Programski Sistemi

Web servisi i Java: JAX-WS

Šta su Web servisi?

- Web servisi su programski sistemi projektovani da podrže interoperabilnu komunikaciju između mašina preko mreže
- Interoperabilnost znači da softverski sistemi nezavisno od tehnologije u kojoj su napisani mogu da komuniciraju i razmenjuju podatke
- W3C Web Services Architecture Group definicija:
  - "Softverski sistem identifikovan uz pomoc URI, definisan, opisan i otkriven pomocu XML artefakta, koji interaguje sa drugim softverskim sistemima koristeci XML poruke preko Internet protokola"

Zašto Web servisi?

- Uspešne implementacije Web servisa obezbedila su rešenja za brojne probleme
- Transport
  - Potreba za procenjivanjem realnih cena je otežana kada pošiljku prenosi više prevoznika
- Bankarstvo
  - Potreba za obezbedjivanjem bankarskih usluga većem broju korisnika
  - Potreba za integrisanjem velikog broja internih aplikacija
- Državna uprava
  - Više resora moraju da dele podatke

Java i Web servisi

- XML je usvojen kao format za razmenu podataka
- Pošto se XML i Java odlično slažu, zauzeli su centralno mesto u razvoju Web servisa
- Java API-ji za XML i platforma J2EE su idealna kombinacija za razvoj i objavljivanje Web servisa
- Pošto se Web servisi najčešće koriste preko Weba, kao protokol za razmenu XML poruka se koristi HTTP, ali mogu da se koriste i drugi Internet protokoli: HTTPS, TCP/IP, SMTP, FTP i sl.

AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

WSDL

- Web servisi se objavljuju i pokreću na serveru
- Klijent mora da zna koje usluge servis nudi, ali ne mora da poznaje detalje implementacije (npr. programski jezik u kome je napisan servis ili potpise metoda)
- Za opis usluga koristi se WSDL (Web Service Description Language)
- Postoji mogućnost da se iz WSDL kreiraju klase kako za serversku, tako i za klijentsku stranu

UNIVERZITET SINGIDUNUM AKREDITOVANI PRIVATNI UNIVERZITET

Osnovni koraci za Web servise

1. Programiranje Web servisa
2. Objavljivanje Web servisa (WSDL)
3. Pronalaženje Web servisa (UDDI)
4. Povezivanje klijenta sa servisom (SOAP)

UNIVERZITET PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

Web Services Description Language (WSDL)

- WSDL opis je XML koji pruža sve informacije o Web servisu, npr.
  - Ime servisa
  - Operacije koje izvršava
  - Parametre za te operacije
  - Mesto gde treba slati zahteve
- Klijent Web servisa koristi WSDL dokument da bi otkrio šta servis nudi i kako da mu pristupi

PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

Simple Object Access Protocol (SOAP)

- SOAP (Simple Object Access Protocol) je standardni komunikacioni protokol
- SOAP definiše standarde za razmenu XML poruka i preslikavanje tipova podataka tako da aplikacije koje primenjuju te standarde mogu međusobno da komuniciraju
- SOAP poruka je sama po sebi XML dokument i obično se zove SOAP koverta (envelope)
- Nezavisan od vrste mreže, transporta i programskog jezika

PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

UDDI

- Web servis može da registruje svoj interfejs u registru (Universal Description Discovery, and Integration, ili UDDI) da bi klijenti mogli da ga pronađu
- UDDI je XML registar i pokazuje na javni WSDL fajl servisa koji se nalazi na Internetu, a koji potencijalni klijenti mogu da preuzmu
- Kada klijent sazna interfejs servisa i format poruka, može da pošalje zahtev za servisom i da na njega dobije odgovor

Java API for XML Based Web Services (JAX-WS)

- Java API za Web servise (uključen u Java 6), trenutno u verziji 2.2
- Koristi anotacije za definisanje Web servisa (paket **javax.xml.ws**)
- Za protokol u komunikaciji koristi se SOAP
- Definiše skup API-ja i anotacija koje maskiraju složenost SOAP protokola. To znači da ni klijent ni servis ne moraju da generišu i parsiraju SOAP poruke, jer to obavlja sam API
- Za prevođenje Java objekata u XML koristi se biblioteka JAXB (Java Architecture for XML Binding)

