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 şi 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 şi 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
  - ❶ **interoperabilitate** intre aplicatii ce ruleaza pe platforme cu caracteristici diferite
  - ❷ **extensibilitate**
  - ❸ **descrieri ale functionalitatilor** in limbaje care pot fi **procesate in mod automat**
  - ❹ **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

- **❶ 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 interfetei cu operatiile pe care le ofera serviciul web se face folosind WSDL (Web Service Description Language)
  - caracteristici
    - a) interfata 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)

- ❷ **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)

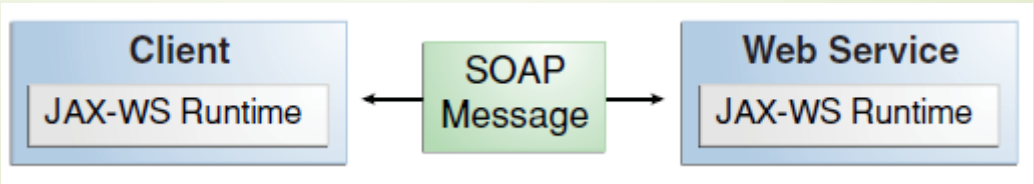
- JAX-WS
  - folosit in aplicatiile integrate pentru intreprinderi
  - cerinte complexe, QoS (Quality of Service)
  - standarde pentru securitate si fiabilitate
  - interoperabilitate intre clienti si servere conforme cu protocoale pentru servicii web
- JAX-RS
  - servicii web conforme cu arhitectura REST
    - ❶ cuplare slaba
    - ❷ scalabilitate
    - ❸ 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
  - ❶ orientate pe mesaje
  - ❷ orientate pe apeluri la distanta (RPC – Remote Procedure Call)
- comunicatie 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
  - ❶ independenta de platforma a limbajului de programare Java
  - ❷ entitatile implicate in implementarea si invocarea serviciului web nu trebuie sa ruleze pe o platforma Java (folosesc tehnologii W3C: HTTP, SOAP, WSDL)



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
  - interfata 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 interfata poate fi definita explicit – daca se doreste – folosind elementul `endpointInterface` din adnotarea `WebService`

Conditiile ce trebuie indeplinite de structura care defineste serviciul web

- ❶ adnotata cu una din insemnarile `javax.jws.WebService` / `javax.jws.ServiceProvider`
- ❷ definirea interfetei se poate face explicit folosind elementul `endpointInterface` al adnotarii `WebService` sau implicit, in absenta acestuia
- ❸ metodele care definesc functionalitatea serviciului web trebuie sa fie publice si sa nu fie declarate `static` / `final`
- ❹ metodele ce implementeaza functionalitatea serviciului web trebuie sa fie adnotate cu insemnarea `javax.jws.WebMethod`
- ❺ parametrii si rezultatele intoarse ale metodelor apelabile de utilizatorii serviciului web trebuie sa fie tipuri compatibile JAXB

Conditiile ce trebuie indeplinite de structura care defineste serviciul web (cont'd)

- ❻ clasa ce implementeaza serviciul web nu trebuie declarata `final` si nu trebuie sa fie abstracta
- ❼ clasa care implementeaza serviciul web trebuie sa defineasca un constructor public implicit
- ❽ clasa care implementeaza serviciul web nu trebuie sa defineasca metoda finalize
- ❾ clasa care implementeaza serviciul web poate defini metode adnotate cu insemnarile `javax.annotation.PostConstruct` / `javax.annotation.PreDestroy` pentru evenimente legate de ciclul de viata al serviciului web
  - `@PostConstruct` – metoda invocata inainte ca serviciul web sa accepte invocari de la clienti
  - `@PreDestroy` – metoda invocate inainte ca serviciul web sa isi inceteze activitatea

