Spago JMS Adapter

Authors Angelo Bernabei
Gianfranco Boccalon

Document Goal

The goal of this document is to explain the implementation of the JMS adapter of Spago.

Versions history

Version/Release n° :	1.0	**Date Version/Release** :	05/10/2005
Description:	First release		

References

You can find more information about Spago framework in the following documents available on SourceForge (http://sourceforge.net/projects/spago):

[1] Spago Overview

[2] Spago User Guide.

1 Introduction

An *adapter* is a component that accepts requests on a particular channel and translates them in a channel independent format, which in case of Spago is XML.
The JMS adapter receives messages on a queue and according to the content dispatches the requests on the Spago services.

2 Adapter JMS

The messages put on the queue must be in XML and have to contain the parameter **ACTION_NAME or PAGE**, which allows to Spago to identify the service to execute.
The mechanism is similar to the AdapterEJB invocation, in fact the JMS adapter is a Message Driven Bean.
The JMS channel is stateless.

Here you find a sample message to put on a queue to execute a particular service:

```
<SERVICE_REQUEST ACTION_NAME= “MessageReader” parameter=”45” />
```

The service configuration follows the rules of a classic Spago application; in this case it’s sufficient to configure the action ***MessageReader*** in the file *actions.xml*:

```
<ACTION name="MessageReader" class="it.eng.jms.test.MessageReader" scope="SESSION">
    <CONFIG></CONFIG>
</ACTION>
```

The class that implements the service (it.eng.jms.test.MessageReader) can access the parameter “parameter” in this way:

```
String parameter = (String)request.getAttribute("parameter");
```

The adapter consists of a Message Driver Bean EJB that has to be configured properly according to the application server.

2.1.1 Adapter configuration in JBoss

If you are using JBoss, you can configure the adapter putting these information in the **ejb-jar.xml** file:

```
<message-driven>
    <ejb-name>AdapterJMS</ejb-name>
    <ejb-class>it.eng.spago.dispatching.jmschannel.JmsAdapterMDB</ejb-class>
    <transaction-type>Container</transaction-type>
    <acknowledge-mode>AUTO_ACKNOWLEDGE</acknowledge-mode>
    <message-driven-destination>
```

```
            <destination-type>javax.jms.Queue</destination-type>
        </message-driven-destination>
        <resource-ref>
            <res-ref-name>jms/QCF</res-ref-name>
            <res-type>javax.jms.QueueConnectionFactory</res-type>
            <res-auth>Container</res-auth>
        </resource-ref>
    </message-driven>
```

In the **jboss.xml** file you have to put this configuration:

```
    <message-driven>
            <ejb-name>AdapterJMS</ejb-name>
            <destination-jndi-name>queue/spagoQueue</destination-jndi-name>
            <resource-ref>
                <res-ref-name>jms/QCF</res-ref-name>
                <jndi-name>ConnectionFactory</jndi-name>
            </resource-ref>
    </message-driven>
```

The application server can handle an EJB pool that listen directly the queue (in the sample configuration of this document the queue is called *queue/spagoQueue*).
In JBoss you define the queue in the file **jbossmq-destinations-service.xml**:

```
    <mbean code="org.jboss.mq.server.jmx.Queue"
            name="jboss.mq.destination:service=Queue,name=spagoQueue">
      <depends
            optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
    </mbean>
```

Spago offers also a utility class *it.eng.spago.dispatching.jmschannel.**AdapterJMSProxy*** that is a proxy to simplify the writing of a message on a queue.