Referentne implementacije

- Metro nije Java EE specifikacija, već open-source referentna implementacija specifikacije za Java Web servise
- Sadrži JAX-WS i JAXB, a podržava i ranije API-je kao što je JAX-RPC
- Omogućuje programiranje i deploy Web servisa i klijenata
- Napravljen u Glassfish zajednici

UNIVERZITET PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

Razmena poruka

- Uprskos mnogobrojnim specifikacijama, pojmovima, standardima, API-jima i implementacijama, pisanje i pozivanje Web servisa je veoma lako
- Web servis može biti obična Java klasa sa anotacijom **@javax.jws.WebService** ako nam odgovaraju podrazumevana podešavanja

UNIVERZITET PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

Web servis: primer (1)

- Primer Web servisa koji proverava broj kreditne kartice (podrazumeva da je broj ispravan ako je neparan)

```
@WebService
public class CardValidator {

    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(➥
                              creditCard.getNumber().length() - 1);

        if (Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

Web servis: primer (2)

- Objekat CreditCard razmenjuje se između klijenta i SOAP Web servisa
- Razmena se obavlja preko XML dokumenta, pa je neophodan metod za transformisanje Java objekta u XML dokument
- Tu na scenu stupa JAXB sa jednostavnim anotacijama i svojim API-jem
- Objekat CreditCard mora da bude označen anotacijom **@javax.xml.bind.annotation.XmlRootElement**, i JAXB će ga transformisati između XML-a i Jave u oba smera

Web servis: primer (3)

- Uz anotacije izbegava se implementiranje XML parsiranja na niskom nivou, jer sve to obavlja JAXB u pozadini

```
@XmlRootElement
public class CreditCard {

    private String number;
    private String expiryDate;
    private Integer controlNumber;
    private String type;

    // Constructors, getters, setters
}
```

Web servis: primer (4)

- Klijent Web servisa treba samo da napravi instancu CreditCard i da pozove servis

```
public class Main {

    public static void main(String[] args) {

        CreditCard creditCard = new CreditCard();
        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/10");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator = ↪
                new CardValidatorService().getCardValidatorPort();

        cardValidator.validate(creditCard);

    }
}
```

Web servis: primer (5)

- Web servis CardValidator se ne poziva direktno, već preko klase CardValidatorService i njenog metoda getCardValidatorPort()
- Nakon što je dobio referencu na CardValidator, klijent može da pozove metodu validate() i da prosledi kreditnu karticu
- Vidljivi deo Web servisa u Javi ne koristi direktno XML, SOAP niti WSDL, ali se u pozadini dešavaju važne operacije za interoperabilnost, tj. kreiraju se artefakti

JAXB

- U pozadini, JAXB obavlja tzv. marshalling (transformisanje Java objekta u XML) i unmarshalling (iz XML-a u Javu)
- Takođe, automatski generiše XML šemu koja obavlja validaciju XML strukture

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<creditCard>
    <controlNumber>6398</controlNumber>
    <expiryDate>12/09</expiryDate>
    <number>1234</number>
    <type>Visa</type>
</creditCard>
```

JAXB: XML šema (1)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="creditCard" type="creditCard"/>

  <xs:complexType name="creditCard">
    <xs:sequence>
      <xs:element name="controlNumber" type="xs:int" minOccurs="0"/>
      <xs:element name="expiryDate" type="xs:string" minOccurs="0"/>
      <xs:element name="number" type="xs:string" minOccurs="0"/>
      <xs:element name="type" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

JAXB: XML šema (2)

- Sastoji se od prostih elemenata (controlNumber, expiryDate itd.) i složenog tipa (creditCard)
- Složeni tipovi modeliraju sadržaj elemenata, tj. određuju moguć skup elemenata koji se koriste u dokumentu
- Svi tagovi koriste prefiks xs (xs:element, xs:string itd.)
- Taj prefiks se zove imenik (namespace) i definisan je u tagu zaglavlja šeme dokumenta

XML imenik (namespace)