Proiectarea unui serviciu web (cont'd) - Exemplu

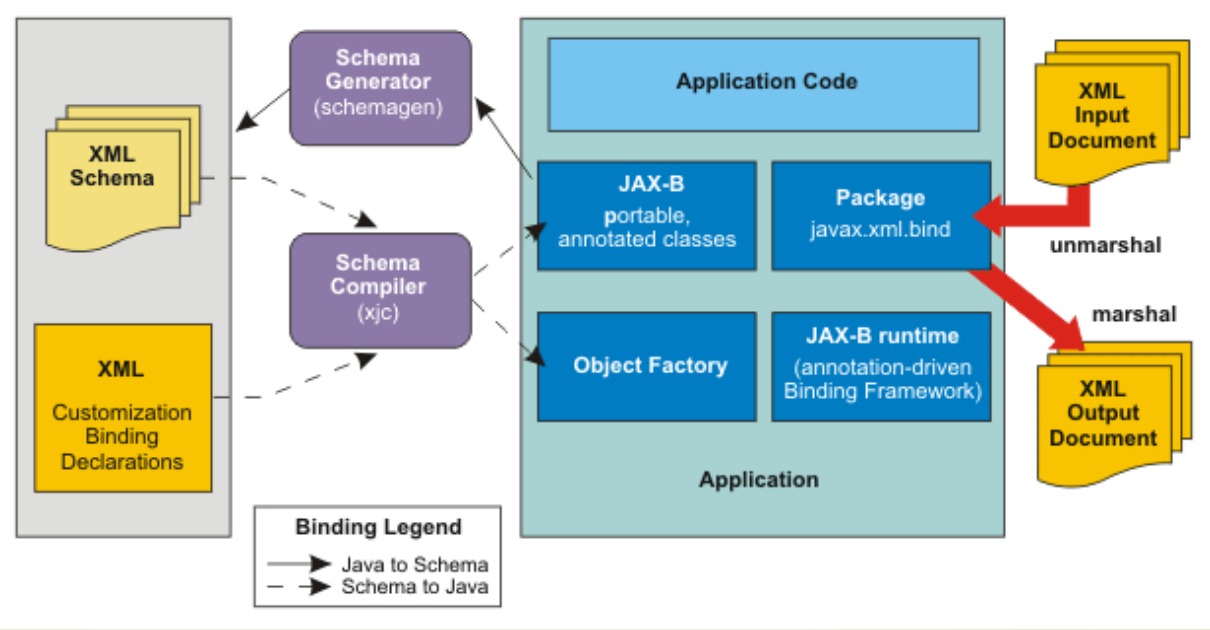
```
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService(serviceName = "Reservation")
public class Reservation {
    public int                       numberOfSeats;
    public PersistentInterval        timetable;
    public PersistentReservationData reservationData;
    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 = "noOfSeatsRequired") int noOfSeatsRequired) {
        // ...
    }
    @WebMethod(operationName = "cancelReservation")
    public boolean cancelReservation(
           @WebParam(name = "customerId") int customerId,
           @WebParam(name = "interval") Interval interval) {
        // ...
    }
}
```

Proiectarea unui serviciu web (cont'd) – Exemplu (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
  - ❶ transportul parametrilor si valorilor intoarse ale metodelor serviciului web prin infrastructura de comunicatie
  - ❷ asigurarea unui mecanism de persistenta, alternativa la JPA, in cazul in care nu se folosesc baze de date relationale
- nu toate clasele Java au corespondent 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="…")` – defineste elementul radacina din cadrul schemei XML
- `@XmlElement` – conversie intre o proprietate JavaBeans / camp al unei clase (prin intermediul metodei getter) si un atribut XML
- `@XmlAccesorType(XmlAccesType.{PACKAGE|FIELD})` – toate atributele clasei vor fi serializabile
  - nu mai este necesara implementarea interfetei `java.io.Serializable`
  - serializarea se face prin intermediul unor documente XML
- `@XmlType(name="…")`
  - defineste un nume pentru tipul de date respective
  - descrierea in limbaj WSDL va cuprinde definitiile metodelor din clasa avand tipul respective, fiind cuprinse in clasele generate la nivelul clientului

