

Aplicatii Integrate pentru Intreprinderi
Semestrul de Toamna 2013

Laborator 7
Construirea unui serviciu web folosind tehnologia JAX-WS

Continut

- Ce este un serviciu web ? Tipuri de servicii web și implementarea lor în Java
- Ce este JAX-WS ?
- Proiectarea unui serviciu web
- Utilizarea JAXB pentru conversia claselor Java în scheme XML
- Interogarea unui serviciu web prin intermediul unui client și folosind un server de aplicații

Ce este un serviciu web ?

- functionalitate implementata in cadrul unei aplicatii, accesibila prin intermediul unei retele de calculatoare (WWW), folosind de regula protocolul HTTP
- caracteristici
 - **interoperabilitate** intre aplicatii ce ruleaza pe platforme cu caracteristici diferite
 - **extensibilitate**
 - **descrieri ale functionalitatilor** in limbaje care pot fi procesate in mod automat
 - **asocierea** (slab cuplata) pentru a oferi functionalitati mai complexe – mesaje ce nu implica cunoasterea caracteristicilor de care dispun platformele pe care ruleaza

Tipuri de servicii web

- ● **servicii web de dimensiuni "mari"**
 - implementate in Java folosind JAX-WS
 - comunicatie client/server folosind mesaje XML care respecta standardul SOAP (Simple Object Access Protocol)
 - arhitectura si formatul mesajelor
 - definirea interfeței cu operatiile pe care le ofera serviciul web se face folosind WSDL (Web Service Description Language)
 - caracteristici
 - a) interfața pentru descrierea functionalitatilor implementate (eventual descrisa in WSDL) – mesaje, operatii, corespondente, locatia serviciului web
 - b) arhitectura trebuie sa corespunda unor cerinte complexe nonfunctionale (tranzactii, securitate, adresare, coordonare)
 - c) procesarea si invocarea functionalitatilor trebuie sa se faca asincron (standarde: WSRM; API-uri: JAX-WS)

Tipuri de servicii web (cont'd)

- ● **servicii web fara stare** (*eng.* RESTful web services)
 - scenarii de integrare ad-hoc
 - folosesc standarde W3C / IETF (HTTP – cu care sunt integrate mai bine decat SOAP, XML, URI, MIME)
 - infrastructura mai accesibila, costuri de dezvoltare mai scazute, mai putine restrictii in adoptare
 - caracteristici
 - a) performanta poate fi imbunatatita prin intermediul unei structuri de stocare
 - b) clientul si serverul au o intelegere comuna asupra contextului si continutul transmis in procesul de interactiune intre ele
 - c) latimea de banda este o resursa limitata – serviciile web fara stare sunt folosite pe dispozitive incorporate (PDA, mobile)
 - d) integrarea cu aplicatii Internet se realizeaza usor folosind AJAX sau JAX-RS sau DWR (Direct Web Remoting)

Tipuri de servicii web (cont'd)

- JAX-WS
 - folosit in aplicatiile integrate pentru intreprinderi
 - cerinte complexe, QoS (Quality of Service)
 - standarde pentru securitate si fiabilitate
 - interoperabilitate intre clienti si servere conforme cu protocoale pentru servicii web
- JAX-RS
 - servicii web conforme cu arhitectura REST
 - ● cuplare slaba
 - ● scalabilitate
 - ● arhitectura simpla
 - invocarea de servicii web concomitent cu dezvoltarea serverului

Ce este JAX-WS ?

- JAX-WS = Java API for XML Web Services (JSR 224)
- dezvoltarea de servicii web / aplicatii care le invoca
 - ● orientate pe mesaje
 - ● orientate pe apeluri la distanta (RPC – Remote Procedure Call)
- comunicare bazata pe XML
 - SOAP – structura antetelor, regulile de codificare, conventiile pentru reprezentarea invocarilor si raspunsurilor
 - WSDL – descrierea unui serviciu ca set de puncte din retea care opereaza asupra mesajelor
 - transparenta pentru programator, mesajele nu trebuie generate sau parsate caci sunt analizate automat de mediul de rulare JAX-WS
 - mesaje transmise prin HTTP

Ce este JAX-WS ? (cont'd)

