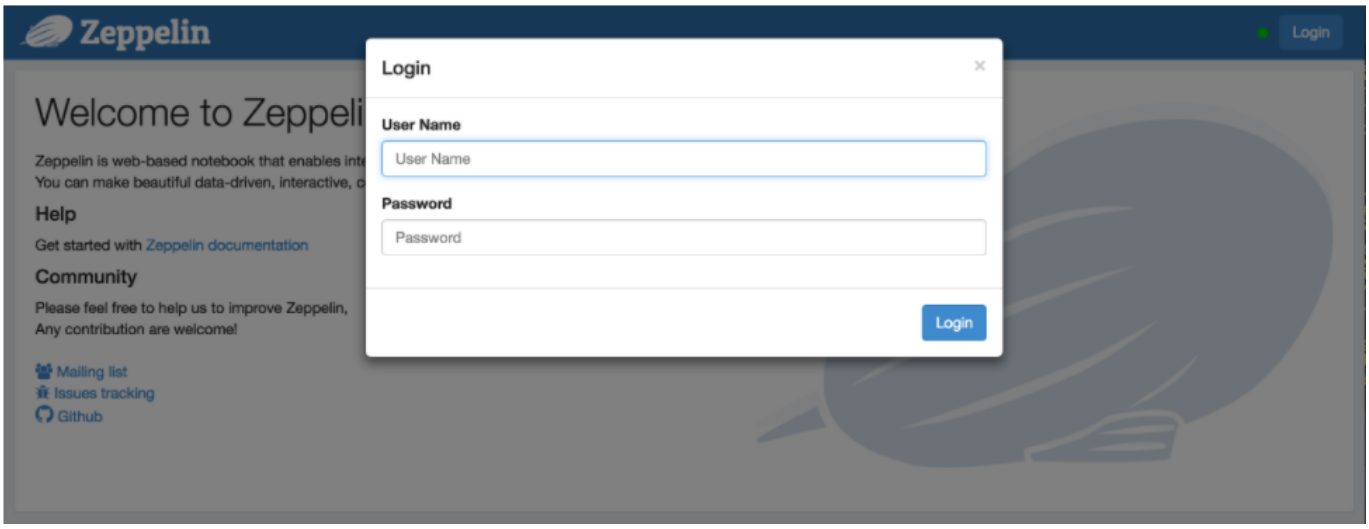

Рис.4.1.: Авторизация в Apache Zeppelin

После пароля через запятую можно указать роли для каждого пользователя:

```
[users]

admin = password1, admin
user1 = password2, role1, role2
user2 = password3, role3
user3 = password4, role2
```

4.3 Группы и разрешения (опционально)

Для использования групп пользователей и прав для них необходимо применить одну из нижеприведенных конфигураций для **LDAP** или **AD** в разделе *[main]* в файле *shiro.ini*:

```
activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.systemUsername = userNameA
activeDirectoryRealm.systemPassword = passwordA
activeDirectoryRealm.searchBase = CN=Users,DC=SOME_GROUP,DC=COMPANY,DC=COM
activeDirectoryRealm.url = ldap://ldap.test.com:389
```

```
activeDirectoryRealm.groupRolesMap = "CN=aGroupName,OU=groups,DC=SOME_GROUP,DC=COMPANY,DC=COM":
↪"group1"
activeDirectoryRealm.authorizationCachingEnabled = false
activeDirectoryRealm.principalSuffix = @corp.company.net

ldapRealm = org.apache.zeppelin.server.LdapGroupRealm
# search base for ldap groups (only relevant for LdapGroupRealm):
ldapRealm.contextFactory.environment[ldap.searchBase] = dc=COMPANY,dc=COM
ldapRealm.contextFactory.url = ldap://ldap.test.com:389
ldapRealm.userDnTemplate = uid={0},ou=Users,dc=COMPANY,dc=COM
ldapRealm.contextFactory.authenticationMechanism = simple
```

И определить роли/группы, которые необходимо иметь в системе, например:

```
[roles]
admin = *
hr = *
finance = *
group1 = *
```

4.4 Настройка Realm (опционально)

Realms отвечают за аутентификацию и авторизацию в **Apache Zeppelin**. По умолчанию **Apache Zeppelin** использует IniRealm (пользователи и группы настраиваются в файле *conf/shiro.ini* в разделах *[user]* и *[group]*). Также можно использовать **Shiro Realms**, такие как JndiLdapRealm, JdbcRealm или создать собственный. Подробная документация о **Apache Shiro Realm** представлена по ссылке.

4.4.1 Active Directory

```
activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.systemUsername = userNameA
activeDirectoryRealm.systemPassword = passwordA
activeDirectoryRealm.hadoopSecurityCredentialPath = jceks://file/user/zeppelin/conf/zeppelin.jceks
activeDirectoryRealm.searchBase = CN=Users,DC=SOME_GROUP,DC=COMPANY,DC=COM
activeDirectoryRealm.url = ldap://ldap.test.com:389
activeDirectoryRealm.groupRolesMap = "CN=aGroupName,OU=groups,DC=SOME_GROUP,DC=COMPANY,DC=COM":
↪"group1"
activeDirectoryRealm.authorizationCachingEnabled = false
activeDirectoryRealm.principalSuffix = @corp.company.net
```

Кроме того, вместо указания *systemPassword* в виде текста в *shiro.ini* администратор может указать то же самое, что и в *hadoop credential*. Необходимо создать keystore-файл, используя командную строку *hadoop credential*, для этого *hadoop* должен быть прописан в *classpath*:

```
hadoop credential create activeDirectoryRealm.systempassword -provider jceks://file/user/zeppelin/
↪conf/zeppelin.jceks
```

Далее следует изменить следующие значения в файле *Shiro.ini* и раскомментировать строку:

```
activeDirectoryRealm.hadoopSecurityCredentialPath = jceks://file/user/zeppelin/conf/zeppelin.jceks
```

4.4.2 LDAP

Для настройки **LDAP Realm** существует два способа. Проще использовать **LdapGroupRealm**. Однако, он менее гибкий при настройке соответствий между группами **LDAP** и пользователями, а также для

авторизации групп пользователей. Далее приведен пример файла с соответствующими настройками:

```
ldapRealm = org.apache.zeppelin.realm.LdapGroupRealm
# search base for ldap groups (only relevant for LdapGroupRealm):
ldapRealm.contextFactory.environment[ldap.searchBase] = dc=COMPANY,dc=COM
ldapRealm.contextFactory.url = ldap://ldap.test.com:389
ldapRealm.userDnTemplate = uid={0},ou=Users,dc=COMPANY,dc=COM
ldapRealm.contextFactory.authenticationMechanism = simple
```

Другим более гибким способом является использование **LdapRealm**. Он позволяет сопоставлять *ldapgroups* с ролями, а также допускает проверку подлинности на основе ролей/групп на сервере *zeppelin*. Пример конфигурации приведен ниже:

```
ldapRealm=org.apache.zeppelin.realm.LdapRealm

ldapRealm.contextFactory.authenticationMechanism=simple ldapRealm.contextFactory.url=ldap://
↪localhost:33389  ldapRealm.userDnTemplate=uid={0},ou=people,dc=hadoop,dc=apache,dc=org
```

Возможность задать параметр ldap paging. Размер по умолчанию - 100

```
ldapRealm.pagingSize = 200 ldapRealm.authorizationEnabled=true ldapRealm.contextFactory.
↪systemAuthenticationMechanism=simple ldapRealm.searchBase=dc=hadoop,dc=apache,dc=org ldapRealm.
↪userSearchBase = dc=hadoop,dc=apache,dc=org ldapRealm.groupSearchBase = ou=groups,dc=hadoop,
↪dc=apache,dc=org ldapRealm.groupObjectClass=groupofnames
```

