PEtALS-SE-JSR181

This document explains how to install, configure and use the petals-se-jsr181 JBI component.

PEtALS Team
Christophe HAMERLING <christophe.hamerling@ebmwebsourcing.com>

- October 2008 -

Table of Contents

List of Tables

PETALS-SE-JSR181

This Service Engine allows to expose an annotated POJO as a JBI Service in the bus.

This component is based on the PEtALS Component Development Kit and on Apache Axis2 for invocation and marshalling.

If you want more details about jsr181, you can consult this the specification on the Java Community Process portal : http://jcp.org/en/jsr/detail?id=181

Chapter 1. Features

The petals-se-jsr181 component is based on the petals-cdk v4.0.x for JBI compatibility and on Apache Axis2 v1.4 for JSR181 support. It provides the following features :

- Expose a POJO as JBI service

Chapter 2. Component Configuration

The component can be configured through its JBI descriptor file like this :

Note

Actually, nothing is configurable in this JBI descriptor, so leave it as it is.

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi:jbi version="1.0" xmlns:jbi="http://java.sun.com/xml/ns/jbi"
 xmlns:petalsCDK="http://petals.ow2.org/components/extensions/version-4.0"
 xmlns:jsr181="http://petals.ow2.org/components/jsr181/version-1.0>
 <jbi:component type="service-engine"
  bootstrap-class-loader-delegation="parent-first">
  <jbi:identification>
   <jbi:name>petals-se-jsr181</jbi:name>
   <jbi:description> The jsr181 Service Engine (based on Axis2)</jbi:description>
  </jbi:identification>
  <jbi:component-class-name>org.ow2.petals.se.jsr181.Component</jbi:component-class-name>
  <jbi:component-class-path>...</jbi:component-class-path>
  <jbi:bootstrap-class-name>org.ow2.petals.se.jsr181.Bootstrap</jbi:bootstrap-class-name>
  <jbi:bootstrap-class-path>...</jbi:bootstrap-class-path>

  <!-- Component Development Kit Parameters -->
  <petalsCDK:acceptor-pool-size>5</petalsCDK:acceptor-pool-size>
  <petalsCDK:processor-pool-size>10</petalsCDK:processor-pool-size>
  <petalsCDK:ignored-status>DONE_AND_ERROR_IGNORED</petalsCDK:ignored-status>
  <petalsCDK:properties-file />
  <petalsCDK:performance-notifications>false</petalsCDK:performance-notifications>
 </jbi:component>
</jbi:jbi>
```

Warning

The class name values in italic should not be modified by the user.

Table 2.1. 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

Table 2.2. Interceptors configuration for Service Unit (CDK)

Parameter	Description	Default	Required
name	Name of the interceptor to use. It must refer to a component interceptor name.	-	Yes

The jsr181 component specific parameters can be also set through JMX during its installation phase.

Chapter 3. Service Configuration

3.1. Expose a Java Class as JBI Service

PROVIDE SERVICE : Expose an Java Class as Service in the JBI environment

The petals-se-jsr181 component can expose a Java Class as JBI ServiceEndpoint. This is done by deploying a Service Unit on it.

When a message is received on a JSR181 linked endpoint from the JBI environment, it is mapped to an Axis2 message and sent to the Axis2 runtime. The linked Java Class is called and the response is processed and returned to the JBI environment.

3.1.1. Service Unit descriptor

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

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi:jbi version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jbi="http://java.sun.com/xml/ns/jbi"
  xmlns:petalsCDK="http://petals.ow2.org/components/extensions/version-4.0"
  xmlns:helloworld="http://petals.ow2.org/helloworld"
  xmlns:jsr181="http://petals.ow2.org/components/jsr181/version-1.0">

  <jbi:services binding-component="false">
    <jbi:provides interface-name="helloworld:Helloworld" service-name="helloworld:HelloworldService"
 endpoint-name="HelloworldEndpoint">
      <petalsCDK:wsdl>Service.wsdl</petalsCDK:wsdl>
      <jsr181:class>org.ow2.petals.usecase.jsr181.TestService</jsr181:class>
    </jbi:provides>
  </jbi:services>
</jbi:jbi>
```

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.