- dezvoltarea aplicatiei JAX-WS
 - server: definirea operatiilor serviciului web precum si implementarea lor
 - client: crearea unui obiect delegat (*eng. proxy*) – obiect local reprezentand serviciul si invocarea metodelor la distanta
- avantajele JAX-WS
 - ● independenta de platforma a limbajului de programare Java
 - ● entitatile implicate in implementarea si invocarea serviciului web nu trebuie sa ruleze pe o platforma Java (folosesc tehnologii W3C: HTTP, SOAP, WSDL)

```

graph LR
    Client["Client  
JAX-WS Runtime"] -- "SOAP Message" --> WebService["Web Service  
JAX-WS Runtime"]
            
```

Proiectarea unui serviciu web

- serviciu web dezvoltat cu JAX-WS = clasa Java (SEI) adnotata cu `javax.jws.WebService`
 - SEI = Service Endpoint Interface / Implementation
 - `@WebService` – defineste clasa ca entitatea care deserveste serviciul web
 - interfața sau clasa unde sunt declarate metodele pe care un client le poate invoca pentru un serviciu
 - !!! nu este necesara definirea explicita a unei interfete (ca in cazul RMI/CORBA)
 - o interfața poate fi definita explicit – daca se dorește – folosind elementul `endpointInterface` din adnotarea `WebService`

Condițiile ce trebuie îndeplinite de structura care definește serviciul web

- **1** adnotată cu una din însemnarile `javax.jws.WebService` / `javax.jws.ServiceProvider`
- **2** definirea interfeței se poate face explicit folosind elementul `endpointInterface` al adnotării `WebService` sau implicit, în absența acestuia
- **3** metodele care definesc funcționalitatea serviciului web trebuie să fie publice și să nu fie declarate `static` / `final`
- **4** metodele ce implementează funcționalitatea serviciului web trebuie să fie adnotate cu însemnarea `javax.jws.WebMethod`
- **5** parametri și rezultatele întoarse ale metodelor apelabile de utilizatorii serviciului web trebuie să fie tipuri compatibile JAXB

Condițiile ce trebuie îndeplinite de structura care definește serviciul web (cont'd)

- **6** clasa ce implementează serviciul web nu trebuie declarată `final` și nu trebuie să fie abstractă
- **7** clasa care implementează serviciul web trebuie să definească un constructor public implicit
- **8** clasa care implementează serviciul web nu trebuie să definească metoda `finalize`
- **9** clasa care implementează serviciul web poate defini metode adnotate cu însemnarile `javax.annotation.PostConstruct` / `javax.annotation.PreDestroy` pentru evenimente legate de ciclul de viață al serviciului web

- **@PostConstruct** – metoda invocată înainte ca serviciul web să accepte invocări de la clienți
- **@PreDestroy** – metoda invocată înainte ca serviciul web să își înceteze activitatea

Proiectarea unui serviciu web (cont'd) - Exemplu

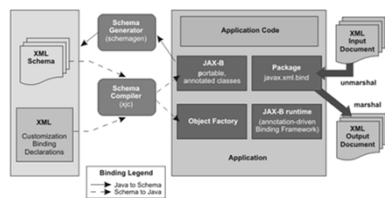
```
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService(serviceName = "Reservation")
public class Reservation {
    public int numberOfSeats;
    public PersistentInterval timetable;
    public PersistentReservationData reservationData;
    public Reservation() {}
    @WebMethod(operationName = "getTimeTable")
    public ArrayList<Interval> getTimeTable() {
        // ...
    }
    @WebMethod(operationName = "getAvailableSeats")
    public int getAvailableSeats(
        @WebParam(name = "interval") Interval interval) {
        // ...
    }
    @WebMethod(operationName = "makeReservation")
    public boolean makeReservation(
        @WebParam(name = "customerID") int customerID,
        @WebParam(name = "interval") Interval interval,
        @WebParam(name = "numberOfSeatsRequired") int numberOfSeatsRequired) {
        // ...
    }
    @WebMethod(operationName = "cancelReservation")
    public boolean cancelReservation(
        @WebParam(name = "customerID") int customerID,
        @WebParam(name = "interval") Interval interval) {
        // ...
    }
}
```

