



Distance Learning System

Java XML i web servisi

SOAP servisi

Šta su veb-servisi?

- Veb-servis aplikacija ili softverska komponenta koja postoji na serveru i koja izlaže neku vrstu resursa klijentima preko mreže, a kompletna komunikacija se obavlja razmenom poruka.
- Takve poruke mogu biti konstruisane korišćenjem XML-a, i to je jedan od razloga zbog kojih su ovom modulu prethodili moduli posvećen XML-u. Ovako opisana komunikacija može se predstaviti sledećom slikom:



Svrha veb-servisa?

- Zamislimo da želimo da napravimo aplikaciju koja bi prikazivala televizijski program. Na primer, za 20 različitih kanala. Da bismo ovakvu aplikaciju opsluživali na dnevnom nivou, bilo bi nam potrebno da svakodnevno budemo u kontaktu s osobama iz 20 različitih televizija i da od njih preuzimamo programe, unosimo ih u aplikaciju i slično.
- Umesto toga, mogli bismo reći svakoj od televizija da nam pripremi program u nekom standardnom formatu, a zatim preuzimati tako pripremljen materijal i parsirati ga u našoj aplikaciji.
- Ovo bi znatno ubrzalo i standardizovalo proces. Ali mogli bismo otići i korak dalje, tako što ćemo tražiti da se i kompletno parsiranje izvrši kod njih, a da nama samo kažu naredbe koje možemo koristiti kako bismo došli do tih informacija.
- Poslednje rešenje, prepoznaje se kao veb-servis.

Posrednik u klijent server komunikaciji

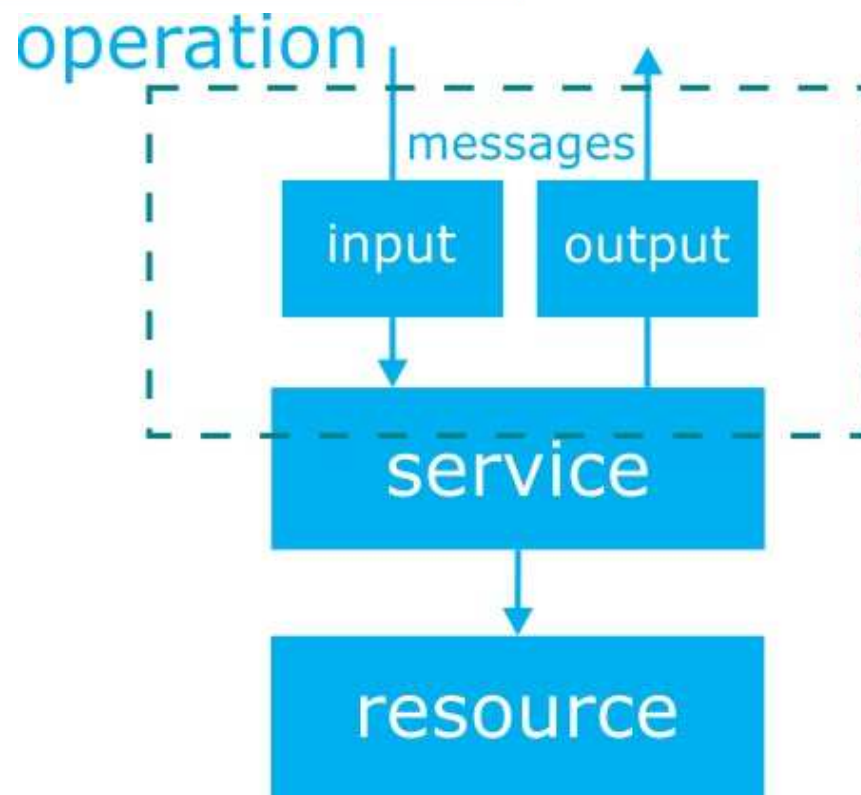
- Kompanije su za svoje poslovanje oduvek koristile klijentske ili serverske sisteme, u kojima klijentske aplikacije komuniciraju sa pozadinskom serverskom logikom preko posrednika koji je u arhitekturnom smislu postavljen između njih. Tradicionalni srednji sloj je donosio mnoge probleme, među kojima su najznačajniji teškoća i cena održavanja, fleksibilnost i veoma teško prilagođavanje komunikaciji preko interneta.
- Zbog svega ovoga, veb-servisi su zamišljeni kao nosioci novog srednjeg, posredničkog sloja u softverskoj arhitekturi enterprajz aplikacija, uz upotrebu veba i XML-a. Veb-servisi prevazilaze malopre navedene nedostatke korišćenjem open standarda i veba. Na taj način se postiže odlična fleksibilnost i znatno se olakšava održavanje.
- Još jedna veoma značajna prednost veb-servisa jeste mogućnost kompanija da zadrže postojeće informacione sisteme, koji često mogu biti napisani u najrazličitijim jezicima. Bez potrebe za pisanjem sistema od nule, potrebne funkcionalnosti mogu biti izložene korisnicima korišćenjem veb-servisa.

