Realizacija Web servisa

1. Uvod

- Šta izaziva potrebu za poslovnom integracijom?
  - – integracija poslovnih aplikacija
  - – spajanja i preuzimanja kompanija
  - – integracija poslovnih partnera

- • Kako se ostvaruje poslovna integracija?
  - – integrisanjem aplikacija
  - – razmenom informacija i saradnjom (interoperabilnost)
  - – standardima

1. Uvod

- Primeri uobicajenih tehnologija distribuiranih racunarskih sistema koje su vec u širokoj upotrebi:
  - – CORBA
    - • Common Object Request Broker Architecture od OMG-a
    - • Jezicki nezavisan i objektno orijentisani standard distribuiranog sistema
    - • **Problem**: nije zagarantovana interoperabilnost ako distributeri nisu implementirali potpunu CORBA specifikaciju
  - – DCOM
    - • Distributed Component Object Model
    - • Microsoftova tehnologija za distribuirano objektno programiranje
    - • **Problem**: implementacija samo u Windows-u, što ogranicava interoperabilnost
    - – Java RMI
  - • Remote Method Invocation
    - • Java distribucija objekata
    - • **Problem**: nije jezicki nezavisna
  - – RMI preko IIOP
    - • Java RMI-u slican stil programiranja koristeci CORBA IIOP za komunikaciju
    - • Tehnike koje klijenti koriste da pristupe poslovnim zrnima Jave
    - (Enterprise JavaBeans)
    - • **Problem**: zavisnost od CORBA-e za komunikaciju i od Jave za interfejs

1. Uvod

- Performanse i proširivost
- • Menadžment
- • Postizanje labavog uparivanja(loose coupling)
- • Definisanje granularnosti interakcija
- • Održavanje konteksta preko više sistema
- –transakcije, bezbednost, stanje sesija
- • Konverzija protokola
- • Pracenje verzija
- • Firewall i drugi bezbednosni izazovi
- • Mrežni protok
- • Otpornost na otkaze (failover)
- • Uocavanje problema i analiza uzroka
- • Cena razvoja, integracije i testiranja

1. Uvod

- *Arhitekturalni stil projektovanja distribuiranih sistema koji obezbeduje aplikativnu funkcionalnost u vidu **servisa** bilo korisnicima aplikacije bilo drugim servisima*

- • Koristi otvorene standarde kao osnovu za prikazivanje fukcionalnosti softvera u vidu servisa
- • Omogucava standardan nacin za prikaz i interakciju funkcionalnosti softvera
- • Dozvoljava da pojedinacne funkcionalnosti softvera budu gradivni blokovi koji se mogu iskoristiti u razvoju drugih aplikacija
- • Fokusira se na sklop aplikacije umesto na implementacione detalje
- • Može se interno koristiti za stvaranje novih aplikacija od postojecih komponenti
- • Može se eksterno koristiti za integraciju sa aplikacijama drugih kompanija

1. Uvod

- *Servis je funkcionalnost aplikacije koja je upakovana kao komponenta koja može biti korišcena više puta u poslovnom procesu.*

- • Obezbedjuje dobro definisan interfejs
- • Deluje kao nezavisna funkcija
- • Koristi poruke da sakrije implementacione detalje
- • Ne zavisi od stanja drugih servisa
- • Pruža informacije pozivaocu i omogucava prevodjenje poslovnih podataka iz jednog validnog i konzistentnog stanja u drugo

1. Uvod

- Web servisi su softverski sistemi dizajnirani 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.

- *“Softverski sistem identifikovan uz pomoc URI, definisan, opisan i otkriven pomocu XML artefakta, koji interaguje sa drugim softverskim sistemima koristeci XML poruke preko protokola baziranih na Internetu."*

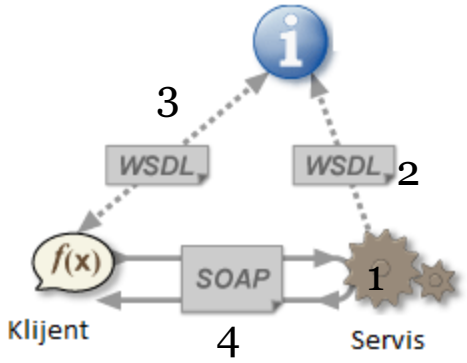
- *(W3C Web Services Architecture Group)*

1. Uvod

- Osnovni koraci u procesu kreiranja Web servisa:

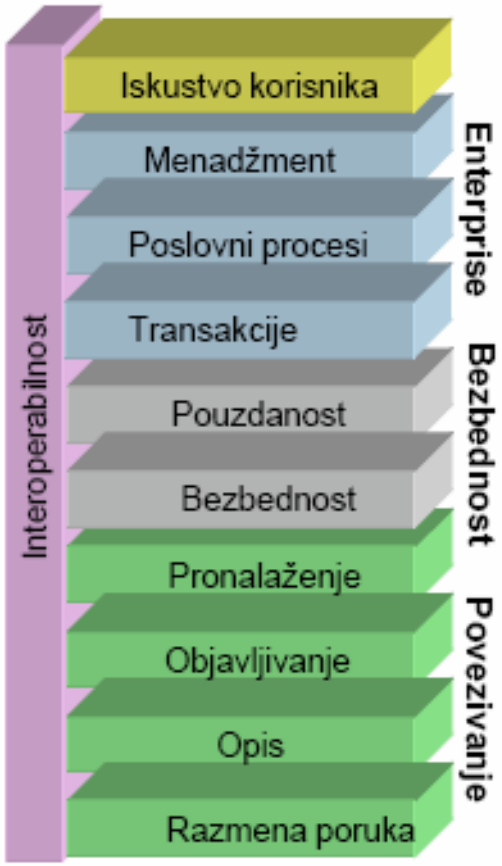
1. Pisanje Web servisa,
2. Objavljivanje Web servisa(WSDL)
3. Pronalaženje Web servisa
4. Povezivanje klijenta sa servisom(SOAP)

Arhitektura Web servisa



1. Uvod

- • Opisan jezikom WSDL koji ne zavisi od transportnog protokola
- • Proširiv
- Interoperabilnost
- • Polazi od dobrih principa distribuiranog dizajna
- • Tehnologija koja omogucava SOA i sisteme na zahtev
- • Standardi, standardi, standardi
- Iskustvo korisnika
- Menadžment Poslovni procesi Transakcije
- Pouzdanost
- Bezbednost
- Pronalaženje
- **Povezivanje**
- **Enterprise**
- **Bezbednost**
- Objavljivanje



1. Uvod

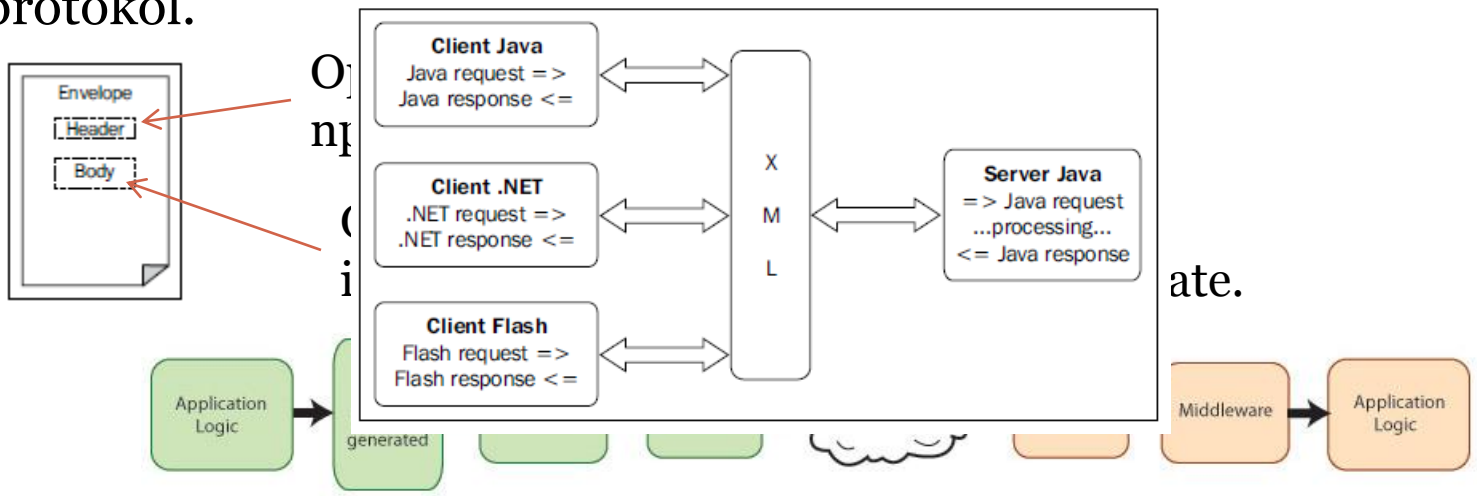
- Uspešne implementacije SOA i web servisa obezbedila su rešenja za izazove u brojnim industrijama:
- **Transport**
  - Potreba za procenjivanjem realnih cena je otežana kada pošiljku prenosi više prevoznika
- **Bankarstvo**
  - Potreba za obezbedjivanjem bankarskih usluga vecem broju korisnika
  - Potreba za integrisanjem velikog broja internih
  - aplikacija
- **Državna uprava**
  - Potreba za deljenim pristupom podacima od strane više resora

