

Nell'articolo di oggi vedremo come mappare i servizi e gestire la loro implementazione.

Occorre prima di tutto scegliere la tecnica di gestione per i messaggi xml. Possiamo usare delle librerie di gestione di messaggi xml come JDOM o SAX oppure delle tecniche di marshalling per la trasformazione di flussi xml in classi java e viceversa, come JAXB e XMLBEANS.

Spring-ws supporta tutte le principali librerie per la gestione dei file xml pertanto la scelta della libreria deve essere fatta valutando anche l'ambiente a disposizione.

In questo caso ho scelto JDOM, una libreria opensource che semplifica la gestione di DOM. Occorre modificare il file pom.xml per introdurre la dipendenza dalla libreria JDOM. Nel nostro caso si traduce in questo set di dipendenze:

```
<dependency>          <groupId>jdom</groupId>          <artifactId>jdom</artifactId>
<version>1.0</version>
</dependency>
<dependency>
  <groupId>jaxen</groupId>
  <artifactId>jaxen</artifactId>
  <version>1.1</version>
</dependency>
<dependency>
  <groupId>javax.xml.soap</groupId>
  <artifactId>saaj-api</artifactId>
  <version>1.3</version>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>com.sun.xml.messaging.saaj</groupId>
  <artifactId>saaj-impl</artifactId>
  <version>1.3</version>
  <scope>runtime</scope>
</dependency>
```

All'interno del file `spring-ws-servlet.xml` definiamo il bean che implementa il servizio di RicercaLibro.

```
<bean id="ricercaLibroEndpoint" class="it.finazzo.libreria.ws.RicercaLibroEndpoint"/>
```

Tale classe estende la classe di Spring che gestisce l'integrazione con JDOM

```
public class RicercaLibroEndpoint extends AbstractJDomPayloadEndpoint {    protected  
Element invokeInternal(Element arg0) throws Exception {
```

```
    return null;
```

```
    } }
```

Definiti il bean ci preoccupiamo di definire il mapping dei servizi, ovvero quale classe invocare in corrispondenza di uno specifico servizio:

```
<bean  
class="org.springframework.ws.server.endpoint.mapping.PayloadRootQNameEndpointMapping  

```

```
<property name="mappings">
```

```
<props>
```

```
<prop
```

```
key="{http://www.example.org/schema}RicercaLibroRequest">ricercaLibroEndpoint</prop>
```

```
</props>
```

```
</property>
```

```
</bean>
```

In questo caso stiamo dicendo al gestore delle mappe dei servizi, la classe implementata da Spring-Ws, che il servizio RicercaLibro deve invocare l'endpoint `ricercaLibroEndpoint`. Per la gestione del mapping spring-ws offre 2 diverse implementazione da scegliere in base al contesto in cui vi trovate.

- `PayloadRootQNameEndpointMapping`
- `SoapActionEndpointMapping`

Il mapping basato su Payload prevede l'analisi della richiesta per stabilire quale servizio stiamo invocando. Tale meccanismo è più lento ma è indipendente dalla specifica SOA e pertanto può essere usato sia in caso di SOA 1.1, 1.2 o qualsiasi altro livello di trasporto.

Il secondo mapping sfrutta l'header SOAPAction, che permette di individuare il corretto routing senza analizzare tutta la risposta. Pertanto è più veloce ma è valido solo con la SOA 1.1, poichè nella SOA 1.2 tale parametro è stato reso deprecato.

A questo punto non resta che provare il nostro servizio tramite Soap Ui e verificare il corretto funzionamento del routing.

[Prossimo](#)