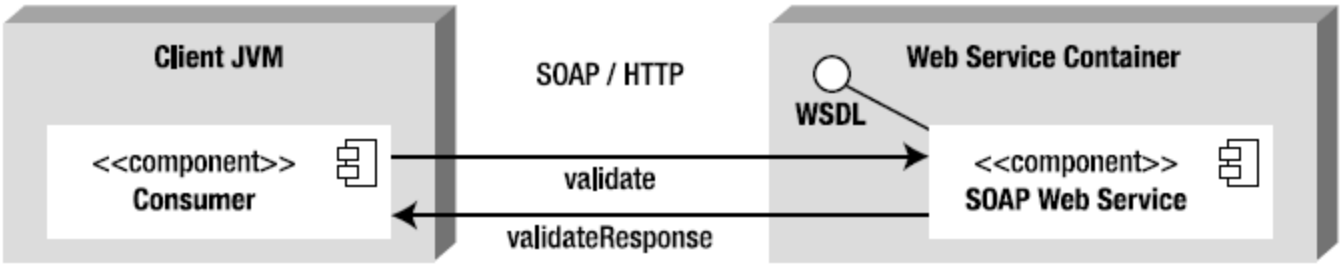
- Imenici kreiraju jedinstvene prefikse za elemente u posebnim dokumentima ili aplikacijama koje se koriste zajedno
- Koriste se da bi se izbegli problemi kada se isto ime za element pojavljuje u nekoliko dokumenata
  - npr. tag <element> se često pojavljuje u više dokumenata sa različitim značenjem
- Imenici su veoma važni za Web servise jer oni često rade sa više XML dokumenata istovremeno (SOAP koverta, WSDL dokument)

JAXB anotacije

@XmlAccessorType	Upravlja preslikavanjem atributa ili getera (FIELD, NONE, PROPERTY, PUBLIC_MEMBER)
@XmlAttribute	Preslikava atribut ili geter u XML atribut prostog tipa (String, Boolean, Integer i sl.)
@XmlElement	Preslikava nestatički i netranzijentni atribut ili geter u XML element
@XmlElements	Kontejner za više anotacija @XmlElement
@XmlEnum	Preslikava enum u XML predstavu
@XmlEnumValue	Označava nabrojanu konstantu
@XmlID	Identifikuje ključno polje XML elementa (tipa String), koje se može koristiti za referenciranje elementa pomoću anotacije @XmlIDREF
@XmlIDREF	Preslikava svojstvo u XML IDREF u šemi
@XmlList	Preslikava svojstvo u listu
@XmlMimeType	Identifikuje tekstualni prikaz MIME tipa za svojstvo
@XmlNs	Identifikuje XML imenik
@XmlRootElement	Obavezna anotacija za svaku klasu koja treba da bude korenski XML element
@XmlSchema	Preslikava ime paketa u XML imenik
@XmlTransient	Obaveštava JAXB da ne povezuje atribut (analogno Javinoj ključnoj reči transient)
@XmlType	Označava klasu kao složeni tip u XML šemi
@XmlValue	Omogućuje preslikavanje klase u jednostavan tip u šemi

Nevidljivi deo

- Kada klijent pozove Web servis CardValidator, dobiće njegov WSDL da bi znao koje interfejse može da koristi
- Zatim će zatražiti proveru kartice (SOAP poruka validate), i na kraju će dobiti odgovor (SOAP poruka validateResponse)



WSDL

- WSDL dokumenti se nalaze u kontejneru Web servisa, a koriste XML za opis onoga što servis radi, kako se pozivaju njegove operacije i gde se on nalazi
- Taj XML dokument ima tačno određenu strukturu:
  - <**definitions**> je root element
  - <**types**> definiše tipove podataka koji se koriste u porukama
  - <**message**> definiše format podataka koji se razmenjuju između Web servisa i klijenta
  - <**portType**> zadaje operacije Web servisa
  - <**binding**> opisuje konkretni protokol (ovde SOAP) i tipove podataka za operacije i poruke definisane za određeni tip porta
  - <**service**> je skup elemenata <port> elements, pri čemu je svaki port povezan sa nekim URL-om

WSDL

```
<definitions targetNamespace="http://chapter14.javaee6.org/" ↪
                        name="CardValidatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://chapter14.javaee6.org/" schemaLocation
        ="http://localhost:8080/chapter14/CardValidatorService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="validate">
    <part name="parameters" element="tns:validate"/>
  </message>
  <message name="validateResponse">
    <part name="parameters" element="tns:validateResponse"/>
  </message>
  <portType name="CardValidator">
    <operation name="validate">
      <input message="tns:validate"/>
      <output message="tns:validateResponse"/>
    </operation>
  </portType>
  <binding name="CardValidatorPortBinding" type="tns:CardValidator">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="validate">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="CardValidatorService">
    <port name="CardValidatorPort" binding="tns:CardValidatorPortBinding">
      <soap:address location = "http://localhost:8080/chapter14/CardValidatorService"/>
    </port>
  </service>
</definitions>
```