Возможность настройки параметра userSearchAttribute

```
ldapRealm.userSearchAttributeName = sAMAccountName ldapRealm.memberAttribute=member
```

Возврат имен пользователей из ldap в ниженем регистре для использования в AD

```
ldapRealm.userLowerCase = true
```

Возможность установить парметр searchScopes в одно из трех значений: subtree (по умолчанию), one или base

```
ldapRealm.userSearchScope = subtree; ldapRealm.groupSearchScope = subtree; ldapRealm.
↪memberAttributeValueTemplate=cn={0},ou=people,dc=hadoop,dc=apache,dc=org ldapRealm.
↪contextFactory.systemUsername=uid=guest,ou=people,dc=hadoop,dc=apache,dc=org ldapRealm.
↪contextFactory.systemPassword=S{ALIAS=ldcSystemPassword}
```

Включение поддержки вложенных групп при помощи оператора LDAPMATCHINGRULEINCHAIN

```
ldapRealm.groupSearchEnableMatchingRuleInChain = true
```

Дополительная настройка соответствий между физическими группами и логическими ролями приложений

```
ldapRealm.rolesByGroup = LDNUSERS: userrole, NYKUSERS: userrole, HKGUSERS: userrole, GLOBALADMIN:␣
↪adminrole
```

Дополнительный список ролей, которым разрешена аутентификация. В случае если список не представлен, всем ролям разрешается аутентификация (вход)

Данные изменения не влияют на специфические права url. Для url будут работать те права, котрые указаны в разделе [urls]

```
ldapRealm.allowedRolesForAuthentication = adminrole,userrole ldapRealm.permissionsByRole=␣
↪userrole = :ToDoItemsJdo::, *:ToDoItem::*; adminrole = * securityManager.sessionManager =
↪$sessionManager securityManager.realms = $ldapRealm ```
```

4.4.3 PAM

Поддержка аутентификации с помощью PAM позволяет повторно использовать существующие модули аутентификации в узле, где запущен **Zeppelin**. В типичных системных модулях, например, *sshd*, *passwd* и других сервис настраивается в */etc/pam.d/*. Можно повторно использовать один из этих сервисов или создать свой собственный для **Zeppelin**. Для активации аутентификации **PAM** требуется два параметра: 1 – realm: использование **Shiro realm**; 2 – service: настроенный в */etc/pam.d/* сервис. Название должно совпадать с именем файла в */etc/pam.d/*.

```
[main]
 pamRealm=org.apache.zeppelin.realm.PamRealm
 pamRealm.service=sshd
```

4.4.4 ZeppelinHub

ZeppelinHub – это сервис, синхронизурующий блокноты **Apache Zeppelin** и обеспечивающий легкое взаимодействие с ними. Для подключения **ZeppelinHub** необходимо применить следующее изменение в *conf/shiro.ini* в разделе *[main]*:

```
### A sample for configuring ZeppelinHub Realm
zeppelinHubRealm = org.apache.zeppelin.realm.ZeppelinHubRealm
## Url of ZeppelinHub
zeppelinHubRealm.zeppelinhubUrl = https://www.zeppelinhub.com
securityManager.realms = $zeppelinHubRealm
```

Important: ZeppelinHub не относится к проекту Apache Zeppelin

4.5 Безопасность Cookie в сессиях Zeppelin (опционально)

Zeppelin может быть настроен выставлением флага **HttpOnly** в настройка **cookie** для сессии. С такой конфигурацией cookie-файлы **Zeppelin** не могут быть доступны через скрипты на стороне клиента, тем самым предотвращая большинство атак типа **Cross-Site scripting** (**XSS**).

Чтобы включить безопасную поддержку файлов **cookie** через **Shiro**, необходимо добавить следующие строки в *conf/shiro.ini* в раздел *[main]*, а затем задать *sessionManager*:

```
cookie = org.apache.shiro.web.servlet.SimpleCookie
cookie.name = JSESSIONID
cookie.secure = true
cookie.httpOnly = true
sessionManager.sessionIdCookie = $cookie
```

4.6 Защита информации Zeppelin (опционально)

По умолчанию любой пользователь, определенный в *[users]*, может видеть информацию об интерпретаторах, учетных данных и настройках в **Apache Zeppelin**. В случае если данную информацию необходимо скрыть, поскольку **Shiro** обеспечивает защиту на уровне url, следует закомментировать или раскомментировать приведенные ниже строки в *conf/shiro.ini*:

```
[urls]

/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
```

В таком случае информацию об интерпретаторах, учетных данных и настройках в **Apache Zeppelin** могут видеть только пользователи с ролью *admin*. При необходимости предоставления прав другим пользователям следует изменить роли в разделе *[users]*.

4.7 Альтернативные методы аутентификации

HTTP аутентификация с помощью NGINX

Глава 5

Интерфейс Apache Zeppelin

- *Главная страница*
- *Меню*
- *Набор инструментов*

5.1 Главная страница

При первом обращении к серверу **Zeppelin** открывается главная страница, представленная на Рис.5.1.

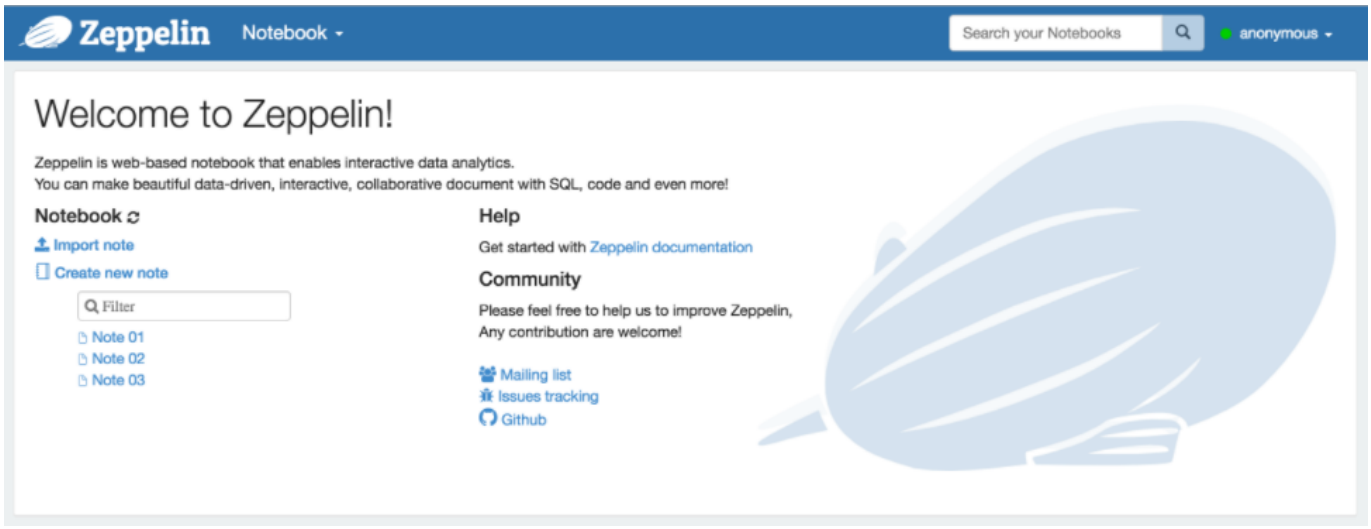

Рис.5.1.: Главная страница Apache Zeppelin

В левой части страницы перечислены все существующие блокноты. По умолчанию блокноты сохраняются в папке *$ZEPPELIN_HOME/notebook*.