Tipovi veb-servisa

- Sa tehničke tačke gledišta, veb-servisi mogu biti implementirani na različite načine. Dva najčešća tipa veb-servisa su **SOAP** ili takozvani „big“ veb-servisi i **RESTful** veb-servisi.

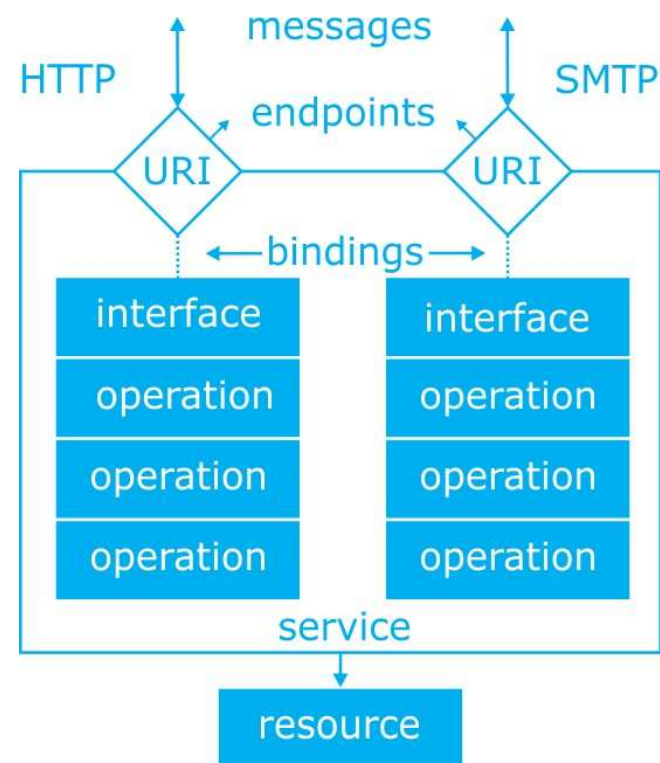
SOAP veb-servisi

- Jedan od najčešćih oblika veb-servisa jesu veb-servisi zasnovani na XML jeziku za definisanje poruka koji se naziva **SOAP**
- Razmena SOAP poruka naziva se operacija i podrazumeva poziv funkcije i dobijeni odgovor



Binding

- Binding sadrži konkretne detalje o tome kako je interfejs povezan sa protokolom za razmenu poruka. Kombinacija bindinga i URI-ja, naziva se krajnja tačka ili endpoint, a kolekcija ovih tačaka čini jedan veb-servis