WSDL

- WSDL i njegova odgovarajuća šema generišu se alatima koji transformišu metapodatke o Web servisu u XML
- JAX-WS ima uslužne alate koji automatski generišu odgovarajuće artefakte
- WSDL opisuje apstraktni interfejs Web servisa, dok SOAP poruke obezbeđuju konkretnu implementaciju sa definicijom XML strukture poruka koje se razmenjuju

AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM

SOAP

- SOAP šalje XML poruku preko HTTP requesta i prima odgovor preko HTTP response
- SOAP poruka je XML dokument sa sledećim elementima:
  - <**Envelope**> obavezan, root tag koji definiše poruku i imenik koji se koriste u dokumentu
  - <**Header**> sadrži opcione atribute poruke ili aplikacije, npr. informacije o bezbednosti ili mrežnom rutiranju
  - <**Body**> obavezan tag koji sadrži poruku koja se razmenjuje između aplikacija
  - <**Fault**> obezbeđuje informacije o greškama koje su se desile tokom obrade poruke

SOAP poruke

The SOAP Envelope for the Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" ↵
                  xmlns:cc="http://chapter14.javaee6.org/">
   <soap:Header/>
   <soap:Body>
      <cc:validate>
         <arg0>
            <controlNumber>1234</controlNumber>
            <expiryDate>10/10</expiryDate>
            <number>9999</number>
            <type>VISA</type>
         </arg0>
      </cc:validate>
   </soap:Body>
</soap:Envelope>
```

The SOAP Envelope for the Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:validateResponse xmlns:ns2="http://chapter14.javaee6.org/">
         <return>true</return>
      </ns2:validateResponse>
   </soap:Body>
</soap:Envelope>
```

JAX-WS model

- Klasa koja implementira Web servis:
  - Mora da ima anotaciju @javax.jws.WebService
  - Mora da bude public, a ne sme da bude final niti abstract
  - Mora da ima default public konstruktor
  - Ne sme da definiše metod finalize()
  - Mora da bude stateless, tj. ne treba da pamti stanja klijenta između različitih poziva metoda

Pristupi kreiranju servisa

- SOAP Web servis može da počne da se razvija tako što će se napisati WSDL fajlovi, pa će onda na osnovu njih klijent generisati skup Java klasa
  - To je tzv. top-down approach, poznat i kao contract first
- Drugi pristup je da se WSDL generiše na osnovu postojećih Java klasa
  - to je bottom-up pristup
- Preporučuje se top-down pristup jer bi servisi trebalo da budu implementaciono neutralni
- Bottom-up se preporučuje kada postojeću aplikaciju želimo da osposobimo da bude servis

JAX-WS anotacije

- @WebService
  - Označava da je klasa ili interfejs Web servis
- @WebMethod
  - Preimenuje metod ili ga isključuje iz WSDL-a
- @WebResult
  - Zajedno sa anotacijom @WebMethod kontroliše generisano ime poruke u WSDL-u
- @WebParam
  - Prilagođava parametre metoda Web servisa

PRVI AKREDITOVANI UNIVERZITET SINGIDUNUM

JAX-WS anotacije

```
@WebService(name = "CreditCardValidator", portName = "ValidatorPort")
public class CardValidator {

    @WebMethod(operationName = "ValidateCreditCard")
    @WebResult(name = "IsValid")
    public boolean validate(@WebParam(name = "CreditCard") CreditCard creditCard) {

        Character lastDigit = creditCard.getNumber().charAt(➥
                              creditCard.getNumber().length() - 1);
        if (Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        } else {
            return false;
        }
    }

    @WebMethod(exclude = true)
    public void validate(String ccNumber) {
        // business logic
    }
}
```

Pozivanje SOAP Web servisa

- Alati wsimport i wsgen uključeni su u JDK 1.6 i GlassFish
- wsimport na osnovu WSDL fajla generiše JAX-WS artefakte, a wsgen čita klasu servisa i generiše WSDL

UNIVERZITET PRVI AKREDITOVANI PRIVATNI UNIVERZITET SINGIDUNUM