Proiectarea unui serviciu web (cont'd) – Exemflu (2)

- Reservation – clasa care implementeaza serviciul web
 - adnotata cu `@WebService(serviceName="Reservation")`
- definirea unui constructor implicit – apelat in momentul in care
- metodele accesibile clientilor
 - `getTimetable, getAvailableSeats, makeReservation, cancelReservation`
 - adnotarea `@WebMethod(operationName="**")` le face vizibile catre clienti
 - parametrii metodelor trebuie sa fie adnotate cu `@WebParam(name="**")`, specificandu-se un nume sugestiv pentru semnificatia argumentelor

Utilizarea JAXB pentru conversia claselor Java in scheme XML

- JAXB – Java Architecture for XML Binding
- maparea tipurilor de date din limbajul de programare Java in definitii XML se face in mod transparent (pentru programator) prin intermediul JAXB



Utilizarea JAXB pentru conversia claselor Java in scheme XML (cont'd)

- utilitare
 - `xjc` – compilarea unei scheme XML intr-o clasa Java
 - `schemagen` – generarea unei scheme XML dintr-o clasa Java
 - cadru general de rulare
- transformarea intre definitii ale schemelor XML (XSD) si clase Java Beans
- adnotarile contin toate informatiile necesare pentru realizarea conversiei dintr-un format intr-altul, utila pentru **transmisia prin infrastructura de comunicatie**
 - impachetare: obiecte Java → documente XML
 - despachetare: documente XML → obiecte Java
- conversia datelor in cadrul serviciilor web folosind JAX-WS, dar si independent de aceasta

Utilizarea JAXB pentru conversia claselor Java in scheme XML (cont'd)

- utilitatea JAXB
 - ● transportul parametrilor si valorilor intoarse ale metodelor serviciului web prin infrastructura de comunicatie
 - ● asigurarea unui mecanism de persistenta, alternativa la JPA, in cazul in care nu se folosesc baze de date relationale
- nu toate clasele Java au corespondenti intr-o schema XML
- pentru clase Java definite de utilizator, se folosesc adnotari care specifica mecanismul de conversie intre aceste tipuri de date

Conversia scheme XML – limbajul Java

tip schemă XML	tip Java
xsd:string	java.lang.String
xsd:integer	java.math.BigInteger
xsd:int	int
xsd:long	long
xsd:short	short
xsd:decimal	java.math.BigDecimal
xsd:float	float
xsd:double	double
xsd:boolean	boolean
xsd:byte	byte
xsd:QName	javax.xml.namespace.QName
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]
xsd:unsignedInt	long
xsd:unsignedShort	int
xsd:unsignedByte	short
xsd:time	javax.xml.datatype.XMLGregorianCalendar
xsd:date	javax.xml.datatype.XMLGregorianCalendar
xsd:g	javax.xml.datatype.XMLGregorianCalendar
xsd:anySimpleType	java.lang.Object
xsd:anySimpleType	java.lang.String
xsd:duration	javax.xml.datatype.Duration
xsd:NOTATION	javax.xml.namespace.QName

Conversia limbajul Java – scheme XML

tip Java	tip schemă XML
java.lang.String	xs:string
java.math.BigInteger	xs:integer
java.math.BigDecimal	xs:decimal
java.util.Calendar	xs:dateTime
java.util.Date	xs:dateTime
javax.xml.namespace.QName	xs:QName
java.net.URI xs:string	xs:string
javax.xml.datatype.XMLGregorianCalendar	xs:anySimpleType
javax.xml.datatype.Duration	xs:duration
java.lang.Object	xs:anyType
java.awt.Image	xs:base64Binary
javax.activation.DataHandler	xs:base64Binary
javax.xml.transform.Source	xs:base64Binary
java.util.UUID	xs:string

Adnotari JAXB

- `@XmlRootElement(namespace="...")` – definește elementul rădăcina din cadrul schemei XML
- `@XmlElement` – convertește între o proprietate JavaBeans / câmp al unei clase (prin intermediul metodei getter) și un atribut XML
- `@XmlAccessorType(XmlAccessType.FIELD)` – toate atributele clasei vor fi serializabile
 - nu mai este necesară implementarea interfeței `java.io.Serializable`
 - serializarea se face prin intermediul unor documente XML
- `@XmlType(name="...")`
 - definește un nume pentru tipul de date respective
 - descrierea în limbaj WSDL va cuprinde definițiile metodelor din clasa având tipul respective, fiind cuprinse în clasele generate la nivelul clientului