Adnotari JAXB (cont'd)

Adnotare	Descriere
@XmlRootElement(namespace="…")	defineşte elementul rădăcină din cadrul schemei XML
@XmlElement	face conversia între o proprietate JavaBeans sau un câmp al unei clase şi un atribut XML
@XmlAccesorType(XMLAccessType.PACKAGE|FIELD)	este aplicat unui pachet sau unei clase Java, specificând daca proprietățile JavaBeans sau atributele vor fi serializate
@XmlType(propOrder={"atr1","atr2",..., "atrn" })	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 si avand corespondenta intr-o schema XML
- tipul de date este `Interval` / `ReservationData`

Adnotari JAXB (cont'd) Exemplu (2)

```
@XmlRootElement
public class PersistentInterval {
    public ArrayList<GregorianCalendar>
        startingIntervals;
    public ArrayList<GregorianCalendar>
        endingIntervals;
    @XmlElement
    public ArrayList<GregorianCalendar>
        getStartingIntervals() {
        return startingIntervals;
    }
    @XmlElement
    public ArrayList<GregorianCalendar>
        getEndingIntervals() {
        return endingIntervals;
    }
}
```

```
@XmlRootElement
public class PersistentReservationData {
    private ArrayList<Integer> customerIds;
    private ArrayList<GregorianCalendar>
        startingIntervals;
    private ArrayList<GregorianCalendar>
        endingIntervals;
    private ArrayList<Integer> numberOfSeats;
    @XmlElement
    public ArrayList<Integer> getCustomerIds() {
        return customerIds;
    }
    @XmlElement
    public ArrayList<GregorianCalendar>
        getStartingIntervals() {
        return startingIntervals;
    }
    @XmlElement
    public ArrayList<GregorianCalendar>
        getEndingIntervals() {
        return endingIntervals;
    }
    @XmlElement
    public ArrayList<Integer> getNumberOfSeats() {
        return numberOfSeats;
    }
}
```

Incarcarea si Descarcarea obiectelor persistente

```
JAXBContext contextPersistentInterval;
JAXBContext contextPersistentReservationData;
public Reservation() {
    try {
        contextPersistentInterval =
        JAXBContext.newInstance(PersistentInterval.class);
        contextPersistentReservationData =
        JAXBContext.newInstance(PersistentReservationData.class);
    }
    catch (Exception exception) {
        System.out.println ("exceptie: "+exception.getMessage());
    }
    if (new File("timetable.xml").exists()) {
        timetable = unpackPersistentInterval("timetable.xml");
    } else {
        timetable = new PersistentInterval();
        packPersistentInterval(timetable,"timetable.xml");
    }
    if (new File("reservationData.xml").exists()) {
        reservationData =
        unpackPersistentReservationData("reservationData.xml");
    } else {
        reservationData = new PersistentReservationData();
        packPersistentReservationData
                    (reservationData,"reservationData.xml");
        }
    }
```

Incarcarea si Descarcarea obiectelor persistente (cont'd)

```
public final void packPersistentInterval
                  (PersistentInterval object, String file) {
        try {
            Marshaller conversion =
            contextPersistentInterval.createMarshaller();
            conversion.marshal(object, new FileOutputStream(file));
        }
        catch (Exception exception) { }
}
public final void packPersistentReservationData
                  (PersistentReservationData object, String file) {
        try {
            Marshaller conversion =
            contextPersistentReservationData.createMarshaller();
            conversion.marshal(object, new FileOutputStream(file));
        }
        catch (Exception exception) { }
}
```