Есть возможность фильтровать блокноты по имени, используя форму ввода текста. Также можно создавать новые, обновлять список существующих блокнотов (в случае если они вручную скопированы в папку *$ZEPPELIN_HOME/notebook*) и импортировать их.

При нажатии кнопки “Import Note” открывается новое диалоговое окно (Рис.5.2.). Блокнот можно импортировать с локального диска или из удаленного места по указанному URL-адресу. По умолчанию название импортируемого блокнота остается оригинальным, но при желании его можно изменить.

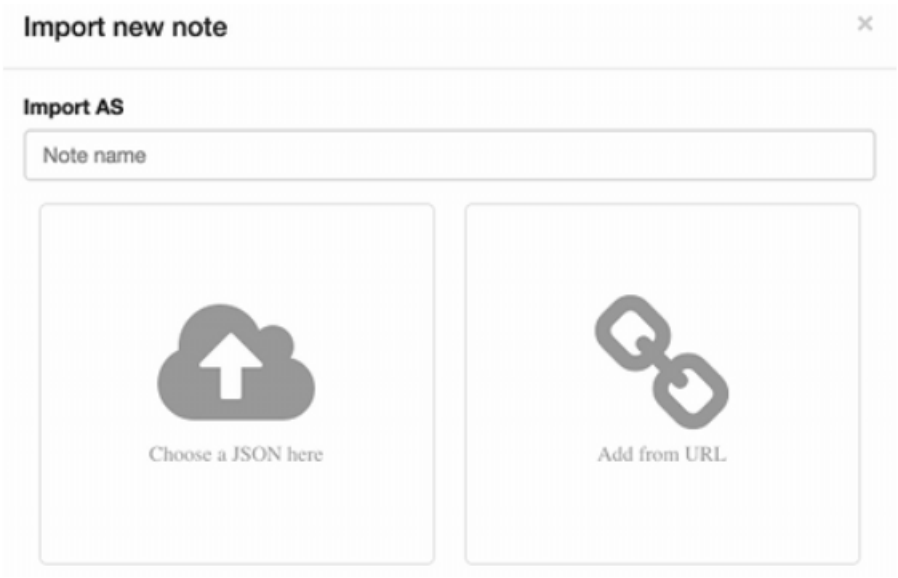

Рис.5.2.: Импорт новой заметки

5.2 Меню

Панель меню “Notebook” предлагает почти те же функции, что и раздел управления блокнотами на главной странице (Рис.5.1.). В раскрывающемся контекстном меню можно выбрать (Рис.5.3.):

- Открыть выбранный блокнот;
- Фильтр блокнотов по имени;
- Создать новый блокнот.

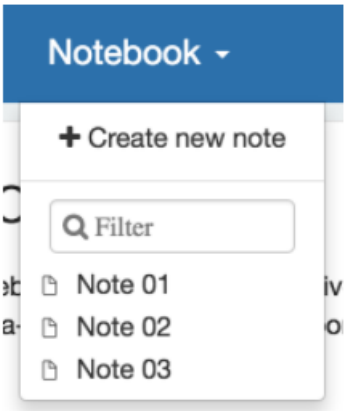

Рис.5.3.: Панель меню Notebook

Меню настроек дает доступ к конфигурации и отображает информацию о **Zeppelin** (Рис.5.4.). При использовании настроек по умолчанию имя пользователя задается как *anonymous*. Настройка аутентификации описана в разделе Аутентификация Apache Shiro для Apache Zeppelin.

По ссылке “About Zeppelin” (см. Рис.5.4.) можно получить информацию об установленной версии **Apache Zeppelin** (Рис.5.5.).

Перейдя по ссылке “Interpreter” (см. Рис.5.4.), можно выполнить следующие функции (Рис.5.6.):

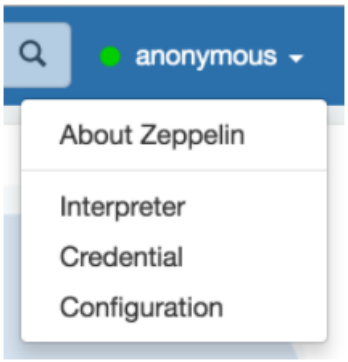

Рис.5.4.: Меню настроек

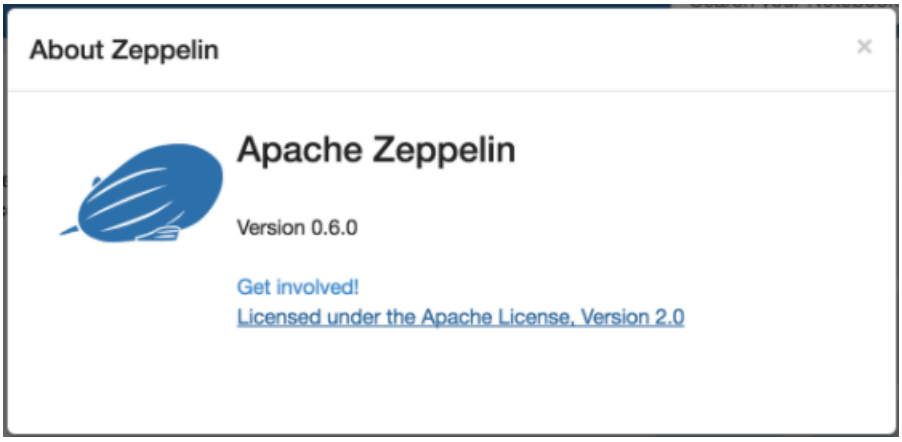

Рис.5.5.: About Zeppelin

- Настроить существующий интерпретатор;
- Добавить/удалить интерпретатор.

