

Pràctica SOA

Introducció

SOA, és una Arquitectura Orientada als Serveis. En aquesta pràctica es duen a terme les diferents architectures que ens permeten desenvolupar els serveis desitjats. Aquesta memòria conté una explicació dels diferents serveis desenvolupats, la tecnologia que s'utilitza, llenguatge, tipus de projecte, classes i anotacions utilitzades, entre altres.

XML: Marshaling y Unmarshaling con JAXB

Aquesta part de la pràctica s'ha programat mitjançant Maven en un projecte Java.

Per implementar aquest projecte són necessàries dues classes:

- **Capcelera.java**: classe on es defineixen els atributs necessaris.
En el nostre cas, hem implementat aquesta classe de la forma *Simple*, que consta d'una capçalera i 4 definicions d'elements XML.
Aquesta classe utilitza les següents etiquetes:
 - **@XMLRootElement**: Element arrel que conté tots els XMLElements.
 - **@XMLAttribute**: Atribut de l'element.
 - **@XMLElement**: Element que forma part del fitxer XML.
- **Marshall.java**: aquesta classe utilitza un objecte de la classe anterior (capcelera.java).
En aquesta classe es defineixen dos mètodes que són cridats des de la funció main():
 - **marshall(File file, JAXBContext jaxbContent)**: funció que emmagatzema els elements en un fitxer en format *.XML*.
 - **unmarshall(File file, JAXBContext jaxbContent)**: funció que desenvolupa el fitxer XML per emmagatzemar-ho en un objecte.

A la següent imatge, podem veure la prova on s'executen les funcions *marshall* i *unmarshall* de forma successiva.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GrpHdr MsgId="2">
  <CtrlSum>1324</CtrlSum>
  <CreDtTm>2010-06-10T08:35:30</CreDtTm>
  <NbOfTx>3452</NbOfTx>
</GrpHdr>
Capcelera: [
Id: 2
Data: 2010-06-10T08:35:30
NumOps: 3452
CtrlSum: 1324 ]
```

Rest Web Service “enriquecido”

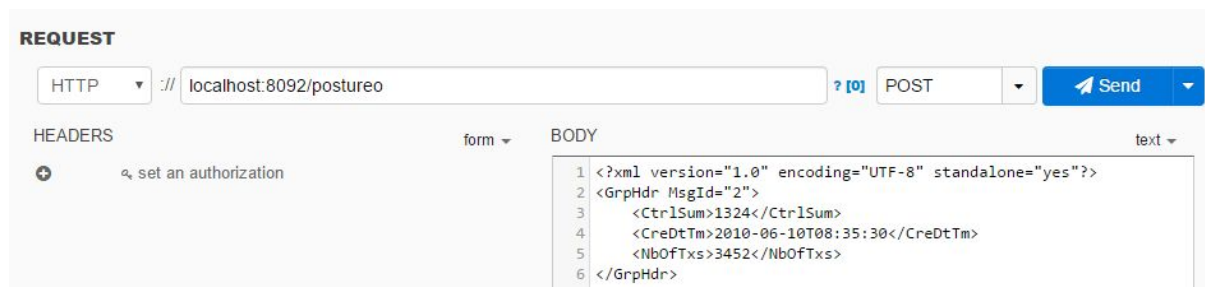
Aquesta segona part de la pràctica també està implementada amb Java utilitzant Maven. Per implementar el *Rest Web Service* hem utilitzat la classe *Capcelera* i la funció *unmarshall* de la part anterior.

En aquesta part, seguint l'esquelet proporcionat, tenim la missió de desenvolupar dues funcions que s'executaran en la part del servidor:

- **POST**: funció que rebrà el contingut del fitxer XML com a String i el transformarà a objecte (mitjançant la funció *unmarshall*) per comprovar que aquest té el format correcte. A més, si aquest és correcte, s'emmagatzemarà el contingut en una llista (ArrayList). Finalment, retornarà un identificador que identifica el XML.
- **GET**: funció que rep junt la petició l'identificador de l'XML desitjat. A partir d'aquest paràmetre es buscarà dins la llista. En cas de que aquest existeixi es retornarà, en cas contrari es retornarà un missatge d'error (error 400).
Tots aquests missatges de resposta els hem englobat dins d'un objecte de tipus JSON que construïm per poder-los enviar des del servidor al client.

Utilitzarem com a client l'extensió de Google Chrome: DHC. Amb aquesta eina podem dur a terme les peticions i així comprovar el bon funcionament d'aquesta part.

Missatge de Request per POST:



REQUEST

HTTP ? [0] POST

HEADERS form BODY text

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <GrpHdr MsgId="2">
3   <CtrlSum>1324</CtrlSum>
4   <CreDtTm>2010-06-10T08:35:30</CreDtTm>
5   <NbOfTxs>3452</NbOfTxs>
6 </GrpHdr>
```