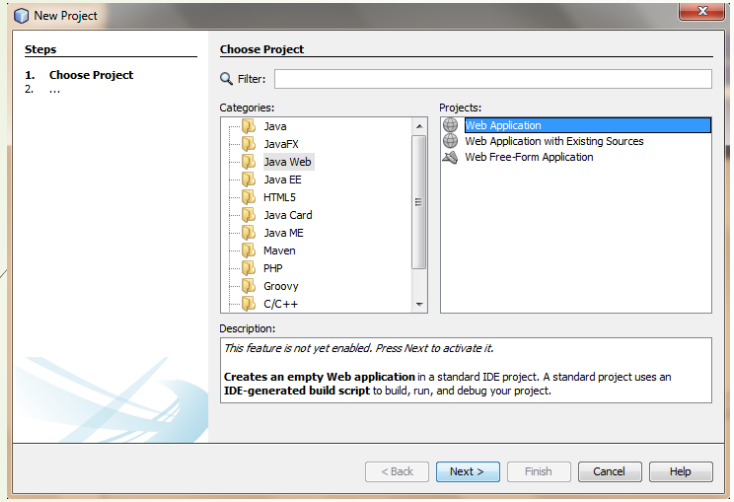
Incarcarea si Descarcarea obiectelor persistente (cont'd)

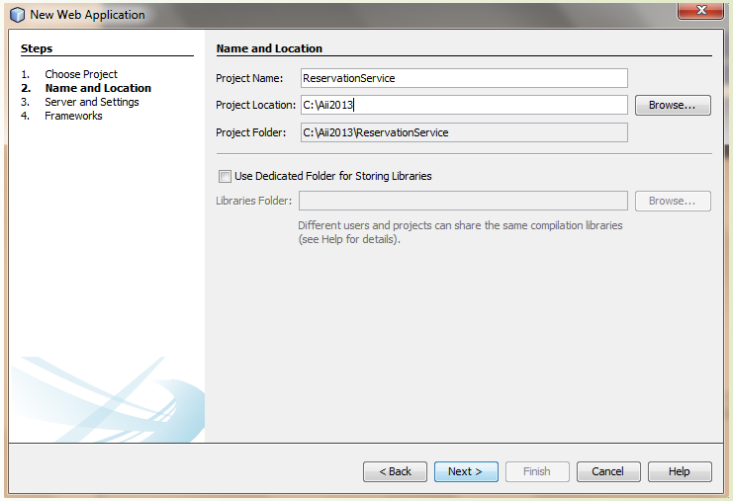
```
 public final PersistentInterval unpackPersistentInterval(String file) {
        try {
            Unmarshaller conversion =
            contextPersistentInterval.createUnmarshaller();
            return (PersistentInterval)
                   conversion.unmarshal(new FileInputStream(file));
        }
        catch (Exception exception) { }
        return null;
}
public final PersistentReservationData unpackPersistentReservationData
                                        (String file) {
        try {
            Unmarshaller conversion =
            contextPersistentReservationData.createUnmarshaller();
            return (PersistentReservationData)
                   conversion.unmarshal(new FileInputStream(file));
        }
        catch (Exception exception) { }
        return null;
}
```

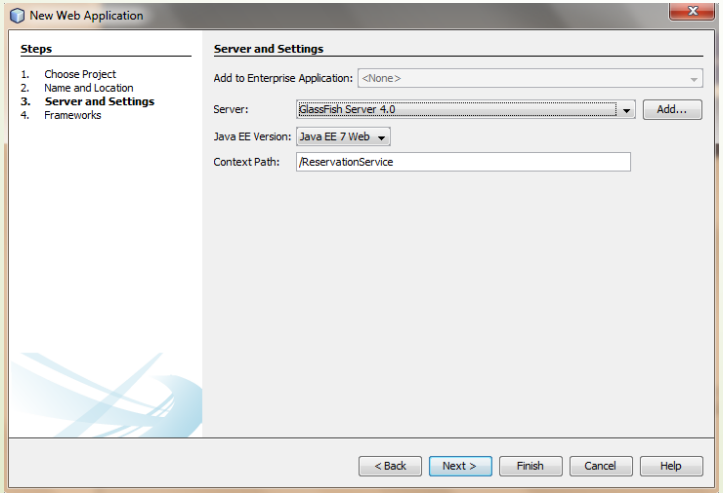
Incarcarea si Descarcarea obiectelor persistente (cont'd)