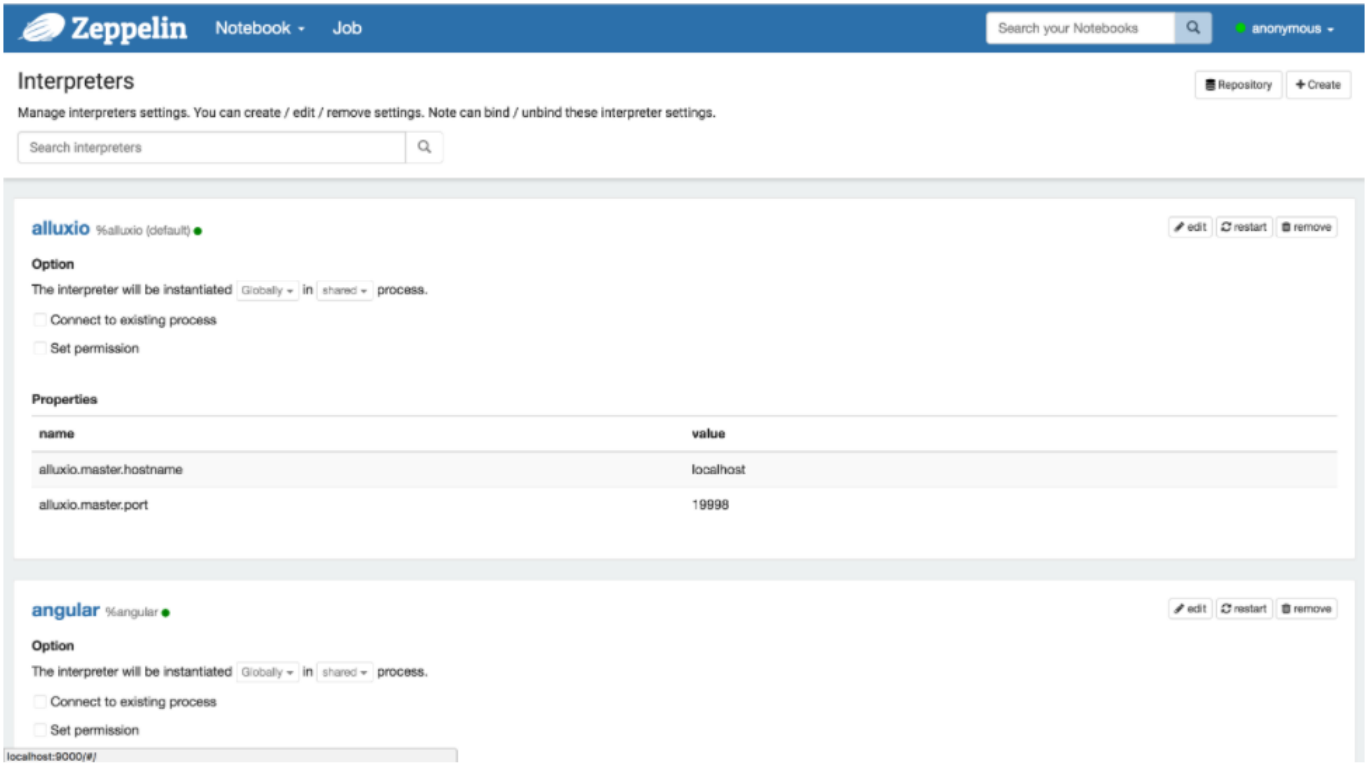

Рис.5.6.: Управление интерпретаторами

Ссылка “Credential” (см. Рис.5.4.) позволяет для источников данных сохранять учетные данные, которые передаются интерпретаторам (Рис.5.7.).

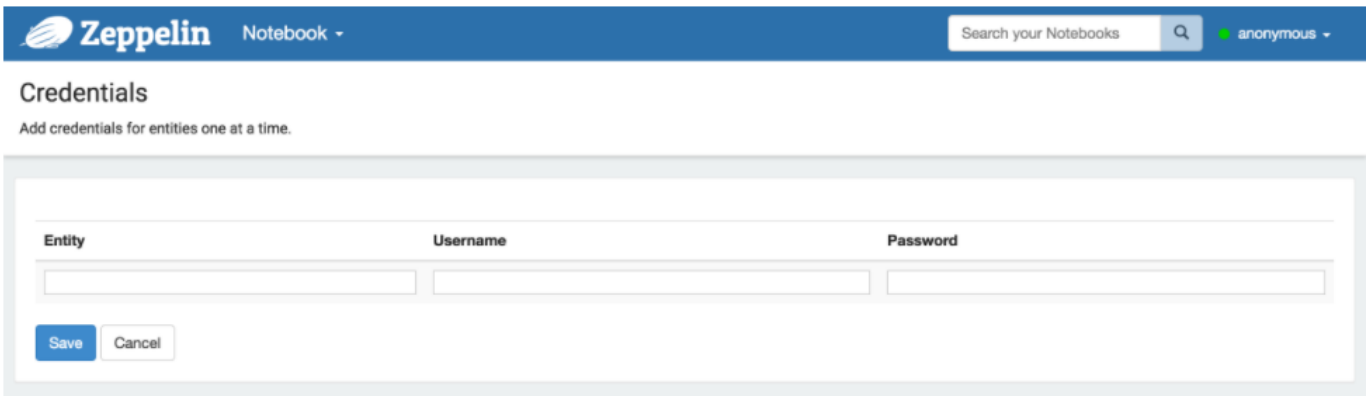

Рис.5.7.: Учетные данные

Ссылка “Configuration” (см. Рис.5.4.) отображает все настройки **Apache Zeppelin**, которые заданы в файле конфигурации *$ZEPPELIN_HOME/conf/zeppelin-site.xml* (Рис.5.8.).

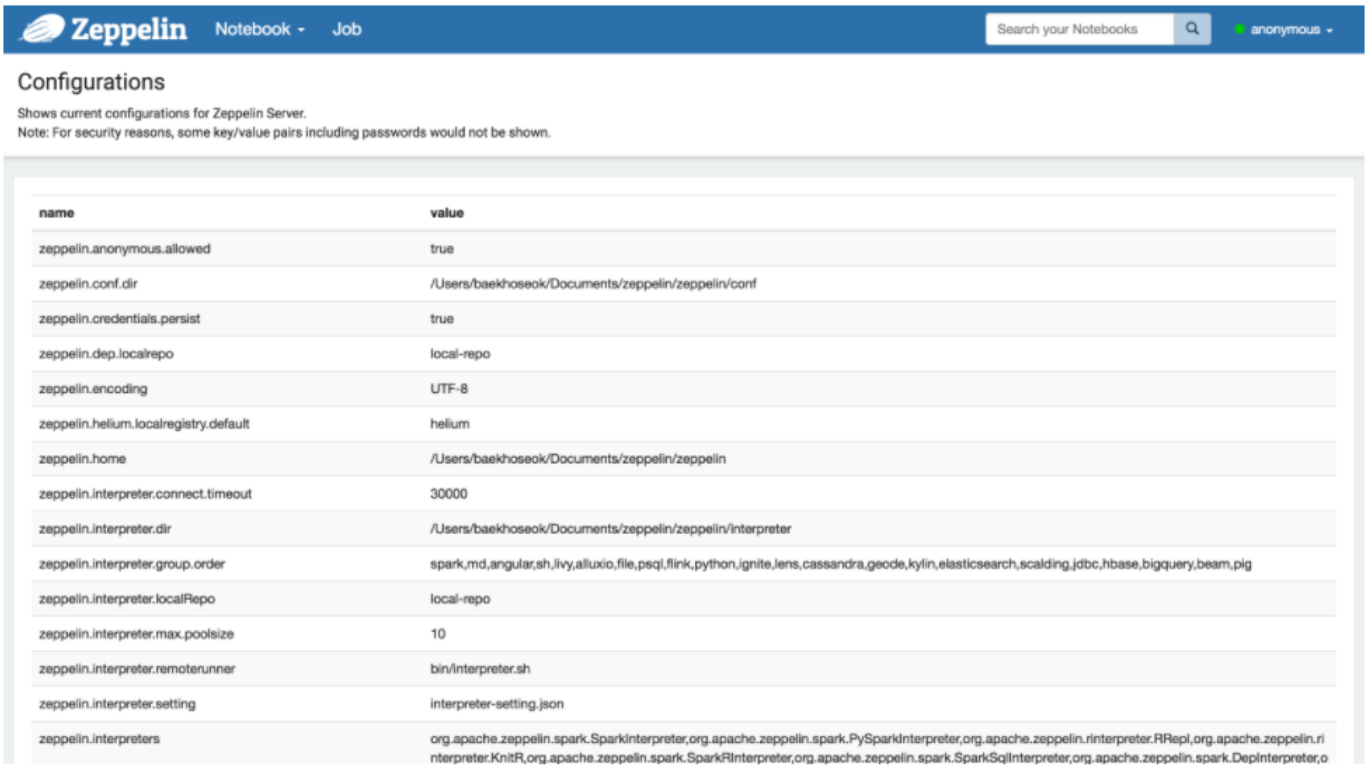

Рис.5.8.: Конфигурация Apache Zeppelin

5.3 Набор инструментов

Каждый блокнот **Apache Zeppelin** состоит из нескольких параграфов (Рис.5.9.). Блокнот можно рассматривать как контейнер параграфов.

Каждый параграф состоит из двух разделов: *code section*, в который помещается исходный код, и *result section*, где можно увидеть результат выполнения кода (Рис.5.10.).

В правом верхнем углу каждого параграфа есть несколько команд:

- Выполнить код параграфа;
- Скрыть/показать *code section*;
- Скрыть/показать *result section*;
- Настроить параграф.

