PEtALS-SE-POJO

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

PEtALS Team
Marie SAUVAGE <>
Roland NAUDIN <roland.naudin@ebmwebsourcing.com>

- September 2008 -

Table of Contents

List of Tables

PETALS-SE-POJO

The POJO (Plain Old Java Object) Service Engine allows to expose your own Java classes as Services. It is useful to refactor your own Business logic as services.

Each POJO service is represented by a `provides` section in a POJO Service Unit.

This component provides only services and doesn't act as a consumer of service.

This component is based on the PEtALS CDK.

Chapter 1. Component Configuration

no specific configuration for this component

Table 1.1. Configuration of the component (CDK)

Parameter	Description	Default	Required	Scope
acceptor-pool-size	The size of the thread pool used to accept Message Exchange from the NMR. Once a message is accepted, its processing is delegated to the processor pool thread.	5	Yes	Runtime
processor-pool-size	The size of the thread pool used to process Message Exchanges. Once a message is accepted, its processing is delegated to one of the thread of this pool.	10	Yes	Runtime
performance-notifications	Enable the performance notifications in the component. The CDK proposes to a performance notification feature to the component implementor. If you enable this feature, you must use the related method accessible in the `AbstractComponent` class.	-	No	Runtime
performance-step	When the performance notification feature is enabled, it is possible to define a step on the notifications. When there is an heavy message traffic, it is recommanded to increase this step to avoid performance disturbance.	-	No	Runtime
properties-file	Name of the file containing properties used as reference by other parameters. Parameters reference the property name in the following pattern `${myPropertyName}`. At runtime, the expression is replaced by the value of the property. The value of this parameter is : • an URL • a file relative to the PEtALS installation path	-	No	Installation
ignored-status	When the component receives an acknowledgement message exchange, it can skip the processing of these message according to the type of the acknowledgment. If you decide to not ignore some acknowledgement, the component listeners must take care of them. Accepted values : `DONE_AND_ERROR_IGNORED`, `DONE_IGNORED`, `ERROR_IGNORED` or `NOTHING_IGNORED`	`DONE_AND_ERROR_IGNORED`	Yes	Component
jbi-listener-class-name	Qualified name of the class extending **AbstractJBIListener**	-	Yes	Component
external-listener-class-name	Qualified name of the class extending **AbstractExternalListener**	-	No	Component

Definition of CDK parameter scope :

- *Component* : The parameter has been defined during the development of the component. A user of the component can not change its value.
- *Installation*: The parameter can be set during the installation of the component, by using the installation MBean (see JBI specifications for details about the installation sequence). If the parameter is optional and has not been defined during the development of the component, it is not available at installation time.
- *Runtime* : The paramater can be set during the installation of the component and during runtime. The runtime confguration can be changed using the CDK custom MBean named `RuntimeConfiguration`. If the parameter is optional and has not been defined during the development of the component, it is not available at installation and runtime times.

Chapter 2. Service Configuration

2.1. Expose a business logic as service

PROVIDE SERVICE : Expose a Java class as a Service

2.1.1. Service Unit descriptor

The POJO class(es) and their depending libraries must be set in JAR(s) file(s) at the root directory of the POJO Service Unit package.

The POJO JBI descriptor must contain a `provides` section for each POJO to expose in the JBI bus. For each provides section, the name of the POJO class must be filled.

Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- JBI descriptor for PEtALS' "petals-se-pojo" (POJO), version 2.0 -->
<jbi:jbi version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jbi="http://java.sun.com/xml/ns/jbi"
  xmlns:pojo="http://petals.ow2.org/components/pojo/version-2.0"
  xmlns:petalsCDK="http://petals.ow2.org/components/extensions/version-4.0"
  xmlns:generatedNs="http://POJO/test">

  <!-- Import a Service into PEtALS or Expose a PEtALS Service => use a BC. -->
  <jbi:services binding-component="false">

    <!-- Import a Service into PEtALS => provides a Service. -->
    <jbi:provides
      interface-name="generatedNs:POJO"
      service-name="generatedNs:POJOService"
      endpoint-name="POJOServiceEndpoint">

      <!-- CDK specific elements -->
      <petalsCDK:wsdl xsi:nil="true" />

      <!-- Component specific elements -->
      <pojo:class-name>test.SamplePojoService</pojo:class-name>
    </jbi:provides>
  </jbi:services>
</jbi:jbi>
```

Table 2.1. Service Unit attributes to provide services

Attribute	Description	Default	Required
class-name	The name of the Java class to expose as a service.	-	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,
- At least a jar containing the POJO 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 these rules :

- no specific interface implementation is required.
- if a `public setComponentContext(ComponentContext context)` setter method is defined, the component set its `ComponentContext` instance with this method at the initialization of the POJO.
- if a `public setDeliveryChannel(DeliveryChannel channel)` setter method is defined, the component set its `DeliveryChannel` instance with this method at the initialization of the POJO.
- if a `public setJBIListener(AbstractJBIListener jbiListener)` setter method is defined, the component set its `JBIListener` instance with this method at the initialization of the POJO.
- if a `public setLogger(Logger logger)` setter method is defined, the component set its `Logger` instance with this method at the initialization of the POJO.
- if a `public void init()` method is defined, the compenent invoke it at the initialization of the POJO..
- a `public boolean onExchange(Exchange exchange)` MUST be provided.
- all methods can throw exceptions.

A sample class following those rules :

```
package test;

import java.util.logging.Level;
import java.util.logging.Logger;

import javax.jbi.component.ComponentContext;

import org.ow2.petals.component.framework.api.message.Exchange;
import org.ow2.petals.component.framework.listener.AbstractJBIListener;

public class SamplePojoService {

  AbstractJBIListener jbiListener;
  Logger logger;
  ComponentContext ctx;

  public void setJBIListener(AbstractJBIListener jbiListener) {
    this.jbiListener = jbiListener;
  }
  public void setComponentContext(ComponentContext ctx) {
    this.ctx = ctx;
  }
  public void setLogger(Logger logger) {
    this.logger = logger;
  }

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

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

The method `onExchange(Exchange exchange)` is invoked when a MessageExchange targeting the POJO service is received on the component. The POJO service must process the service here.

If the POJO service sets a response on the exchange to support patterns `InOut` or `InOptionalOut`, the method has to return `true`. The component will send the response back.

Otherwise, the method return `false` and the component responds with a DONE acknowledgement.

If any exception if raised, depending of the nature of the exception (`RuntimeException` , `FaultException`...) and the pattern, a Fault or an ERROR acknowledgement would be sent back.