PEtALS-SE-Pojo

This document explain how to install and configure the petals-se-pojo JBI component.

PEtALS Team
Marie Sauvage <>
- June 2007 -

Table of Contents

List of Tables

PETALS-SE-POJO

The POJO (Plain Old Java Object) Service Engine allows to deploy Java classes as Services. It can be usefull to develop simple business processes or workflows.

A POJO service is deployed as a ServiceUnit on the petals-engine-pojo component.

The POJO serviceEndpoint is activated in the JBI environment with the information contained in the ServiceUnit descriptor file.

The POJO service can interact with the JBI environment as a JBI component.

Chapter 1. Component Configuration

The following attributes can be set during the installation phase to configure the component, using the params element of the `jbi-install-component` ANT task:

no configuration for this component

Table 1.1. component installation configuration attributes

Attribute	Description	Default	Required

Chapter 2. Service Configuration

2.1. Expose a Java class as service

PROVIDE SERVICE : Expose a Java class as a Service

2.1.1. Service Unit descriptor

The POJO class and other required libraries have to be compiled and available as JAR files at the top directory of the structure of the service unit.

Take a look at Service Assembly pojoService.zip. It contains a Helloworld POJO that log the In message and return, if the MEP allows a response, a helloworld message.

The Service Unit descriptor file (`jbi.xml`) looks like this :

```
<?xml version="1.0" encoding="UTF-8"?><jbi:jbi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:petals="http://petals.ow2.org/extensions"
    xmlns:jbi="http://java.sun.com/xml/ns/jbi"
    version="1.0">
 <jbi:services binding-component="true">
  <jbi:provides interface-name="petals:Pojo"
   service-name="petals:servicePojo"
   endpoint-name="endpointPojo">
   <petals:params>
    <petals:param name="class-name">org.ow2.petals.samples.pojo.SamplePojoService</petals:param>
   </petals:params>
  </jbi:provides>
 </jbi:services>
</jbi:jbi>
```

JMS communication attributes :

Table 2.1. service-unit attributes to provide services

Attribute	Description	Default	Required
provides	Name of the JBI service that will be activated to expose the JMS Destination into the JBI environment. interface (qname), service (qname) and endpoint (string) name are required.		
class-name	the name of the Java class to use.		Yes

2.1.2. Service Unit content

The Service Unit has to contain the following elements, packaged in an archive:

- The META-INF/jbi.xml descriptor file, has described above,
- A jar containing the java class to expose

```
service-unit.zip
  + META-INF
       - jbi.xml (as defined above)
       - mypojoclasses.jar
```

Chapter 3. POJO Service Specifications

A Java class acting as a POJO service must follow the rules :

- no specific interface implementationis required.
- if a setXXX(ComponentContext context) setter method is defined, it is called with the ComponentContext of the petals-engine-pojo component.
- if a setXXX(DeliveryChannel channel) setter method is defined, it is called with the DeliveryChannel of the petals-engine-pojo component.
- if a setXXX(Logger logger) setter method is defined, it is called with the Logger of the petals-engine-pojo component.
- if a void init() method is defined, it is called when the service unit has been deployed.
- a boolean onXXX(Exchange exchange) MUST be provided.
- all methods can throw exceptions.

A sample class following those rules :

```
public class SamplePojoService {

  public void setChannel(DeliveryChannel channel) {
    this.channel = channel;
  }
  public void setCtx(ComponentContext ctx) {
    this.ctx = ctx;
  }
  public void setLogger(Logger logger) {
    this.logger = logger;
  }

  public boolean onExchange(Exchange exchange)
    throws Exception {
    [...]
    channel.sendSync(anotherExchange);
    [...]
    return false;
  }

  public void init() {
    logger.log(Level.INFO, "SamplePojo inits.");
  }
}
```

The method onXXX(Exchange exchange) is called when a MessageExchange for the activated serviceEndpoint is received on the petals-engine-pojo component. The POJO service has to do its process here.

If the POJO service sets a response on the exchange and the exchange pattern is InOut or InOptionalOut, the method has to return true. Then the petals-engine-pojo component sends the exchange back.

Otherwise, the method has to return false and the petals-engine-pojo component sends a DONE status back.

If the method throws an exception, the petals-engine-pojo component sends a Fault message back, if the pattern permits it.

Chapter 4. Samples

See the following Service Assemblies samples that illustrate the configuration of this component :

http://wiki.petals.objectweb.org/xwiki/bin/download/Components.Engine/pojo/pojoService.zip

This sample contains a Helloworld POJO that log the In message and return, if the MEP allows a response, a *helloworld* message.