Для перехода к настройкам параграфа необходимо нажать на значок шестеренки, при этом открывается контекстное меню (Рис.5.11.).

В диалоговом окне отображается следующая информация и возможные действия:

- Идентификатор параграфа (в данном примере *20150924-163507_134879501*);
- Ширина параграфа. Поскольку Zeppelin использует сетчатую систему Twitter Bootstrap, ширина каждого параграфа может быть изменена от *1* до *12*;
- Переместить параграф на 1 уровень вверх;
- Переместить параграф на 1 уровень вниз;
- Создать новый параграф;

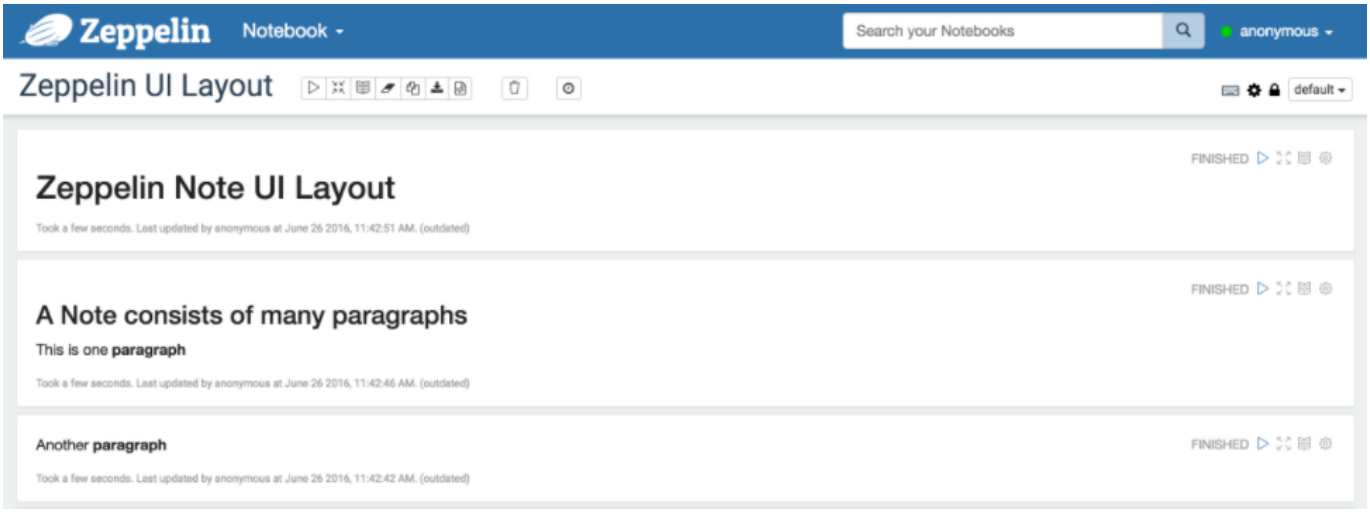

Рис.5.9.: Шаблон блокнота

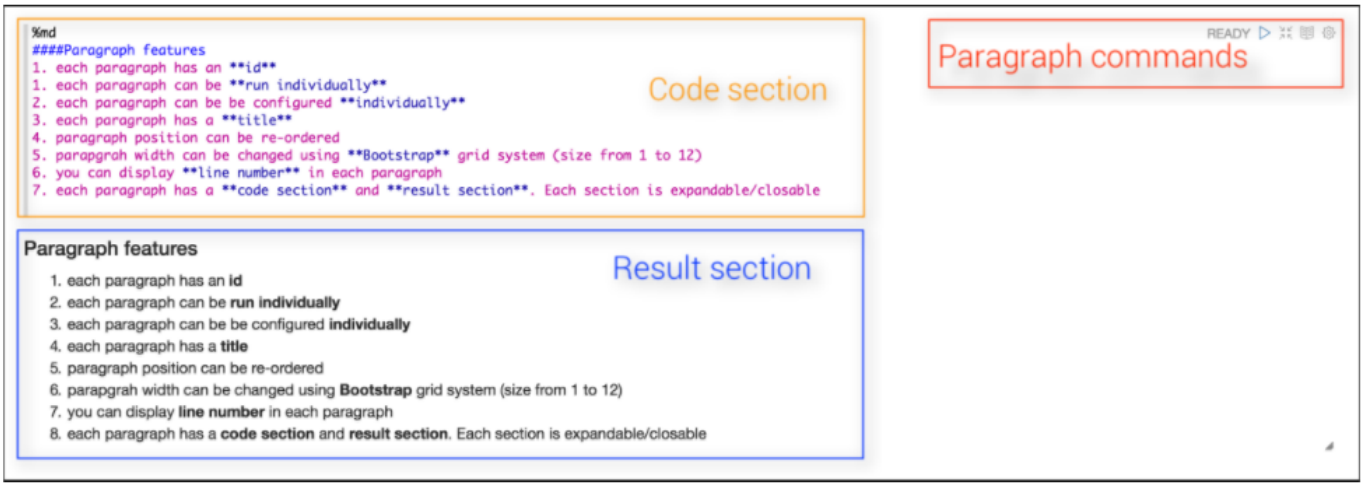

Рис.5.10.: Разделы параграфа

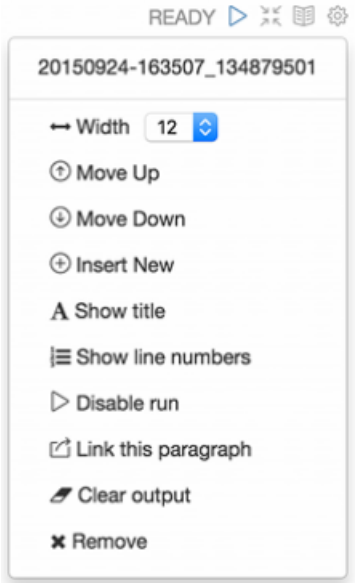

Рис.5.11.: Контекстное меню настроек параграфа

- Изменить название параграфа;
- Показать/скрыть номера строк в *code section*;
- Отключить кнопку запуска для параграфа;
- Экспортировать текущий параграф как *iframe* и открыть *iframe* в новом окне;
- Очистить *result section*;
- Удалить текущий параграф.

5.3.1 Панель инструментов блокнота

В верхней части экранной формы блокнота **Apache Zeppelin** находится панель инструментов, которая представляет собой командные кнопки настройки, безопасности и отображения (см. Рис.5.9.).

В левом углу панели инструментов отображается название блокнота, необходимо нажать на него, чтобы открыть форму ввода и обновить его. По центру панели находятся следующие кнопки:

- Выполнить все параграфы последовательно в порядке их отображения;
- Скрыть/показать *code section* всех параграфов;
- Скрыть/показать *result section* всех параграфов;
- Очистить *result section* всех параграфов;
- Копировать текущий блокнот;
- Экспортировать текущий блокнот в файл *JSON*. При этом *code section* и *result section* всех параграфов будут экспортированы. При наличии тяжелых данных в *result section*, рекомендуется их очистить перед экспортом;
- Зафиксировать текущее содержимое блокнота;
- Удалить блокнот;
- Запланировать выполнение всех параграфов, используя синтаксис *CRON*.

Справа от панели инструментов блокнота располагаются кнопки конфигурации:

- Отобразить все сочетания клавиш клавиатуры;
- Настроить интерпретаторы, привязанные к текущему блокноту;
- Настроить права для блокнота;
- Переключить режим отображения блокнота на *default*, *simple* или *report*.