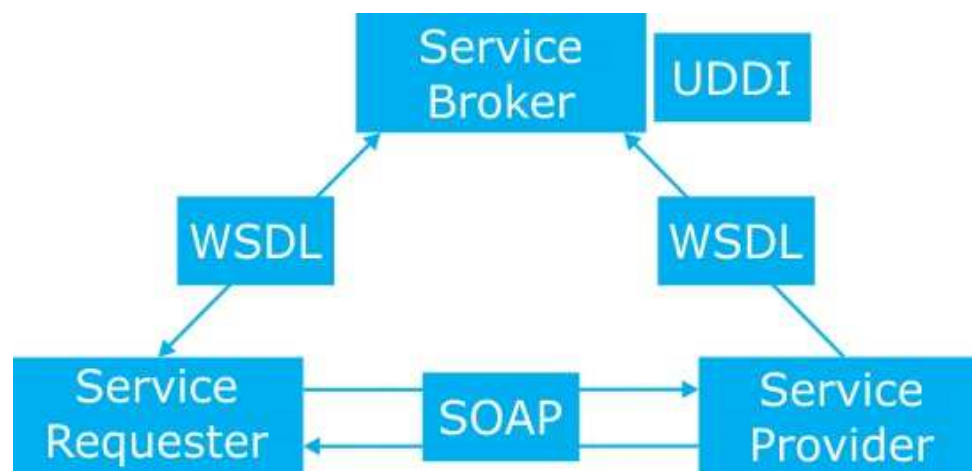


WSDL

- SOAP veb-servisi se najčešće koriste u sprezi sa jezikom za opis veb-servisa koji se skraćeno naziva **WSDL** (Web Services Description Language). Ovo je opisni jezik veb-servisa, zasnovan na XML-u, na osnovu koga je moguće ustanoviti šta servis radi i kako.
- WSDL je drugim rečima neka vrsta formalnog dogovora između SOAP veb-servisa i klijenata, po kome se definišu svi detalji koji su neophodni za interakciju sa veb-servisom

Okruženje SOAP servisa

- Okruženje u kome SOAP veb-servisi funkcionišu zahteva postojanje klijenta, odnosno onoga ko zahteva servis (service requester), servera, odnosno onoga ko omogućava servis i na kraju, servisnog brokera



RESTful veb-servisi

- SOAP veb-servisi mogu biti na raspolaganju preko brojnih protokola sa kojima smo se do sada upoznali: HTTP, [SMTP](#), [FTP](#), dok su RESTfull veb-servisi dostupni isključivo preko HTTP protokola.
- Centralni deo REST-a jeste [identifikator](#) resursa ili URI. Za identifikaciju resursa koriste se MIME tipovi. Kada klijent zahteva resurs od RESTfull veb-servisa, servis odgovara MIME reprezentacijom.
- Klijent koristi nama već poznate HTTP metode POST, GET, PUT i DELETE kako bi vršio manipulaciju resursima koje servis izlaže.
-

Veb-servisi u Javi

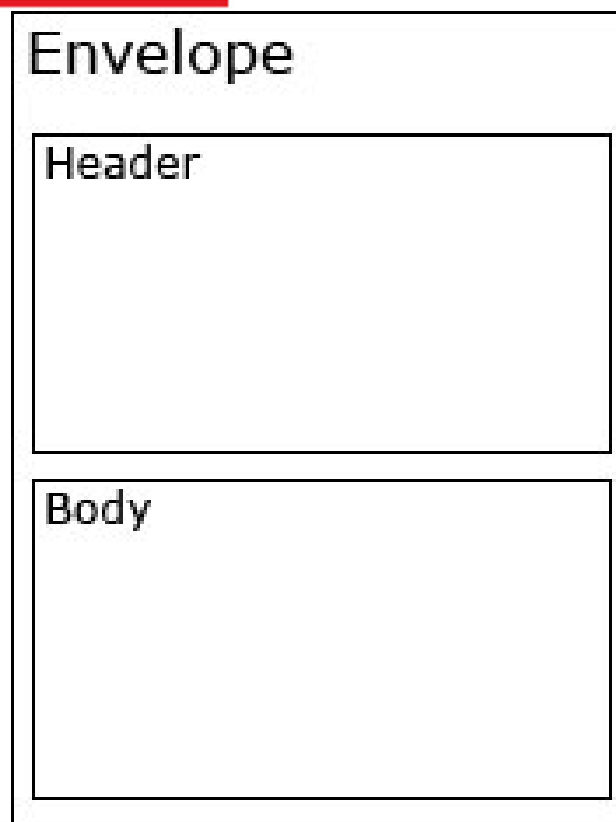
- S obzirom na to da su veb-servisi i XML veoma tesno povezani, sa mnogim funkcionalnostima neophodnim za korišćenje veb-servisa u Javi već smo se upoznali proučavanjem xml-a. Programski jezik Java poseduje nekoliko biblioteka funkcionalnosti za rad sa veb-servisima. Pored JAXP funkcionalnosti postoje i:
 - Java API for XML Web Services (**JAX-WS**) – ovo je glavni API za izgradnju veb-servisa i klijenata koji komuniciraju sa servisima korišćenjem XML-a; JAX-WS predstavlja zamenu za stariju Java tehnologiju koja se zove **JAX-RPC** – Java API for Remote Procedure Call; funkcionalnosti su smeštene u paketu javax.xml.ws,
 - Java Architecture for XML Binding – **JAXB** – ovo je API sa kojim smo se već upoznali i koji se koristi za mapiranje XML podataka u specifične Java tipove,
 - Soap with [Attachments](#) API for Java (**SAAJ**) – ovo je API za kreiranje, slanje i primanje SOAP poruka sa priložima ili bez njih i
 - Java API for RESTful Web Services ili skraćeno **JAX-RS** – definiše biblioteke koje implementiraju REST arhitekturu u Java servlet kontejneru; jednostavno, ova implementacija postoji na serveru i zadužena je da zahtev upućen određenom metodom mapira na odgovarajuću metodu same klase.

