Spago JMS Adapter

Redatto da Angelo Bernabei
Gianfranco Boccalon

Scopo del Documento

Il documento ha l'obiettivo di descrivere l'implementazione dell'adapter JMS sviluppato per il framework Spago.

Informazioni sulla versione

Versione/Release n° :	1.0	**Data Versione/Release** :	05/10/2005
Descrizione modifiche:	Versione iniziale		

Riferimenti

Per ulteriori informazioni sul framework Spago si rinvia ai seguenti documenti disponibili su SourceForge (http://sourceforge.net/projects/spago):

[1] Spago Overview

[2] Spago User Guide.

1 Introduzione

Per *adapter* si intende un componente che riceve richieste su un particolare canale e le trasforma nel formato indipendente dal canale di Spago, ossia l'XML.
L'adapter JMS riceve messaggi in una coda e sulla base del contenuto è in grado di eseguire un servizio di Spago.

2 Adapter JMS

I messaggi inseriti nella coda devono essere in formato XML e contenere il parametro **ACTION_NAME** o **PAGE** che identifica il servizio da eseguire. La modalità è del tutto analoga all'AdapterEJB, di fatto l'adapter JMS è un Message Driven Bean.
Per sua natura il canale è di tipo stateless.

Ecco un esempio di messaggio da inserire in una coda JMS per attivare un determinato servizio:

```
<SERVICE_REQUEST ACTION_NAME= "MessageReader" parametro="45" />
```

La configurazione del servizio rispecchia le normali modalità tipiche del framework Spago, in questo caso è sufficiente configurare l'action ***MessageReader*** nel file *actions.xml*:

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

La classe che implementa l'action avrà a disposizione nella ServiceRequest il parametro "*parametro*" come segue:

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

L'adapter è implementato utilizzando un EJB di tipo MDB che deve essere configurato all'interno dei file XML di deploy a seconda dell'application server utilizzato.

2.1.1 Configurazione dell'adapter in JBoss

Nel caso di JBoss l'adapter va configurato inserendo le seguenti informazioni nel file **ejb-jar.xml**:

```
<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>
```

Inoltre nel file **jboss.xml** va inserita la seguente configurazione:

```
    <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>
```

L'application server è configurabile in modo che sia possibile gestire un pool di EJB che ascoltino direttamente sulla coda (che in questa configurazione di esempio si chiama *queue/spagoQueue*).
Nel caso di Jboss la coda deve essere definita nel file **jbossmq-destinations-service.xml** inserendo la seguente riga 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>
```

Inoltre in Spago è presente la classe di utility *it.eng.spago.dispatching.jmschannel.**AdapterJMSProxy*** che consiste di un Proxy per facilitare l'inserimento di un messaggio nella coda.