Глава 6

Интерпретаторы в Apache Zeppelin

6.1 Обзор

В разделе рассказывается об интерпретаторах, их группах и настройках в **Apache Zeppelin**. Концепция интерпретатора **Zeppelin** позволяет подключать любой язык/фреймворк обработки данных. В настоящее время **Zeppelin** поддерживает множество интерпретаторов, таких как **Scala** (с **Apache Spark**), **Python** (с **Apache Spark**), **Spark SQL**, **JDBC**, **Markdown**, **Shell** и другие.

6.2 Интерпретатор Zeppelin

Интерпретатор **Zeppelin** – это плагин, который позволяет пользователям **Apache Zeppelin** использовать определенный язык/фреймворк обработки данных. Например, чтобы использовать Scala-код в **Zeppelin**, понадобится *%spark* интерпретатор.

На странице интерпретатора при нажатии кнопки “+Create” в открывшемся диалоговом окне в выпадающем списке поля “Interpreter” отображаются все доступные интерпретаторы на сервере (Рис.6.1.).

6.3 Настройка интерпретатора

Настройка интерпретатора **Zeppelin** – это конфигурация данного интерпретатора на сервере **Zeppelin**. Например, ниже приведены свойства, необходимые для подключения интерпретатора **hive JDBC** к **Hive** серверу (Рис.6.2.).

Для экспорта свойств в качестве переменной среды окружения необходимо, чтобы имя свойства состояло из символов верхнего регистра, цифр и подчеркивания (*[A-Z_ 0-9]*). В противном случае свойства задаются как свойство **JVM**.

Каждый блокнот может быть связан с несколькими конфигурациями одного интерпретатора, для этого следует использовать значок настройки в правом верхнем углу блокнота (Рис.6.3.).

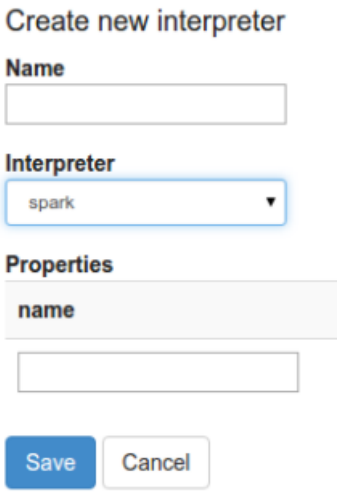

Рис.6.1.: Создать новый интерпретатор

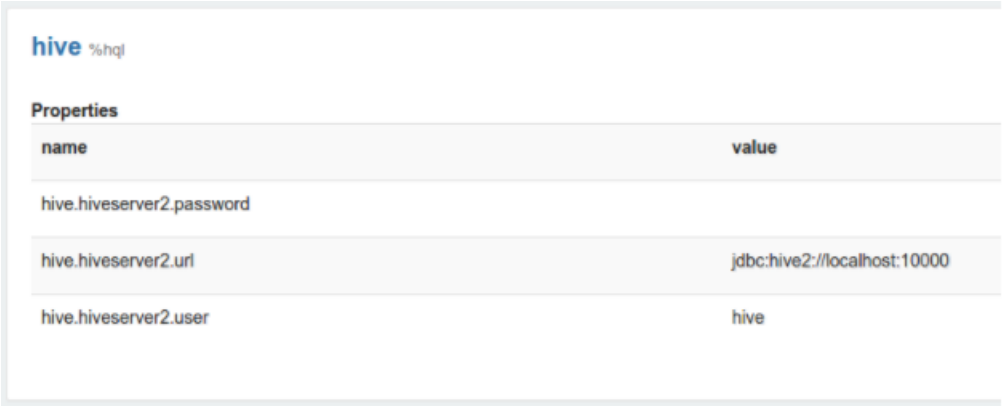

Рис.6.2.: Свойства интерпретатора

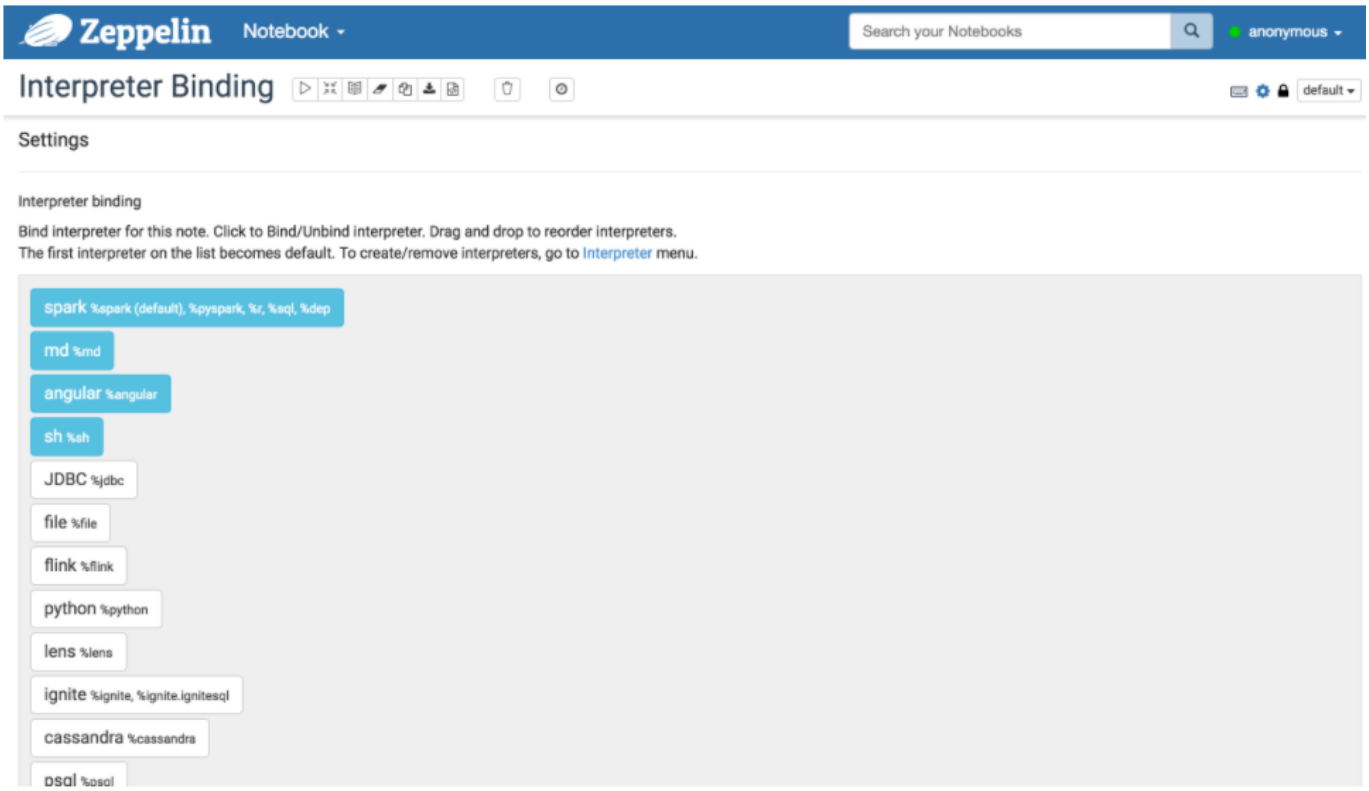

Рис.6.3.: Настройки интерпретатора

6.4 Группа интерпретаторов

Каждый интерпретатор принадлежит к группе интерпретаторов. **Группа интерпретаторов (Interpreter Group)** – это единый блок, позволяющий одновременно управлять (start/stop) несколькими интерпретаторами. По умолчанию каждый интерпретатор принадлежит к одной группе, но группа может содержать больше интерпретаторов. Например, группа интерпретаторов **Spark** включает поддержку **Spark**, **pySpark**, **Spark SQL** и загрузчик зависимостей *%dep*.