SOAP veb-servisi

- Simple Object Access Protocol
- Prenosi se najčešće (ali ne i uvek) putem http protokola
- Podrazumeva opis i poruke u xml formatu

Sastavni delovi SOAP dokumenta

- Soap envelope
 - Header
 - Sadrži dodatne informacije
 - Ne utiče na direktnu funkcionalnost servisa
 - Opcion je
 - Body
 - Sadrži prave podatke
 - Obavezan je



Soap http telo

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://www.nbs.rs/TempExchangeRatesService">
      <UserName>string</UserName>
      <Password>string</Password>
      <LicenceID>long</LicenceID>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <GetCurrentExchangeRates xmlns="http://www.nbs.rs/TempExchangeRatesService">
      <nExchangeRatesListTypeID>decimal</nExchangeRatesListTypeID>
    </GetCurrentExchangeRates>
  </soap:Body>
</soap:Envelope>
```

<http://pastie.org/pastes/10275175/text>

Korišćenje web servisa - SAAJ

- SAAJ (**SOAP with Attachments API for Java**) je Java API za rukovanje SOAP porukama na niskom nivou (XML)
- SAAJ se koristi u kombinaciji sa DOM XML-om
- SAAJ API se nalazi u paketu **javax.xml.soap**

Platforma sa primerima SOAP servisa

<http://www.webservicex.net/WS/wscatlist.aspx>

Kreiranje SAAJ XML poruke

- Glavni akter u komunikaciji sa servisom jeste jedna SOAP poruka
- SOAP poruka se u SAAJ-u predstavlja objektom klase **SOAPMessage**, a kreira pomoću klase **MessageFactory**:

```
MessageFactory mFac = MessageFactory.newInstance();  
SOAPMessage sReq = mFac.createMessage();
```


Kreiranje SAAJ XML poruke

- Svaka poruka sadrži "soap part". Ovaj deo se automatski kreira u poruci, i sadrži sve ostale elemente poruke.

```
SOAPPart sPart = sReq.getSOAPPart();
SOAPEnvelope envelope = sPart.getEnvelope();
envelope.setPrefix("soap");
envelope.addNamespaceDeclaration("xsi", "http://www.w3.org/2001/XMLSchema-instance");
envelope.addNamespaceDeclaration("xsd", "http://www.w3.org/2001/XMLSchema");
envelope.addNamespaceDeclaration("soap", "http://schemas.xmlsoap.org/soap/envelope/");
envelope.addNamespaceDeclaration("web", "http://www.webserviceX.NET/");
envelope.removeAttribute("xmlns:SOAP-ENV");
envelope.getHeader().setPrefix("soap");
SOAPBody sBody = envelope.getBody();
sBody.setPrefix("soap");
```



```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://www.webserviceX.NET/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Header/>
  <soap:Body>
```

Kreiranje SAAJ XML poruke


- Rukovanje elementima poruke je slično DOM-u
- Metod writeTo, omogućava slanje kreirane poruke u tok

```
SOAPElement cRateElement = sBody.addChildElement("ConversionRate", "web");
SOAPElement fromCurrencyElement = cRateElement.addChildElement("FromCurrency", "web");
fromCurrencyElement.setValue("EUR");
SOAPElement toCurrencyElement = cRateElement.addChildElement("ToCurrency", "web");
toCurrencyElement.setValue("USD");
sReq.getMimeHeaders().addHeader("SOAPAction", "http://www.webserviceX.NET/ConversionRate");
sReq.writeTo(System.out);
```

Kreiranje SAAJ XML poruke

- Rukovanje elementima poruke je slično DOM-u
- Metod writeTo, omogućava slanje kreirane poruke u tok

