1. Application configuration
   1. Database
      1. Database driver

Several things must be in place to interact with a database from a Camel route:

* The database driver (MySQL Java connector bundle jar) must be installed on the target Karaf container

This MySQL connector is OSGI-compatible and already available in the maven central repository.

If it hadn't been the case the jar could have been deployed to Nexus and wrapped around a bundle.

fusedev>container-connect pgw

fusedev>osgi:install -s [wrap:]mvn:mysql/mysql-connector-java/5.1.39

fusedev>osgi:list

* The camel jdbc (or sql) feature must be enabled on the target container

Most of the capabilities of the Fuse framework are packaged in “features” that are not enabled by default for lightness reasons. A feature basically represents a set of related jar files. camel-jdbc, camel-amq, camel-http, camel-jetty… are such features.

Features can be integrated into the application profile (preferred way) or be part of another profile that will be deployed in conjunction with the application one

fusedev>profile-edit -f <featureName>/0.0.0 <application-profile-name>

fusedev>profile-edit -f <featureName>/0.0.0 <application-profile-name>

* The API bundles must be installed on the target container

The camel-jdbc and sql features (related to database persistence) make use of low level APIs from Apache that also have to be deployed on the application container.

fusedev>feature:list

fusedev>container-connect pgw

fusedev>osgi:install -s mvn:commons-pool/commons-pool/1.6

The Fabric8 maven plugin includes some parameters to integrate all the above configuration in the pom.xml file of the application, which prevents us from having to type all the above commands by hand. When the application is deployed, Fuse automatically deploys the listed features and bundles. This makes the application self-contained, and the application can therefore safely be moved from one container to another.

The resulting blueprint configuration is as follows:

<fabric8.bundles>

mvn:mysql/mysql-connector-java/5.1.39

mvn:commons-dbcp/commons-dbcp/1.4

mvn:commons-pool/commons-pool/1.6

</fabric8.bundles>

<fabric8.features>

spring camel-sql camel-jdbc jdbc

camel-blueprint

</fabric8.features>

* + 1. Datasource
* Finally a datasource bean must be configured in the blueprint.xml file

<bean class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource" id="mysqldatasourceBean">

<property name="url" value="jdbc:mysql://10.40.11.26:3306/fusedb" />

<property name="user" value="admin" />

<property name="password" value="admin" />

</bean>

* + 1. Datasource OSGI service

The datasource is here integrated into the application. It could have also have been turned into an OSGI service, bundled into its own profile and separately deployed to the container.

In such a case the other applications would have been able to get a reference to this service by looking it up in the OSGI service registry.

Datasource profile

*<bean class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource" id="mysqldatasourceBean">*

<property name="url" value="jdbc:mysql://10.40.11.26:3306/fusedb" />

<property name="user" value="admin" />

<property name="password" value="admin" />

*</bean>*

*<service interface="javax.sql.DataSource" ref="mysqldatasourceBean">*

*<service-properties>*

*<entry key="osgi.jndi.service.name" value="jdbc/mysqldatasource"/>*

*</service-properties>*

*</service>*

Application profile

*<reference id="mysqldataSourceSVC" interface="javax.sql.DataSource">*

*<filter="(osgi.jndi.service.name=jdbc/mysqldatasource)">*

*</reference>*

* + 1. Camel-jdbc vs camel-sql

In addition the to high-level object API such as JPA and hibernate, camel has 2 low-level APIs to write into a database.

* **Camel-jdbc**

The sql query is in the body of the request when the database is called using the datasource bean in the URI.

*<to uri="jdbc:mysqldatasourceBean"/>*

With this method several different datasource can be called easily.

* **Camel-SQL**

Camel sql uses the sql query in the URI. The message body can be used to store query parameters in a Map but it's easier to use them directly from header properties. As the datasource is not mentioned in the command, it has to be integrated into the sql component itself. Only one datasource can be used with this method that offers a more flexible way to update the sql queries.

*<bean class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource" id="mysqldatasourceBean">*

*<property name="url" value="jdbc:mysql://localhost:3306/fusedb"/>*

*<property name="user" value="admin"/>*

*<property name="password" value="admin"/>*

*</bean>*

*<bean id="sql" class="org.apache.camel.component.sql.SqlComponent">*

*<property name="dataSource" ref="mysqldatasourceBean" />*

*</bean>*

*…*

*<to uri=”sql:insert into sms(groupid…) values (:#${header.groupId});” >*

* 1. Fabric8 properties

Fabric8 profile contains 2 types of artifacts:

- bundles

- configuration files

The maven bundle plugin automatically add properties file to the generated profile if they are named <file>.properties and are located under src/main/fabric8.

In Blueprint, those properties file need to be declared in a Property placeholder section and have an identifier corresponding to the name of the file without “.properties”.

*<cm:property-placeholder id="pgw.placeholder" persistent-id="pgw"/> // for a file named pgw.properties.*

Then the properties can be used in the Camel code using the syntax {{property\_name}}.

* 1. Web services
     1. CXF endpoints

CXF endpoints are defined in a Camel context as beans.

The endpoint definition is based on a SOAP service. A SOAP service is defined in a wsdl file by the combination of a service definition and a service binding.

<cxf:cxfEndpoint

id="**saajCXFBean**"

address="<http://localhost:9090/attachment>"

endpointName="s:AttachServiceSoap"

serviceName="s:AttachService"

wsdlURL="saaj.wsdl" xmlns:s="http://tempuri.org/">

<cxf:properties>

<entry key="dataFormat" value="PAYLOAD"/>

</cxf:properties>

</cxf:cxfEndpoint>

...

<from uri="cxf:bean:**saajCXFBean**"/>