Технически, интерпретаторы **Zeppelin** одной группы запускаются в одной **JVM**. Каждый из интерпретаторов может относиться к одной группе. Все их свойства перечислены в настройках интерпретатора (Рис.6.4.).

6.5 Режим привязки интерпретатора

Каждая конфигурация интерпретатора может быть сделана в одном из приведенных режимов привязки: *"shared"* – общедоступный, *"scoped"* – ограниченный, *"isolated"* – отдельный (Рис.6.5.). В режиме *"shared"* каждый блокнот, связанный с конфигурацией интерпретатора, совместно использует один экземпляр интерпретатора. В режиме *"scoped"* каждый блокнот создает новый экземпляр интерпретатора в том же процессе интерпретатора. В режиме *"isolated"* каждый блокнот создает новый процесс интерпретатора.

6.6 Подключение к существующему удаленному интерпретатору

Существует возможность запуска потока интерпретатора пользователем **Zeppelin** на удаленном узле. Для этого необходимо создать экземпляр *RemoteInterpreterServer* и запустить его следующим образом:

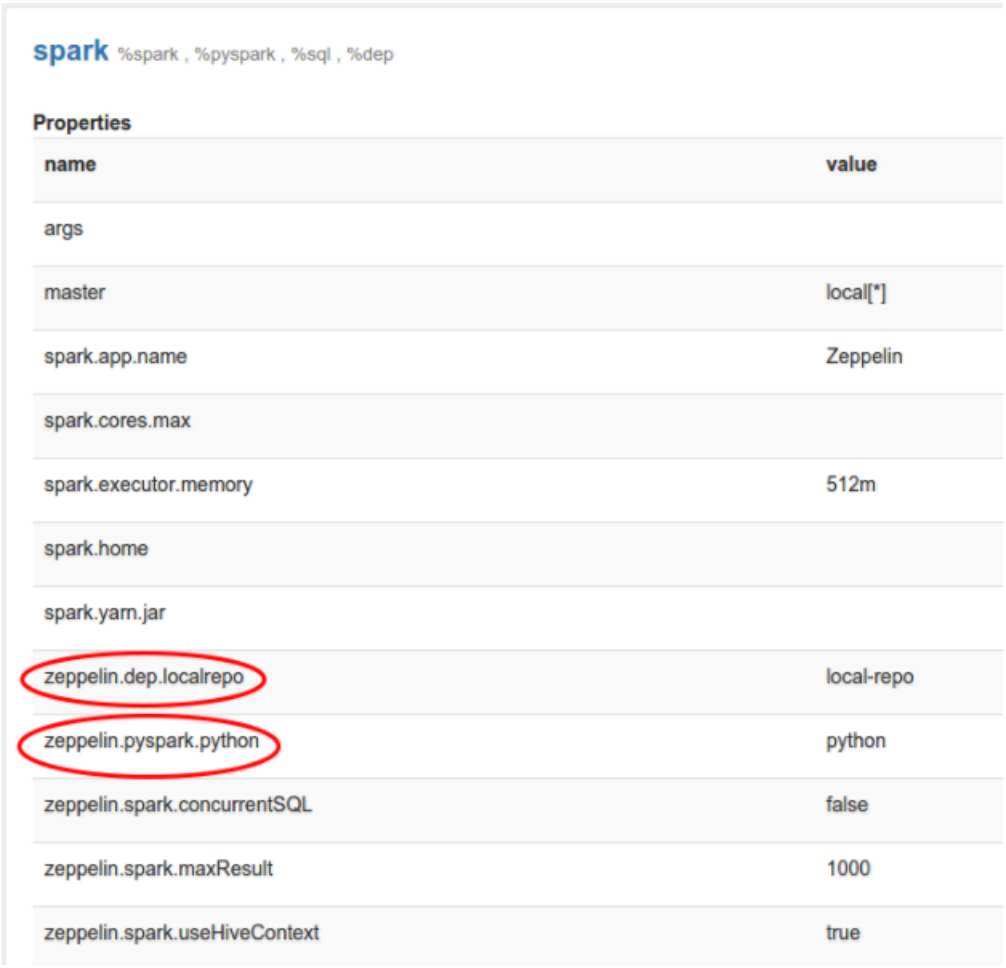

Рис.6.4.: Свойства групп интерпретатора

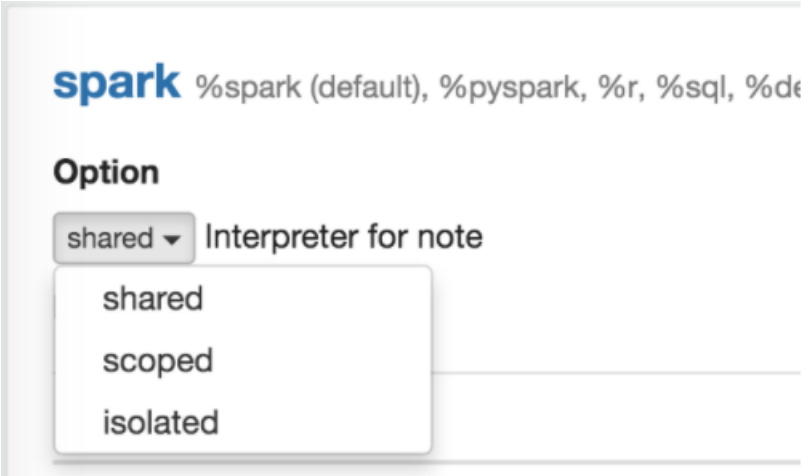

Рис.6.5.: Режимы привязки интерпретатора

```
RemoteInterpreterServer interpreter=new RemoteInterpreterServer(3678);
// Here, 3678 is the port on which interpreter will listen.
interpreter.start()
```

Данный код запускает поток интерпретатора внутри процесса. После запуска интерпретатора можно настроить **Zeppelin** для подключения к *RemoteInterpreter*, установив флаг *"Connect to existing process"* и указав узел (*Host*) и порт (*Port*), который слушает процесс интерпретатора (Рис.6.6.).

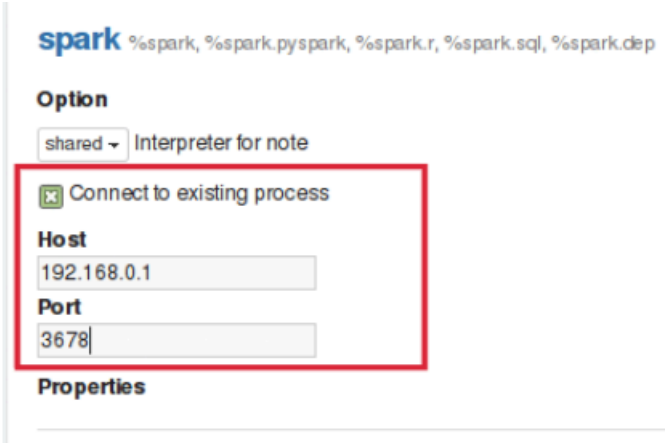

Рис.6.6.: Подключение к удаленному интерпретатору

Глава 7

Python 2 & 3 Интерпретатор для Apache Zeppelin

7.1 Конфигурирование

Для настройки **Python** для **Apache Zeppelin** необходимо задать параметры, представленные в таблице.

Таблица7.1.: Параметры конфигурации

Параметр	Значение по умолчанию	Описание
zeppelin.python	python	Путь к уже установленному бинарному пакету Python (может быть *python2* или *python3*). Если *python* не прописан в *$PATH*, можно задать полный путь (например: */usr/bin/python*)
zeppelin.python.maxResult	1000	Максимальное количество отображаемых строк датафрейма

7.2 Включение интерпретатора Python

Для включения интерпретатора **Python** в блокноте необходимо нажать значок “Gear” и выбрать “Python”.