```
SOAPElement cRateElement = sBody.addChildElement("ConversionRate", "web");
SOAPElement fromCurrencyElement = cRateElement.addChildElement("FromCurrency", "web");
fromCurrencyElement.setValue("EUR");
SOAPElement toCurrencyElement = cRateElement.addChildElement("ToCurrency", "web");
toCurrencyElement.setValue("USD");
sReq.getMimeHeaders().addHeader("SOAPAction", "http://www.webserviceX.NET/ConversionRate");
sReq.writeTo(System.out);
```



```
<soap:Body>
  <web:ConversionRate>
    <web:FromCurrency>EUR</web:FromCurrency>
    <web:ToCurrency>USD</web:ToCurrency>
  </web:ConversionRate>
</soap:Body>
```

Kreiranje SAAJ XML poruke

- Da bi poruka bila poslata, potrebno je kreirati SOAP konekciju
- Metod call, šalje sinhronu poruku servisu
- Dolazna poruka takođe ima metod writeTo, koja vrši ispis u tok

```
SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();  
SOAPConnection sc = scf.createConnection();  
  
SOAPMessage sRes = sc.call(sReq, "http://www.webservices.net/CurrencyConvertor.asmx");  
sRes.writeTo(System.out);
```

Parsiranje poruke

- Svi xml elementi SOAP poruke predstavljaju se DOM elementima, pa je tako za parsiranje dolazne poruke najbolje koristiti DOM api:

```
SOAPBody responseBody = sRes.getSOAPBody();
System.out.println(
    responseBody.getElementsByTagName("ConversionRateResult").
        item(0).getTextContent()
);
```

Vežba

(jwsx042014 SAAJExample)

- Pokušajte samostalno da iskoristite neki od servisa (po sopstvenom izboru) sa sledeće adrese
- <http://www.websvcex.net/WS/wscatlist.aspx>

Kreiranje SOAP web servisa

- Java sadrži biblioteku za rad sa web servisima (**JAX-WS**)
- Ova biblioteka nalazi se u sklopu **Java EE**

Struktura JAX-WS SOAP servisa (jwsx042014 AddNumbers)

- JAX-WS realizuje SOAP servis uz pomoć tri komponente
 - **Ugovor servisa**
 - **Implementacija servisa**
 - **Distributer servisa (endpoint)**
- Ugovor servisa je interfejs koji opisuje funkcionalnosti servisa
- Implementacija servisa su stvarne funkcionalnosti, koje rade ono što je definisano interfejsom
- Da bi servis bio dostupan putem url-a (a što najčešće treba da bude slučaj), on mora da bude hostovan na neki način
- Servisi mogu biti hostovani pomoću sopstvenog sistema (self hosting), ili u sastavu nekog postojećeg web servera.

Kreiranje ugovora servisa

- Prvi korak u kreiranju servisa je kreiranje njegovog ugovora
- Ugovor servisa nije ništa drugo do interfejs dekorisan anotacijama **WebService** za interfejs, I **WebMethod** za metode

WebService

```
import javax.jws.WebMethod;
import javax.jws.WebService;
@WebService
public interface AddNumbersContract {
    @WebMethod
    public double add(double opa, double opb);
}
```

WebMethod

Kreiranje implementacije servisa

- Implementacija servisa je klasa koja implementira interfejs kojim se definiše ugovor servisa
- Ova klasa mora biti dekorisana anotacijom WebService
- U ovom slučaju, anotacija WebService mora da ima definisan atribut endpointInterface, koji označava ugovor na koji se servis odnosi

```
import javax.jws.WebService;

@WebService
public class AddNumbersImpl implements AddNumbersContract {
    @Override
    public double add(double opa, double opb) {
        return opa + opb;
    }
}
```

Hosting servisa – self hosting

- U slučaju self hostinga, dovoljno je da se aktivira metod publish, klase Endpoint, i da mu se kao parametri proslede respektivno: ciljna adresa servisa i instanca klase servisa