- 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

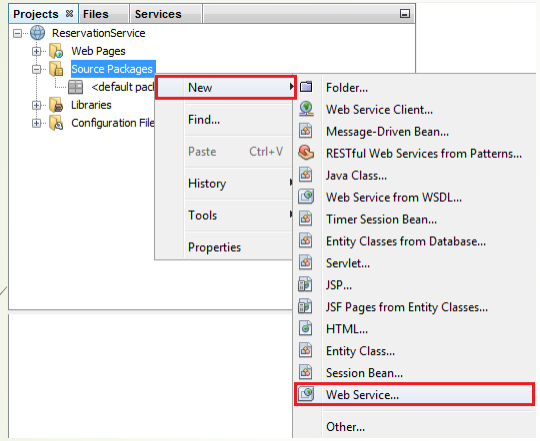


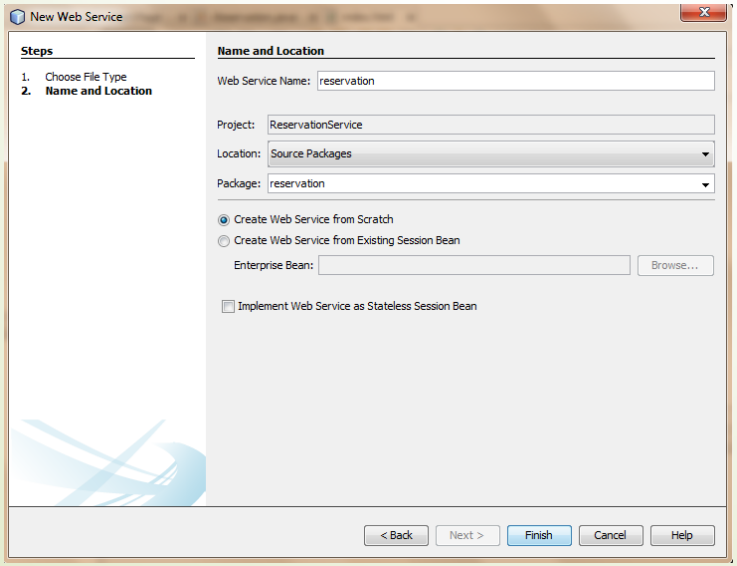




- ❶ se creeaza un proiect JavaWeb → Web Application
- ❷ se indica numele
- ❸ se allege tipul de server de aplicatii si versiunea Java EE
- ❹ se specifica si alte tehnologii cu care este integrat serverul (Spring Web MVC, Struts, Java Server Faces, Hibernate)

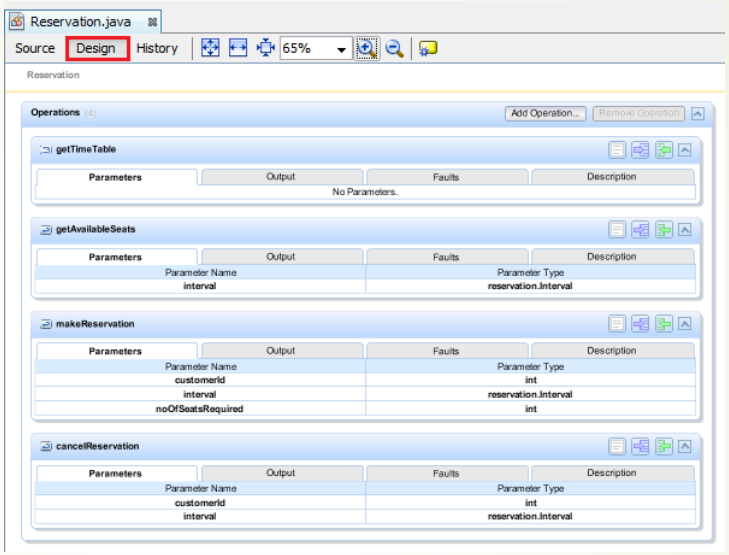
Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)