Adnotari JAXB (cont'd)

Adnotare	Descriere
<code>@XmlRootElement(namespace="...")</code>	definește elementul rădăcină din cadrul schemei XML
<code>@XmlElement</code>	face conversia între o proprietate JavaBeans sau un câmp al unei clase și un atribut XML
<code>@XmlAccessorType(XmlAccessType.PACKAGE FIELD)</code>	este aplicat unui pachet sau unei clase Java, specificând dacă proprietățile JavaBeans sau atributele vor fi serializate
<code>@XmlType(propOrder={"atn1","atn2",...,"atnN"})</code>	permite stabilirea structurii documentului XML prin indicarea ordinii în care atributele clasei vor fi copiate

Adnotari JAXB (cont'd)

Exemplu (1)

```

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlType;
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="Interval")
public class Interval {
    // ...
}
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="ReservationData")
public class ReservationData {
    // ...
}
    
```

- toate atributele clasei vor fi serializate
- atributele au tipul `GregorianCalendar`, serializabil și având corespondența într-o schemă XML
- tipul de date este `Interval / ReservationData`

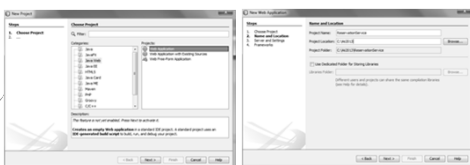
Incarcarea si Descarcarea obiectelor persistente (cont'd)

```
public final PersistentInterval unpackPersistentInterval(String file) {
    try {
        Umarsaller conversion =
            context.PersistentInterval.createUmarsaller();
        return (PersistentInterval)
            conversion.unmarshal(new FileInputStream(file));
    } catch (Exception exception) { }
    return null;
}
public final PersistentReservationData unpackPersistentReservationData
    (String file) {
    try {
        Umarsaller conversion =
            context.PersistentReservationData.createUmarsaller();
        return (PersistentReservationData)
            conversion.unmarshal(new FileInputStream(file));
    } catch (Exception exception) { }
    return null;
}
```

Incarcarea si Descarcarea obiectelor persistente (cont'd)


- contextul JAXB referitor la persistenta va fi incarcat cu tipul de date pentru care se doreste conversia
- starea obiectelor trebuie scrisa / citita in cazul fiecarui obiect persistent, atunci cand este accesat de metode accesibile la distanta
- metode
 - **marshal** – scrie un obiect compatibil JAXB intr-un document XML potrivit mecanismului de conversie definit
 - **unmarshal** – citeste un obiect compatibil JAXB dintr-un document XML potrivit mecanismului de conversie definit
- aplicate pe obiecte de tip Marshaller / Unmarshaller corespunzatoare obtinute din contextul JAXB referitor la persistenta asociate tipului de date respectiv

Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0




- ● se creeaza un proiect JavaWeb → Web Application
- ● se indica numele
- ● se alege tipul de server de aplicatii si versiunea Java EE
- ● se specifica si alte tehnologii cu care este integrat serverul (Spring Web MVC, Struts, Java Server Faces, Hibernate)

Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)



- ▶ se adauga o resursa de tipul Web Service la proiect
- ▶ se indica
 - ▶ numele serviciului web
 - ▶ pachetul din care acesta va face parte
- ▶ serviciul web
 - ▶ poate fi creat de la zero
 - ▶ poate fi generat dintr-o clasa Session Bean


Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)



Modulul *Design* permite gestiunea operatiilor pe care serviciul web le pune la dispozitia utilizatorilor in mod vizual (adaugare, modificare, stergere) precum si atribute ce tin de calitatea serviciului:

- ▶ optimizarea transferului datelor binare
- ▶ fiabilitatea transmiterii mesajelor
- ▶ securitate

Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)