```
import javax.xml.ws.Endpoint;

public class AddNumbers {
    public static void main(String[] args) {
        Endpoint.publish(
            "http://localhost:8083/addnumbers",
            new AddNumbersImpl()
        );
    }
}
```

**Naravno, u produkciji,
servis neće imati
lokalnu adresu**

Hosting servisa – na postojećem serveru

- Kada postoji server, tada se servis distribuira u obliku standardne web aplikacije, a deploy se vrši u zavisnosti od servera

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a

Location: ☒ Packaged File to Be Uploaded to the Server

TempVerter.war

☐ Local Packaged File or Directory That Is Accessible from Gl

Type: *

Context Root:
Path relative to server's base URL.

Application Name: *

Virtual Servers:
Associates an Internet domain name with a physical serve

Status: ☒ **Enabled**
Allows users to access the application.

Implicit CDI: ☒ **Enabled**
Implicit discovery of CDI beans

Precompile JSPs: ☐
Precompiles JSP pages during deployment.

Run Verifier: ☐
Verifies the syntax and semantics of the deployment desc

Force Redeploy: ☐
Forces redeployment even if this application has already b

Keep State: ☐
Retains web sessions, SFSB instances, and persistently c

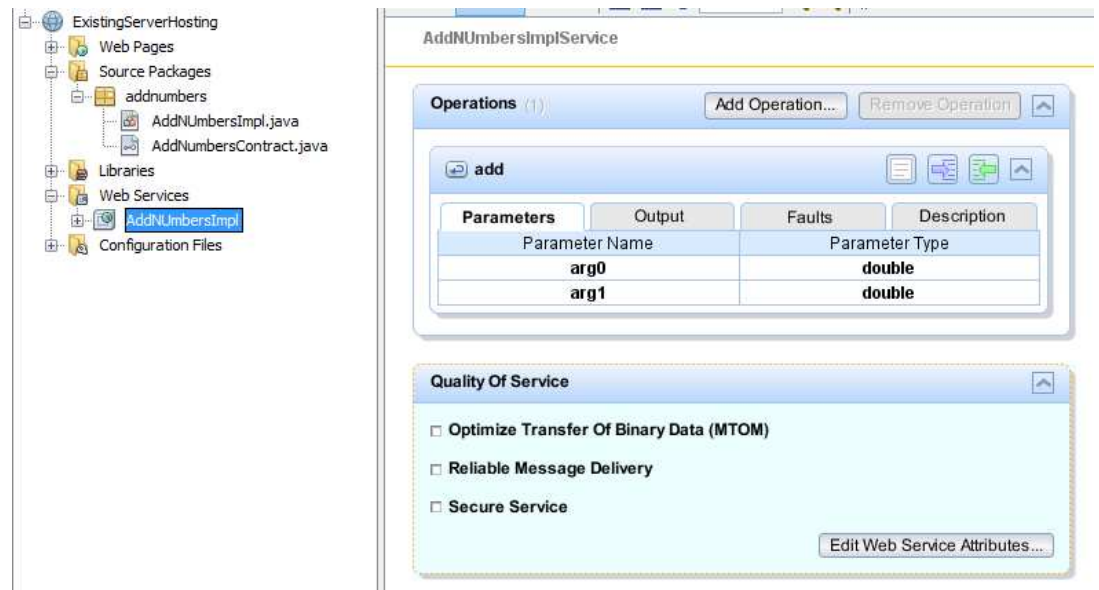
Deployment Order:
A number that determines the loading order of the applicati

Libraries:
A comma-separated list of library JAR files. Specify the lib

Description:

Kreiranje SOAP servisa u okviru web aplikacije

- Često se u okviru postojeće web aplikacije kreira i servis
- NetBeans ima dodatne opcije koje pomažu prilikom konstruisanja servisa u ovakvim situacijama
- Takođe, postoji i GUI editor za vizualnu reprezentaciju servisa
- Servis je moguće testirati iz netbeans-a, pomoću glassfish-ove aplikacije za testiranje servisa



WSDL

- WSDL je dokument koji je napisan korišćenjem XML-a. On služi za opisivanje veb-servisa i definisanje servisne lokacije i operacija koje servis izlaže. Stoga se ovakav dokument koristi najčešće od klijenata ili drugih servisa koji bi koristili funkcionalnosti postojećeg servisa, kako bi znali na koji način da se povežu sa servisom.
- Ako se za SOAP poruke može reći da služe za prenošenje podataka i komunikaciju između servisa i klijenta, za WSDL se može reći da pruža informacije o tome kako da se pomenuta komunikacija ostvari.
- WSDL servisa kreiranog u ovoj lekciji može se videti na sličan način kao što je obavljeno i testiranje