- 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

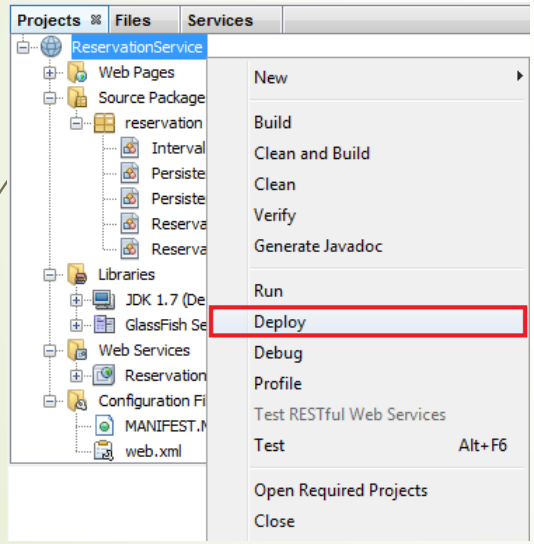
Studiu de Caz: Dezvoltarea unui serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)



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:

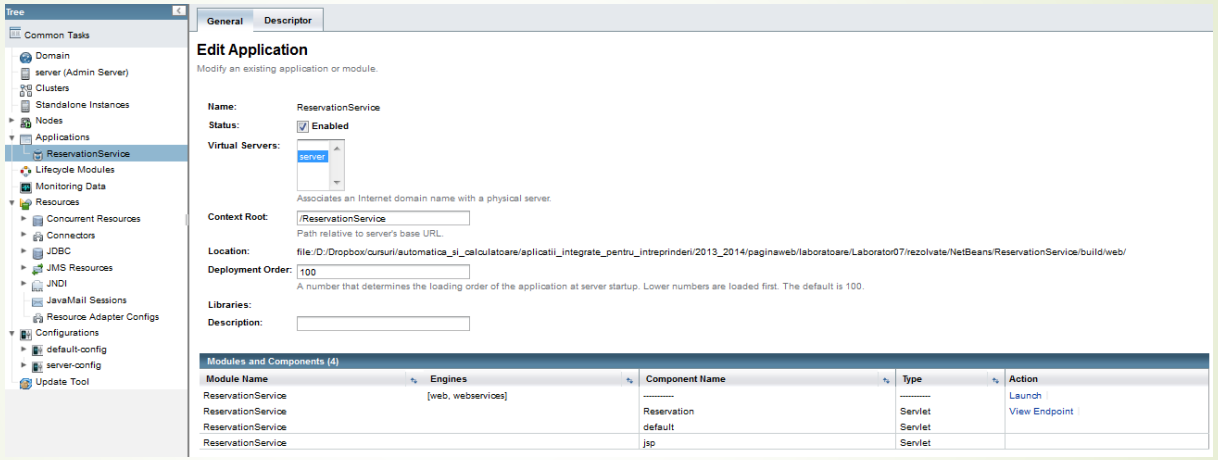
- optimizarea transferului datelor binare
- fiabilitatea transmiterii mesajelor
- securitate

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 si serverul de aplicatii GlassFish 4.0 (cont'd)



- consola de serverului de aplicatii 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

```
<definitions targetNamespace="http://reservation/" name="Reservation">
<types>
    <xsd:schema><xsd:import namespace="http://reservation/"
    schemaLocation="http://localhost:8080/ReservationService/Reservation?xsd=1"/>
    </xsd:schema>
</types>
<message name="getTimeTable">
    <part name="parameters" element="tns:getTimeTable"/>
</message>
<message name="getTimeTableResponse">
    <part name="parameters" element="tns:getTimeTableResponse"/>
</message>
<portType name="Reservation">
    <operation name="getTimeTable">
        <input wsam:Action="http://reservation/Reservation/getTimeTableRequest"
        message="tns:getTimeTable"/>
        <output wsam:Action="http://reservation/Reservation/getTimeTableResponse"
        message="tns:getTimeTableResponse"/>
    </operation>
</portType>
<binding name="ReservationPortBinding" type="tns:Reservation">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
    <operation name="getTimeTable">
        <soap:operation soapAction=""/>
        <input><soap:body use="literal"/></input>
        <output><soap:body use="literal"/></output>
    </operation>
</binding>
<service name="Reservation">
    <port name="ReservationPort" binding="tns:ReservationPortBinding">
        <soap:address
        location="http://localhost:8080/ReservationService/Reservation"/>
    </port>
</service>
</definitions>
```