7.3 Использование интепретатора Python

Для выбора интерпретатора **Python** необходимо в параграфе указать *%python*, а затем ввести все команды. Интерпретатор может работать, только если уже установлен *python* (интерпретатор не предоставляет собственные бинарные файлы *python*). Получить справку можно вызовом команды *help()*.

7.4 Переменные окружения Python

- По умолчанию

По умолчанию *PythonInterpreter* использует команду *python*, определенную в свойстве *zeppelin.python* для запуска процесса *python*. Интерпретатор может использовать все установленные модули (с помощью *pip*, *easy_install* и других)

- Conda

Conda – это система управления пакетами и переменными окружения *python*. Интерпретатор *%python.conda* позволяет переключаться между переменными окружения.

Перечень переменных окружения:

```
%python.conda
```

Активация переменных окружения:

```
%python.conda activate [ENVIRONMENT_NAME]
```

Деактивация переменных окружения:

```
%python.conda deactivate
```

- Докер

Интерпретатор *%python.docker* позволяет *PythonInterpreter* создавать процесс *python* в указанном докер-контейнере.

Активация переменных окружения:

```
%python.docker activate [Repository]
%python.docker activate [Repository:Tag]
%python.docker activate [Image Id]
```

Деактивация переменных окружения:

```
%python.docker deactivate
```

Пример:

```
# activate latest tensorflow image as a python environment
%python.docker activate gcr.io/tensorflow/tensorflow:latest
```

7.5 Использование Zeppelin Dynamic Forms

Динамическую форму **Zeppelin** можно использовать внутри кода **Python** (см. подробное описание).

Zeppelin Dynamic Form может использоваться только в том случае, если в системе установлена *py4j Python library*. Библиотеку можно установить с помощью *pip install py4j*.

Пример:

```
%python
### Input form
print (z.input("f1","defaultValue"))

### Select form
print (z.select("f1",[("o1","1"),("o2","2")],"2"))

### Checkbox form
print("".join(z.checkbox("f3", [("o1","1"), ("o2","2")],["1"])))
```

7.6 Интеграция Matplotlib

Интерпретатор **Python** может автоматически отображать графики *matplotlib* с помощью встроенного модуля *pyplot*:

```
%python
import matplotlib.pyplot as plt
plt.plot([1, 2, 3])
```

Это рекомендуемый метод использования *matplotlib* из блокнота **Zeppelin**. Выходные данные команды по умолчанию преобразовываются в HTML, неявно используя *%html*. Дополнительные настройки можно выполнить с помощью встроенного метода *z.configure_mpl()*. Например:

```
z.configure_mpl(width=400, height=300, fmt='svg')
plt.plot([1, 2, 3])
```

В данном примере изображение задается в формате *SVG 400x300*, которое по умолчанию обычно представляется в формате *600x400* и *PNG* соответственно. В дальнейшем можно будет использовать другую библиотеку, *angular*, которая позволит обновлять график, созданный одним параграфом, непосредственно из другого (выходные данные в таком случае *%angular* вместо *%html*). Функция уже доступна в интерпретаторе **pyspark**.

Если **Zeppelin** не может найти файлы *matplotlib* (которые обычно находятся в *$ZEPPELIN_HOME/interpreter/lib/python*) в *PYTHONPATH*, то программа автоматически устанавливается в *agg* и нижеприведенные инструкции могут использоваться с ограничениями.

Если не удается загрузить встроенные модули, можно использовать *z.show(plt)*:

```
python %python import matplotlib.pyplot as plt plt.figure() (.. ..) z.show(plt)
plt.close()
```

Данная функция *z.show()* может принимать дополнительные параметры для адаптации размеров графика (ширина и высота), а также его выходной формат – *png* или опционально *svg* (Рис.7.1.):

```
%python
z.show(plt, width='50px')
z.show(plt, height='150px', fmt='svg')
```

7.7 Интеграция с Pandas

Система отображения таблиц **Apache Zeppelin** предоставляет встроенные возможности визуализации данных. Интерпретатор **Python** использует его для визуализации датафреймов *Pandas*, аналогично через API функции *z.show()* как в случае интеграции с библиотекой *matplotlib* (*Интеграция Matplotlib*). Например:

```
import pandas as pd
rates = pd.read_csv("bank.csv", sep=";")
z.show(rates)
```

7.8 SQL поверх датафреймов Pandas

Существует удобный интерпретатор *%python.sql*, который по своему использованию похож на **Apache Spark** в **Zeppelin** и позволяет использовать язык **SQL** для запроса к датафреймам **Pandas** и визуализации результатов через встроенную систему отображения таблиц Table Display System.

Предварительные настройки:

- Pandas *pip install pandas*
- PandaSQL *pip install -U pandasql*

В случае, если по умолчанию выбран интерпретатор **Python** (первый в списке интерпретаторов под значком шестеренки), можно его указывать как просто *%sql*:

```
%python
import matplotlib.pyplot as plt
plt.figure()
x = [1, 2, 3, 4, 5, 6, 7, 8]
y = [20, 21, 20.5, 20.81, 21.0, 21.48, 22.0, 21.89]

plt.plot(x, y, linestyle='dashed', marker='o', color='red')
zeppelin_show(plt,width='400px')
plt.close()
```

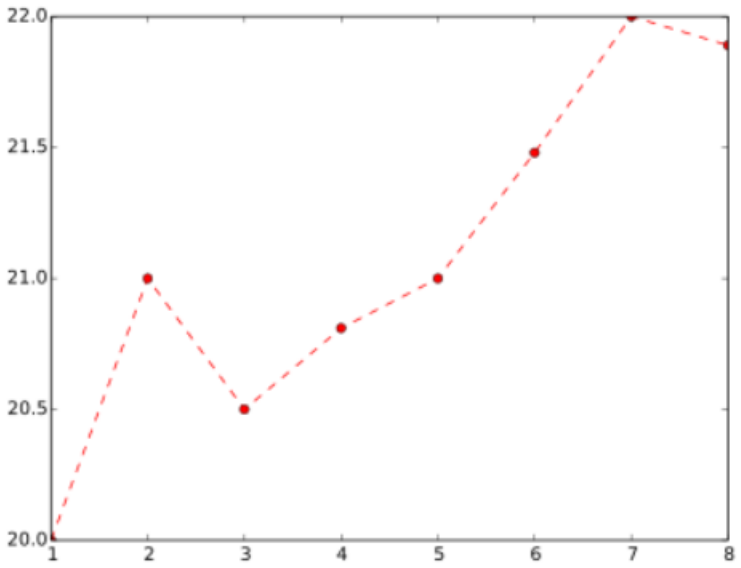

Рис.7.1.: Интеграция Matplotlib

- Первый параграф:

```
import pandas as pd
rates = pd.read_csv("bank.csv", sep=";")
```

- Следующий параграф:

```
%sql
SELECT * FROM rates WHERE age < 40
```

В противном случае – *%python.sql*.

7.9 Техническое описание

Подробные технические сведения о текущей реализации приведены по ссылке python/README.md.

Некоторые функции, еще не реализованные в интерпретаторе **Python**:

- Прерывание выполнения параграфа (способ *cancel ()*) в настоящее время поддерживается только в системах **Linux** и **MacOs**. Если интерпретатор запущен в другой ОС (например, в **MS Windows**), прерывание параграфа завершает работу всего процесса интерпретатора. **JIRA** ticket (*ZEPPELIN-893*) открыт для реализации этой функции в следующей версии интерпретатора;
- Строка состояния в web-интерфейсе (метод *getProgress()*) в настоящее время не реализована;
- Завершение кода в настоящее время не реализовано.