PEtALS-SE-EIP 1.0

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

PEtALS Team
Guillaume Decarnin <guillaume.decarnin@openwide.fr>

- August 2007 -

Table of Contents

Preface 3
1. Operations 4
    1.1. Process 4
    1.2. getEipExtension 4
    1.3. setEipExtension 4
2. Orchestration format 5
    2.1. The "extension" element 5
    2.2. The "call" element 5
    2.3. The execution context 5
        2.3.1. The "chain" element 5
        2.3.2. The "dispatch" element 6
        2.3.3. The "aggregate" element 6
    2.4. Dynamic calls 6
        2.4.1. The "get-calls-by-service" element 6
        2.4.2. The "get-calls-by-xpath" element 6
    2.5. Conditions 7
        2.5.1. The "if" element 7
        2.5.2. The "choose" element 7
        2.5.3. Samples of conditions 7
    2.6. Imbrications 7
        2.6.1. Samples of imbrications 8
    2.7. Patterns 8
    2.8. Sample 8
        2.8.1. Service Assembly sample for EIP component 8
        2.8.2. SampleTestService 9

Preface

This component implements some Enterprise Integration Patterns. The component features are:

- chain services calls
- dispatch messages
- aggregate the results of services calls
- dynamic orchestration with conditions, recipients list in message and integration of routing services

Chapter 1. Operations

1.1. Process

Process the orchestation using the current Service Unit configuration.

1.2. getEipExtension

Return the XML representation of the orchestration.

1.3. setEipExtension

Set a new Service Unit configuration. The content of the message is the new "extension" node to use (same as in jbi.xml).

Chapter 2. Orchestration format

The orchestration process is written in the JBI descriptor if the Service Unit.

The XML format is described in the file `src/main/jbi/META-INF/eip-extension.xsd` (see eip-extension.xsd). This schema is used to validate the configuration on Service Unit deployement and during the setEipExtension operation.

2.1. The "extension" element

The root element `extension` uses the namespace `http://petals.objectweb.org/extensions/eip/`. The root must contain one child : `chain`, `dispatch` or `aggregate`.

Here is a sample jbi.xml:

```
<jbi version="1.0" xmlns='http://java.sun.com/xml/ns/jbi'
     xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
     xmlns:petals="http://petals.objectweb.org/"
     xmlns:extensions="http://petals.objectweb.org/extensions/"
     xmlns:eip="http://petals.objectweb.org/extensions/eip/"
     xmlns:keyvalue="http://petals.objectweb.org/extensions/key-value/">
  <services binding-component="false">
    <provides interface-name="SampleEip"
        service-name="petals:SampleEipService"
        endpoint-name="SampleEipEndpoint">

      <extensions:extensions>
        <extension xmlns="http://petals.objectweb.org/extensions/eip/">
          <!-- orchestration process: -->
          <chain>
            <call service="{http://petals.objectweb.org/}SampleTestService" operation="operation1"/>
            <call service="{http://petals.objectweb.org/}SampleTestService" operation="operation2"/>
            <call service="{http://petals.objectweb.org/}SampleTestService" operation="operation3"/>
          </chain>
          <!-- end of orchestration -->
        </extension>

      </extensions:extensions>
    </provides>
  </services>
</jbi>
```

Download other jbi.xml samples : su-config-samples.zip

2.2. The "call" element

The "call" element invokes a service. The attribute `method` is optional. Its default value is *InOut*.

```
<call service="{http://petals.objectweb.org/}SampleTestService"
      operation="myOperation"
      method="InOnly|InOut|InOptionalOut|RobustInOnly"/>
```

2.3. The execution context

There are 3 execution contexts : `chain`, `dispatch` and `aggregate`. These elements define the transitions between the calls.

2.3.1. The "chain" element

In a "chain" element, the out message of a call is the in message of the next call. After the "chain" bloc, the current message is the result of the last call.

```
<chain>
```

```
  <call service="service1" operation="operation1"/>
  <call service="service2" operation="operation2"/>
  . . .
</chain>
```

2.3.2. The "dispatch" element

In a "dispatch" element, the same message is send to all the services. The order is not important. After a dispatch, the current message is the same as the initial message.

```
<dispatch>
  <call service="service1" operation="operation1"/>
  <call service="service2" operation="operation2"/>
  . . .
</dispatch>
```

2.3.3. The "aggregate" element

n an "aggregate" element, the same message is send to all the services. After an aggregate, the current message is an aggregation of all the results of the inside calls, surrounded with a "aggregate-result" tag.

```
<aggregate>
  <call service="service1" operation="operation1"/>
  <call service="service2" operation="operation2"/>
  . . .
</aggregate>
```

Result of an aggregate with a service who returns a number:

```
<aggregate-result>
  <number>3</number>
  <number>4</number>
  <number>6</number>
</aggregate-result>
```

2.4. Dynamic calls

The EIP component can generate a list of calls and execute them.