Interogarea unui serviciu web prin intermediul unui client

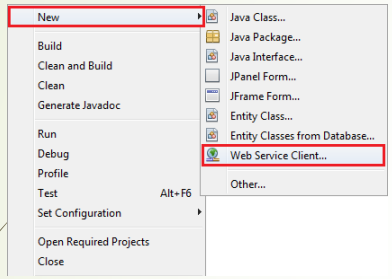
- generarea claselor pe client care permit invocarea serviciului web
  - folosind utilitarele (ce vin impreuna cu Java)
    - `wsgen` – generarea unui serviciu web
    - `wsimport` – importarea unui serviciu web (spre a fi invocate)
  - `<ServiceName>_Service` – obtinerea unui port (obiect delegat al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

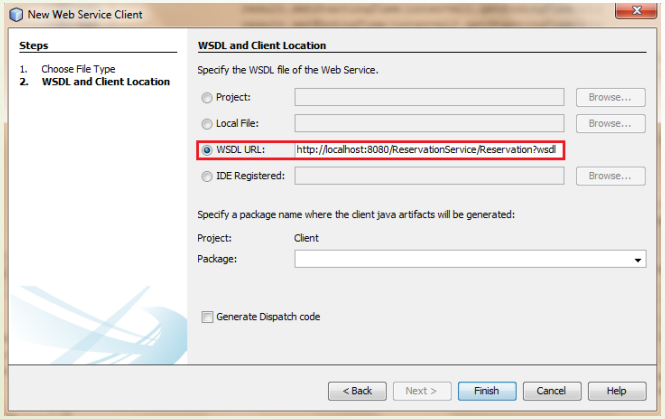
Interogarea unui serviciu web prin intermediul unui client

- folosirea adnotarii `javax.xml.ws.WebServiceRef` spre a indica locatia la care se gaseste descrierea serviciului web (prin elementul `wsdlLocation`)
- 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();
    }
}
```

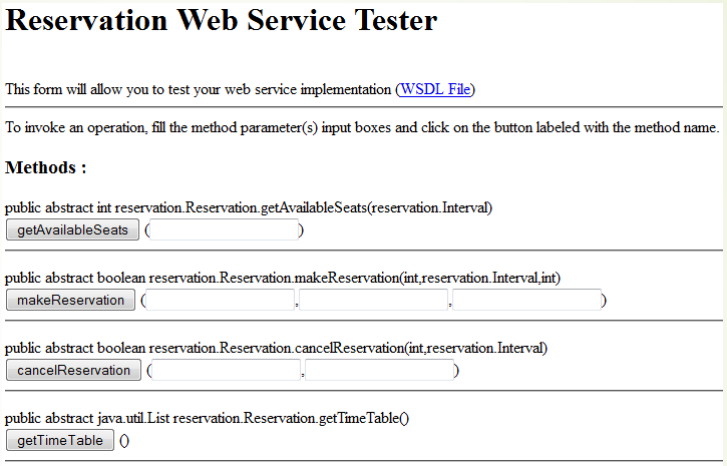
Studiu de Caz: Dezvoltarea unui client ce utilizeaza un serviciu web folosind NetBeans 7.4 si serverul de aplicatii GlassFish 4.0 (cont'd)





- 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)



- testarea functionalitatii serviciului web se poate face in mod vizual, daca nu sunt implicate referinte catre obiecte
- pot fi vizualizate mesajele 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>
```