Table 3.1. Configuration of a Service Unit to provide a service (CDK)

Parameter	Description	Default	Required
wsdl-imports-download	If false, the external imports declared in the service WSDL won't be downloaded, so they won't be replaced by their content.	True	No
wsdl	Path to the WSDL document describing services and operations exposed by the provided JBI endpoints defined in the SU. The value of this parameter is : • an URL • a file relative to the root of the SU package If not specified, a basic WSDL description is automaticaly provided by the CDK.	-	No
timeout	Timeout in milliseconds of a synchronous send. this parameter can be used in conjunction with the `sendSync(Exchange exchange)` method of the Listeners. Set 0 for an infinite timeout.	-	No
org.ow2.petals.messaging.provider.nack	Check PEtALS container document for further details. This propety activates the bypass of acknowledgment messages destinated to this SU.	-	No

Table 3.2. Configuration of a Service Unit to provide a service (jsr181)

Parameter	Description	Default	Required
class	The JSR181 annotated class which will provide the Service.This class must be available in the Service Unit class loader.	-	Yes

3.1.2. Annotated class

The following class is a sample of a JSR181 annotated class (taken from the PEtALS usecases sources) :

```
package org.ow2.petals.usecase.jsr181;

import java.text.SimpleDateFormat;
import java.util.Date;

import javax.jws.WebMethod;
import javax.jws.WebService;

/**
 *
 * @author chamerling - eBM WebSourcing
 *
 *
 * NOTE : The
 * @WebService parameters are not used by the component, the service name is
 *             defined in the service unit. Need to modify the CDK to be able to
 *             create the endpoint from these values...
 */
@WebService(serviceName = "Hello", name = "MyService", targetNamespace = "http://petals.ow2.org")
public class TestService {

    /**
     * Say hello to the world !
     */
    @WebMethod
    public String sayHello(String str) {
        System.out.println("Hey! This is the sayHello operation.");
        return "You say me " + str;
```

```
    }

    /**
     * Get a person from its id only to test 'complex' data binding.
     *
     * @param id
     * @return
     */
    @WebMethod
    public Person getPerson(int id) {
        System.out.println("Get person " + id);
        return new Person(id, "Christophe", "Hamerling", 29, "France");
    }

    /**
     *
     * @return
     */
    @WebMethod
    public String getTime() {
        System.out.println("Get time");
        return new SimpleDateFormat().format(new Date(System.currentTimeMillis()));
    }

    /**
     * NOP
     */
    @WebMethod
    public void voidvoid() {
        System.out.println("The Void operation");
    }

    /**
     * The final operation will be 'specializedOperation'
     */
    @WebMethod(operationName = "specializedOperation")
    public void operation() {
        System.out.println("The specialized operation");
    }

    /**
     *
     * @throws Exception
     */
    @WebMethod
    public String iAmThrowingAnException() throws Exception {
        System.out.println("throw exception");
        throw new Exception("This is a server side Exception");
    }
}
```

The main annotations you may use are :

- The **@WebService** annotation is mandatory and is used by the Axis2 engine to build the service. You can specialize the service name, target namespace and more with the annotation parameter.
- The **@WebMethod** annotation is used to delare the that the method will be see as a JBI operation. You can specialize the operation name and more with the annotation parameters.

More (or less) information is available on the Apache Axis2 page http://ws.apache.org/axis2/1_4/pojoguide.html#jsr181pojows.

Before sending the JBI message to the Axis2 service, the JBI Service Engine will check if :

- If the requested operation exists. If not, an error will be returned in the JBI message exchange.1
- The JBI Message Exchange Pattern (MEP) is compatible with the target operation. For example, in the previous code snippet, an InOut MEP is not compatible with the 'voidvoid' operation and an error will be returned in the JBI message exchange.

With the previous code samples, you can now call the *{http://petals.ow2.org/helloworld}HelloworldService* and operation *sayHello* with an *InOut* MEP with a JBI message payload like '`<sayHello><param0>Hey!!!</param0></sayHello>`' and you will get a response like '`<dlwmin:sayHelloResponse xmlns:dlwmin="http://petals.ow2.org" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><return>You say me Hey!!!</return></dlwmin:sayHelloResponse>`'.

Chapter 4. Samples

The jsr181 service engine samples are available as use cases section. You can find them in the project sources repository here http://svn.forge.objectweb.org/cgi-bin/viewcvs.cgi/petals/trunk/petals-demos/petals-usecases/petals-jsr181/

We recommend you to check this source code to create the JSR181 class and the Service Unit.

Chapter 5. Limitations

1. The WSDL description is not dynamically generated from the Java class at runtime. This is due to an Axis2 and PEtALS class loader issue.

   Youo can find an example on how to generate and include the WSDL description from the Java class with some Maven2 plugins here http://svn.forge.objectweb.org/cgi-bin/viewcvs.cgi/petals/trunk/petals-demos/petals-usecases/petals-jsr181/petals-jsr181-hello/petals-jsr181-suprovide/.

2. The JBI Component Context is not available on the POJO so you can not access to the JBI Delivery Channel to do some light orchestration. This feature will be added soon!