2.4.1. The "get-calls-by-service" element

This element calls a service who returns a list of calls.

```
<get-calls-by-service
  service="{http://petals.objectweb.org/}SampleTestService"
  operation="getCalls"/>
```

The operation "getCalls" have to return a message like this:

```
<calls>
  <call service="myService" operation="myOperation"/>
  <call service="myService2" operation="myOperation2"/>
  . . .
</calls>
```

See the XSD file calls.xsd.

2.4.2. The "get-calls-by-xpath" element

This element allows to read the routing path included in the message.

```
<get-calls-by-xpath
```

```
  base="/message/recipients/recipient"
  service="@service" operation="@operation"/>
```

The concatenation of the attributes "base" and "service" is a XPath, used to extract a list of services from the message. "base" and "operation" extract the operations.

Sample of message who respects the previous XPath:

```
<message>
  <recipients>
    <recipient service="service1" operation="operation1"/>
    <recipient service="service2" operation="operation2"/>
    . . .
  </recipients>
  <content>
    . . .
  </content>
</message>
```

2.5. Conditions

2.5.1. The "if" element

The "if" node has a "test" attibute who contains a XPath boolean expression. The condition is evaluated using the message content. If the condition returns "true", then the bloc inside the "if" is executed.

```
<if test="xpathBooleanExpression">
  . . .
</if>
```

2.5.2. The "choose" element

The "choose" element has two children: "when" and "otherwise". "when" has a "test" attribute, like the "if" element. If the condition is true, the "when" bloc is executed, else the "otherwise" bloc is executed.

```
<choose>
  <when test="xpathBooleanExpression">
    . . .
  </when>
  <otherwise>
    . . .
  </otherwise>
</choose>
```

2.5.3. Samples of conditions

Condition based on the type of the message (root name):

```
<if test="name(/*) = 'rootElement'">
  . . .
</if>
```

Condition based on message content:

```
<if test="sum(/items/item/value) > 100">
  . . .
</if>
```

2.6. Imbrications

It is possible to imbricate the elements. Any bloc can contain these elements:

- call
- get-calls-by-service
- get-calls-by-xpath
- if
- choose
- chain
- dispatch
- aggregate

2.6.1. Samples of imbrications

In a chain, a message is send to 2 services with a dispatch:

```
<chain>
  <call .../>
  <call .../>
  <dispatch>
    <call .../>
    <call .../>
  </dispatch>
</chain>
```

The branches of a parallel send are chain blocs, in order to make complex process:

```
<dispatch>
  <chain>
    <call .../>
    <call .../>
  </chain>
  <chain>
    <call .../>
    <call .../>
    <call .../>
  </chain>
</dispatch>
```

In an aggregate, an element is the result of a chain:

```
<aggregate>
  <call .../>
  <chain>
    <call .../>
    <call .../>
    <call .../>
  </chain>
  <call .../>
</aggregate>
```

2.7. Patterns

See PEtALS Service Engines and Enterprise Integration Patterns for implementation samples of EIP patterns.

2.8. Sample

2.8.1. Service Assembly sample for EIP component

The SA sample sampleeip-sa.zip is provided with this configuration:

```
<extension xmlns="http://petals.objectweb.org/extensions/eip/">
  <chain>
    <call service="{http://petals.objectweb.org/}SampleTestService" operation="inc"/>
    <call service="{http://petals.objectweb.org/}SampleTestService" operation="double"/>
  </chain>
</extension>
```

It needs the SampleTestService, described below.

2.8.2. SampleTestService

This service provides simple operations on text or number. It can be used to test the EIP component.

- Service Engine: petals-se-test-1.4.zip
- Service Assembly: sampletest-sa.zip

2.8.2.1. Operations on numbers

Message format:

```
<number>5</number>
```

- **inc** : increment the number
- **double** : multiplication by 2
- **triple** : multiplication by 3

2.8.2.2. Operations on text

Message format:

```
<text>hello!</text>
```

- **trim** : remove leading and trailing whitespaces
- **clone** : return "hello!hello!"
- **uppercase** : return "HELLO!"
- **addstars** : return "*hello!*"