Missatge de Response en la petició POST:



RESPONSE Elapsed Time: 1.4s

200 OK

HEADERS pretty BODY raw

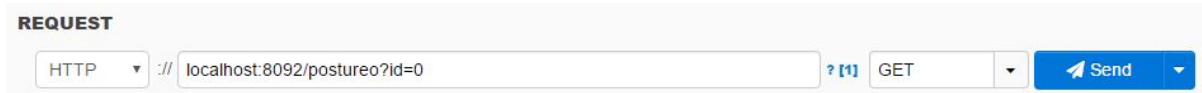
Content-Length: 1 Bytes
Content-Type: text/plain
Date: 2016 May 6 23:10:56 -10m 42s

COMPLETE REQUEST HEADERS Pretty length: 1 Bytes

Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.8,ca;q=0.6
Content-Type: text/plain
Origin: chrome-extension://aejoelaoggemb...
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW...

Podem observar en aquesta captura que ens retorna en el body l'identificador del fitxer XML. Comparant amb el missatge imprès per consola: podem considerar que `Num posts: 1 ID post: 0` la petició POST és correcte.

Missatge de Request del fitxer XML a partir de l'id per GET:



REQUEST

HTTP://localhost:8092/postureo?id=0 GET Send

Missatge de Response en la petició GET:



RESPONSE Elapsed Time: 324ms

200 OK

HEADERS pretty BODY pretty

Content-Length: 206 Bytes

Content-Type: application/json

Date: 2016 May 6 23:15:30

xml: {"<?xml version='1.0' encoding='UTF-8' standalone='yes"

COMPLETE REQUEST HEADERS Pretty lines nums length: 206 Bytes

Accept: */*

Accept-Encoding: gzip, deflate, sdch

Accept-Language: es-ES,es;q=0.8,ca;q=0.6

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW...

Podem observar en aquesta captura que obtenim de nou el mateix contingut XML enviat en la petició POST.

Big Web Service

Primer de tot hem creat un *Dynamic Web Project* i una classe POJO que n'implementa el Web Service. En aquest cas la classe es diu Weeeb i li hem afegit les següents etiquetes de JAX-WS per a que funcioni com a WebService:

- @WebService: Definició del servei Web.
- @WebMethod: Mètode del servei web.

En aquesta classe hi hem implementat el mètode que comprova que l'string que li arriba és correcte. Per fer-ho convertim aquesta cadena de caràcters a un objecte de tipus Capcelera utilitzant l'Unmarshalling del primer punt de la pràctica. Fent això ara podem obtenir els valors dels elements de l'objecte (ID, Data, NumOps, CtrlSum). Només cal comprovar que les longituds de cada propietat estigui sota un màxim establert pel document que se'ns ha facilitat a l'enunciat, Si és correcte el mètode retorna un String que diu "true", en cas contrari, "false".