```
<!--...-->
<!--...-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/a
xmlns:tns="http://addnumbers/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.o
```

Definitions

- Koreni element WSDL dokumenta je **definitions** element koji čini WSDL dokument skupom definicija. Otvarajući tag definitions elementa izgleda ovako (zbog preglednosti su uklonjeni prostori imena):

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://addnumbers/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://addnumbers/" name="AddNumbersImplService">
```

- Atribut *targetNamespace* kreira prostor imena za sve korisnički definisane elemente, dok se atribut *name* odnosi naravno na naziv servisa.
- U okviru korenog definitions elementa postoje ugneždeni sledeći elementi: *types*, *message*, *portType*, *binding* i *service*.

Types

- Element **types** predstavlja korisnički definisane tipove podataka. Za definisanje korisničkih tipova podataka uglavnom se koristi XML Schema.

```
▼<types>
  ▼<xsd:schema>
    <xsd:import namespace="http://addnumbers/"
      schemaLocation="http://localhost:8080/ExistingServerHosting/AddNumbersImplService?xsd=1"/>
    </xsd:schema>
  </types>

  ▼<xs:complexType name="add">
    ▼<xs:sequence>
      <xs:element name="arg0" type="xs:double"/>
      <xs:element name="arg1" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  ▼<xs:complexType name="addResponse">
    ▼<xs:sequence>
      <xs:element name="return" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
```


Message

- **<message>** elementi definišu elemente operacije. Svaki message element se sastoji iz više delova. Ovi delovi mogu da budu shvaćeni kao parametri [funkcija](#) standardnih programskih jezika.

```
▼<message name="add">
  <part name="parameters" element="tns:add"/>
</message>
▼<message name="addResponse">
  <part name="parameters" element="tns:addResponse"/>
</message>
```

- Nakon message elemenata naveden je **<portType>** element. Ovo je najvažniji WSDL element. On opisuje veb-servis, operacije koje on može da obavi i poruke koje se tim poslom mogu razmeniti.

```
▼<portType name="AddNUmbersImpl">
  ▼<operation name="add">
    <input wsam:Action="http://addnumbers/AddNUmbersImpl/addRequest" message="tns:add"/>
    <output wsam:Action="http://addnumbers/AddNUmbersImpl/addResponse" message="tns:addResponse"/>
  </operation>
</portType>
```

Binding

- Deo u kome su povezane sve komponente servisa naziva se binding:

```
▼<binding name="AddNumbersImplPortBinding" type="tns:AddNumbersImpl">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  ▼<operation name="add">
    <soap:operation soapAction=""/>
    ▼<input>
      <soap:body use="literal"/>
    </input>
    ▼<output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

- Podaci najvišeg nivoa o servisu, nalaze se u elementu **service**. Naziv servisa, naziv porta, binding i sama adresa SOAP servera:

```
▼<service name="AddNumbersImplService">
  ▼<port name="AddNumbersImplPort" binding="tns:AddNumbersImplPortBinding">
    <soap:address location="http://localhost:8080/ExistingServerHosting/AddNumbersImplService"/>
  </port>
</service>
```

Korišćenje servisa

- Osim SAAJ-a predstavljenog na prethodnim slajdovima, SOAP web servis se u Javi može konzumirati i na druge načine
 - Automatizovano – korišćenjem netbeans funkcionalnosti
 - Ručno, korišćenjem jax ws klasa za rad sa servisima

Kreiranje servisnog klijenta - **wsimport**

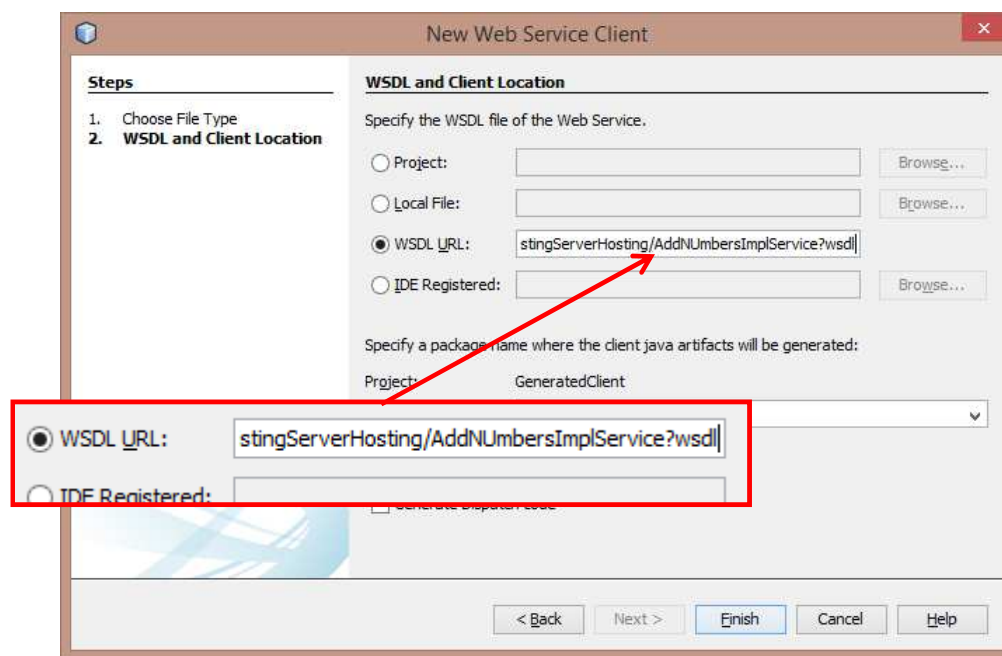
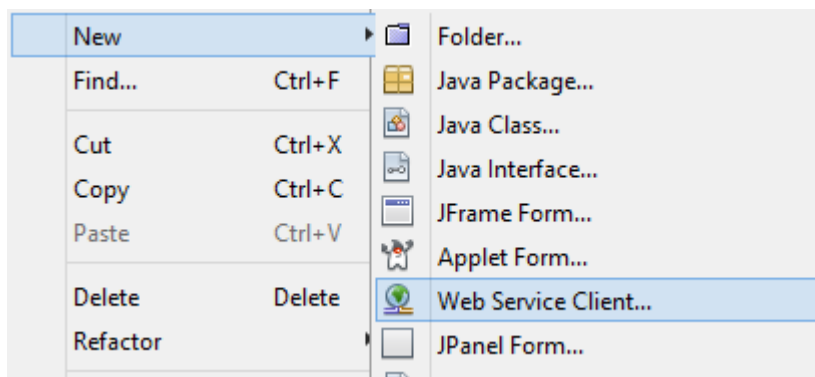
(jwsex042014 ConsumingAddService)

- Moguće je automatski generisati klijentsku logiku i to korišćenjem alata **wsimport**. Ovaj alat se nalazi u bin folderu instalacionog JDK foldera, baš kao i svi alati sa kojima smo do sada radili.
- Program na osnovu wsdl dokumenta generiše klase neophodne za upotrebu servisa
- **wsimport -sAddNUmbersImplService?wsdl**
- Generisane klase, kasnije se standardno koriste u aplikaciji:

```
addnumbers.AddNUmbersImplService service = new addnumbers.AddNUmbersImplService();  
addnumbers.AddNUmbersImpl add = service.getAddNUmbersImplPort();  
System.out.println(add.add(6, 3));
```

Kreiranje klijenta pomoću okruženja (jwsx042014 GeneratedClient)

- NetBeans sadrži šablon za kreiranje SOAP servisnog klijenta



Vežba – wallet servis

(jwsx042014 WalletService / WalletServiceOR)

- Potrebno je kreirati servis za plaćanje.
 - Servis zahteva logovanje korisnika (korisničko ime i šifru)
 - Nakon logovanja, korisnik dobija token (bilo kakav hash)
 - Nakon svakog sledećeg zahteva, korisnik šalje željenu operaciju (withdraw, deposit ili getBalance) zajedno sa tokenom
 - Ukoliko je token ispravan, i operacija moguća (korisnik ima dovoljno novca na računu), operacija se izvršava a korisnik dobija podatke o trenutnom stanju na računu