- ▶ lansarea in executie a serviciului web se face prin optiunea *Deploy*
 - ▶ modificarile realizate asupra implementarilor metodelor oferite de serviciul web sunt vizibile la nivelul clientului atunci cand sunt salvate sursele (nu mai este necesara realizarea operatiei *Deploy*)
 - ▶ este pornit in mod automat serverul de aplicatii GlassFish 4.0

Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 și serverul de aplicații GlassFish 4.0 (cont'd)



- consola de serverul de aplicații GlassFish poate fi accesata la <http://localhost:4848>
 - user: admin
 - parola: generata in mod automat (poate fi vizualizata in NetBeans: Tools → Servers → Glassfish Server 4.0, tab-ul Common)
- serviciile web dezvoltate pot fi consultate in sectiunea Applications
- descrierea serviciului web (WSDL): legatura [View Endpoint](#)

Descrierea WSDL a serviciului web

```
<?xml version='1.0' encoding='UTF-8' ?>
<definitions target="http://reservation/" name="Reservation"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://reservation/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soap12="http://schemas.xmlsoap.org/soap12/envelope/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl12="http://schemas.xmlsoap.org/wsdl/12/">
  <types>
    <xs:element base="xsd:string" name="Reservation" />
  </types>
  <binding name="ReservationBinding" type="tns:Reservation"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    style="rpc"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl12="http://schemas.xmlsoap.org/wsdl/12/">
    <operation name="getReservation"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdl12="http://schemas.xmlsoap.org/wsdl/12/">
      <input wsdl:use="literal" />
      <output wsdl:use="literal" />
    </operation>
  </binding>
  <service name="Reservation"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl12="http://schemas.xmlsoap.org/wsdl/12/">
    <port name="ReservationPort"
      binding="tns:ReservationBinding"
      location="http://localhost:8080/ReservationService/Reservation"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdl12="http://schemas.xmlsoap.org/wsdl/12/" />
  </service>
</definitions>
```

Interogarea unui serviciu web prin intermediul unui client

- generarea claselor pe client care permit invocarea serviciului web
 - folosind utilitarele (ce vin impreuna cu Java)
 - wagen – generarea unui serviciu web
 - wsimport – importarea unui serviciu web (spre a fi invocate)
 - <ServiceName>.Service – obtinerea unui port (obiect delegat al serviciului web)
 - pentru fiecare metoda operation oferita de serviciul web sunt generate
 - Operation
 - OperationResponse
 - clasele corespunzatoare parametrilor si valorilor intoarse de metodele la distanta

Interogarea unui serviciu web prin intermediul unui client

- folosirea adnotării `javax.xml.ws.WebServiceRef` spre a indica locatia la care se gaseste descrierea serviciului web (prin elementul `wsdl:Location`)
- obtinerea elementului delegat (port) pe care pot fi invocate operatiile la distanta
- invocarea operatiilor puse la dispozitie de serviciul web, ca metode ale obiectului delegat

```
public class Client {
    @WebServiceRef(wsdlLocation =
        "META-INF/wsdl/localhost_8080/
        reservationService/ReservationService.wsdl")
    private static reservation.Reservation_Service service;
    public static void main(String[] args) {
        service = new reservation.Reservation_Service();
        reservation.Reservation port = service.getReservationPort();
        List<Interval> timetable = port.getTimeTable();
    }
}
```

Studiu de Caz: Dezvoltarea unui client ce utilizeaza un serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)



- se creeaza un proiect de tip Java Application la care se adauga un client pentru serviciul web (Web Service Client)
- se indica locatia de unde poate fi descarcata descrierea (in format WSDL) serviciului web pentru generarea claselor (se gasesc in Generated Sources)
- in cazul modificarii specificatiilor serviciului web regenerarea claselor se face cu *Refresh*

Studiu de Caz: Dezvoltarea unui client ce utilizeaza un serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)

Reservation Web Service Tester



- testarea functionalitatii serviciului web se poate face in mod vizual, daca nu sunt implicate referinte catre obiecte
- pot fi vizualizate mesaje SOAP interschimbate intre client si server

```
<?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:getTimeTable xmlns:ns2="http://reservation/" />
  </S:Body>
</S:Envelope>
<?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:getTimeTableResponse xmlns:ns2="http://reservation/" />
  </S:Body>
</S:Envelope>
```