Un cop hem creat el projecte i la classe hem aixecat el servei executant-lo al servidor Glassfish. Un cop fet això, testegem el Web Service a través dels enllaços de la següent imatge:

Web Service Endpoint Information

View details about a web service endpoint.

Application Name:	Punt3	
Tester:	/Punt3/WeeebService?Tester	Enllaç per poder provar el Big Web Service
WSDL:	/Punt3/WeeebService?wsdl	Enllaç on es localitza el fitxer WSDL del WeeebService
Endpoint Name:	Weeeb	
Service Name:	WeeebService	
Port Name:	WeeebPort	
Deployment Type:	109	
Implementation Type:	SERVLET	
Implementation Class Name:	sample.Weeeb	
Endpoint Address URI:	/Punt3/WeeebService	
Namespace:	http://sample/	

Aquí podem testejar el servei:

albetr:8090/Punt3/WeeebService?Tester

WeeebService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String sample.Weeeb.myMethod(java.lang.String) throws sample.JAXBException_Exception

myMethod (<?xml version="1.0" encoding="UTF-8" standalone="yes"?><GrpHdr MsgId="2"><CtrlSum>1324</CtrlSum></GrpHdr></div>

Introduïm l'xml , donem a myMethod i veurem els missatges SOAP de request i de response, junt amb el resultat "true" o "false".

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:MyMethod xmlns:ns2="http://sample/">
      <arg0><?xml version="1.0" encoding="UTF-8" standalone="yes"?><GrpHdr MsgId="2"><CtrlSum>1324</CtrlSum></GrpHdr></arg0>
    </ns2:MyMethod>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:MyMethodResponse xmlns:ns2="http://sample/">
      <return>true</return>
    </ns2:MyMethodResponse>
  </S:Body>
</S:Envelope>
```

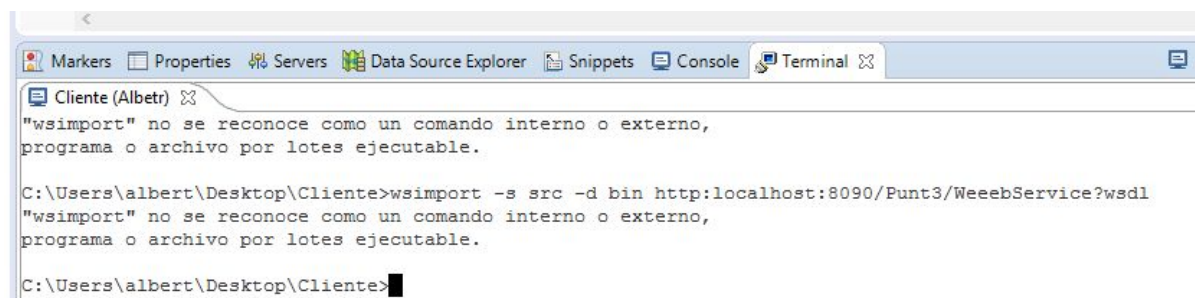
Method returned

java.lang.String : "true"

SOAP Request

Referència al resultat de la comprovació.

A la hora de fer el client no hem pogut avançar, ja que al volguer importar les classes del Server Endpoint no ens reconeixia la comanda “wsimport” encara que obríssim una terminal directament des d’eclipse. Hem posat com a paràmetre de URL la direcció del fitxer WSDL que ens ha donat la interfície de Gassfish (es pot veure a la imatge anterior). La següent imatge representa el problema amb la comanda “wsimport”:



Al no poder accedir al web service des del client, no hem pogut monitoritzar els missatges TCP/IP.

Conclusió

Aquesta pràctica ens ha ajudat a entendre millor la informació que se'ns ha mostrat a les classes de teoria, tenint així un punt de vista més ampli. Tot i haver passat per diferents problemes hem aconseguit solucionar-los i seguir endavant. Finalitzem aquesta pràctica amb una actitud molt positiva i amb ganes de seguir creixent en aquest àmbit.