Leading Open Source Middleware

JOnAS - Camel Guide

JOnAS Team ( )

- -

Table of Contents

Chapter 1. Introduction

The goal of this document is to describe the Camel OSGi service and how to use it in your applications.

Camel is a light ESB (Enterprise Service Bus) developped by Apache (see http://camel.apache.org/). The service provided here is a wrapper of the existing Camel OSGi component. The goal of this wrapper is to manage all the contexts of camel that are running on the gateway in order to concentrate the knowledge of the infrastructure. Therefore, it will be easier to monitor the bus. When an application will use the ESB, it will have to use the service, which will provide the application a new context to work on (deploy new endpoints, new routes, ...).

Chapter 2. Structure and features

The Camel service relies on the OSGi bundles of Camel. It uses iPOJO to provides services on the OSGi gateway.

2.1. The Camel wrapper

The service implementation contains a wrapper of a Camel context, which is instantiated by the service when an application requires one. This wrapper contains :

- A DefaultCamelContext object. This is the Camel context.
- A String that contains the name of the Camel context. This name is used to identify the context in the applications that are using it.
- A FileRegistryComponent object. This is a camel component automatically associated with each Camel context instantiated by the service. It provides a registry that allows bindings between a logical endpoint name and technical one. Instead of typing an entire endpoint definition to configure a route, it is just needed to give the logical entry in the registry.
  Example :

```
                            this.from("registry:cxfEndpoint")
```

will be a lot easier to write than

```
                            this.from("cxf://http://localhost:9000/SayHello?
serviceClass=org.ow2.jonas.samples.camel.example.cxf.webservice.
                                api.ISayHello&dataFormat=POJO")
```

These entries are set from an xml file, that matches the following XSD :

```
<schema targetNamespace="org.ow2.jonas.samples.camel.registry.impl.file:FileRegistry"
  elementFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="org.ow2.jonas.samples.camel.registry.impl.file:FileRegistry">
  <element name="registry" type="tns:registry"></element>

  <complexType name="registry">
    <sequence maxOccurs="unbounded" minOccurs="0">
      <element name="entry" type="tns:entry"></element>
    </sequence>
  </complexType>

  <complexType name="entry">
    <sequence maxOccurs="1" minOccurs="1">
      <element name="logicalName" type="string"></element>
      <element name="technicalName" type="string"></element>
    </sequence>
  </complexType>
</schema>
```

Therefore, the file that allows the developer to use the previous example is :

```
<registry xmlns="org.ow2.jonas.samples.camel.registry.impl.file:FileRegistry"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="org.ow2.jonas.samples.camel.registry.impl.file:FileRegistry
 FileRegistry.xsd">
  <entry>
    <logicalName>cxfEndpoint</logicalName>
```

```
    <technicalName>
    <![CDATA[
        cxf://
        http://localhost:9000/SayHello?
        serviceClass=org.ow2.jonas.samples.camel.example.cxf.webservice.api.ISayHello&
        dataFormat=POJO
    ] ]>
    </technicalName>
  </entry>
</registry>
```

2.2. The service

The service is then able to trigger some useful actions on the Camel contexts it owns, described below. See the javadoc for more details.

- Start a Camel context. A reference is return. Example :

```
    // Start a new context for the application
    this.camelContextName = this.camelService.startNewContext();
```

- Stop a Camel context giving its name. Example :

```
    // Stop a Camel context
    this.camelService.stop(this.camelContextName);;
```

- Add routes on a given Camel context. Example :

```
    ...

    // Prepare a route to add in the created context
    RouteBuilder builder = new RouteBuilder() {
        @Override
        public void configure() throws Exception {
            this.from("joram:queue:queueSample").to("file:///tmp/test");
        }
    };

    // Add the route in the camel context.
    this.camelService.addRoutes(builder, this.camelContextName);

    ...
```

- Add entries to the registry from an xml file. Example :

```
    // Add the registry entries
    ClassLoader cl = this.getClass().getClassLoader();
    InputStream input = cl.getResourceAsStream("registry.xml");
    this.camelService.addRegistry(input, this.camelContextName);
```

- Remove entries from the registry. Example :

```
// Remove entries from the registry
ClassLoader cl = this.getClass().getClassLoader();
InputStream input = cl.getResourceAsStream("registry.xml");
this.camelService.removeRegistry(input, this.camelContextName);
```

- Add component on a given Camel context. For example, add a JMSComponent that uses a JORAM factory, and then use it in a route. Example :

```
// Add the JORAM component
JmsComponent joram = new JmsComponent();
ConnectionFactory connectionFactory;

connectionFactory = (ConnectionFactory) new InitialContext().lookup("CF");

joram.setConnectionFactory(connectionFactory);
JndiDestinationResolver jndiDestinationResolver = new JndiDestinationResolver();
jndiDestinationResolver.setCache(true);

joram.setDestinationResolver(jndiDestinationResolver);

this.camelService.addComponent("joram", joram, this.camelContextName);

...

// Prepare a route to add in the created context
RouteBuilder builder = new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        this.from("joram:queue:queueSample").to("file:///tmp/test");
    }
};

...
```

Chapter 3. Deployment

3.1. Deploy the service

To deploy the service on the OSGi gateway provided by JOnAS 5, the file `deployment_plans/camel.xml` needs to be put in the `/deploy` directory of the running JOnAS 5. You must see in the command line that the Camel service is started.

3.2. Deploy examples

To deploy the examples, the service must be deployed first. Then, the same way as for the service, copy the file `deployment_plans/camel-jms-example.xml` to load the bundle containing the example to run.