1. Uvod

- Interoperabilnost izmedu heterogenih sistema
- B2B (business-to-business) interakcije
  - jeftiniji od EDI (Electronic Data Interchange) i drugih zašticenih alternativa
- Višestruki zahtevi
  - narocito za heterogene platforme, protokole i kanale
- Enkapsulacija postojecih sistema
- Zaštita od izmene sistema
- Standardizacija servisa u okviru velikih projekata (enterprise)
- Integrisanje third-party paketa
- Koreografija procesa
- Razvoj portala

WSDL i SOAP

- Web servisi se objavljuju i pokreću na serveru.
- Klijent mora da zna koje usluge servis nudi.
- Za opis usluga koristi se WSDL(Web Service Description Language).
- Postoji mogućnost da se iz WSDL kreiraju klase kako za serversku, tako za klijentsku stranu
- SOAP (Simple Object Access Protocol) je standardni komunikacioni protokol.



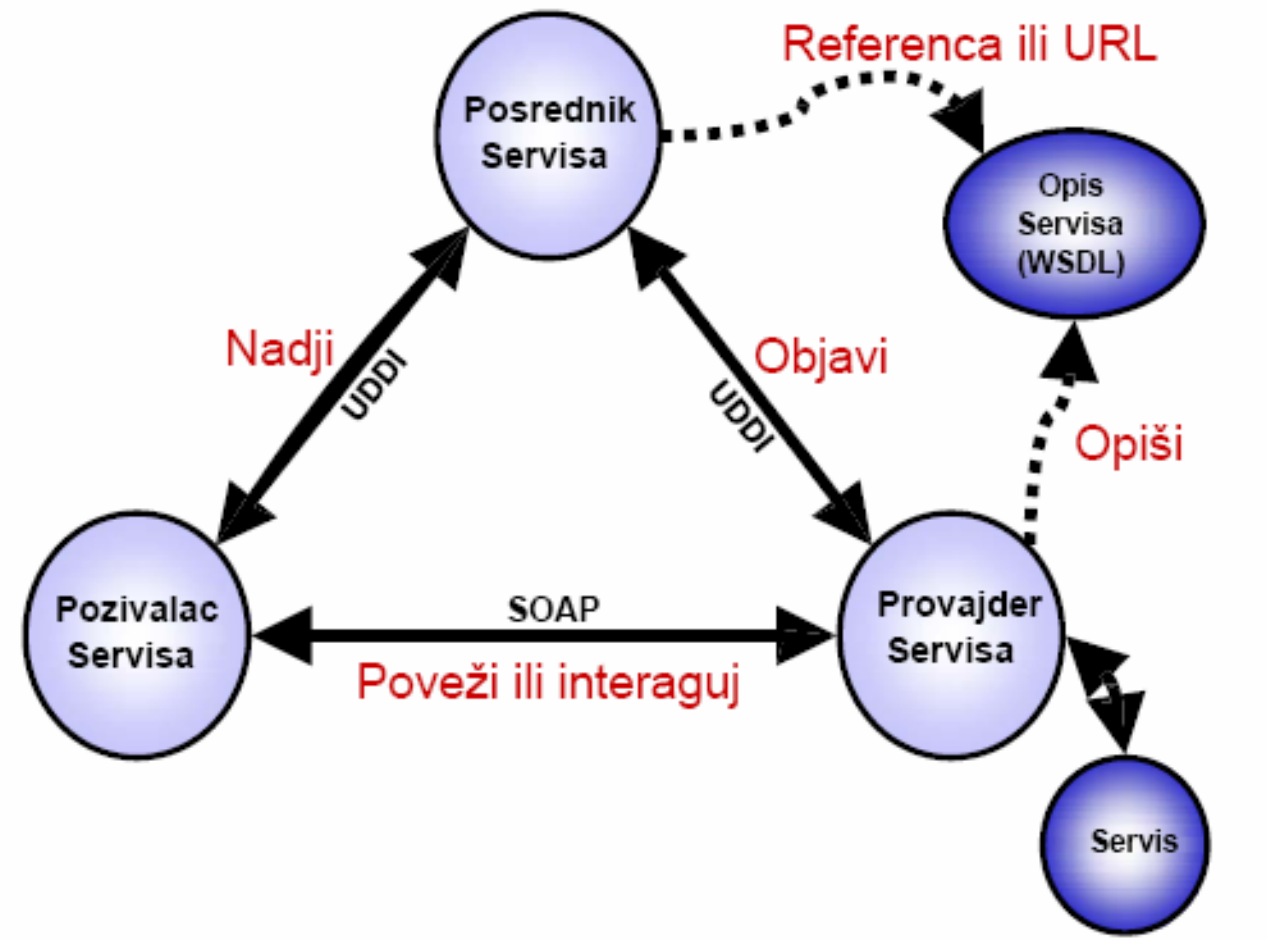
SOAP

- Protokol za prenos XML poruka kroz mrežu
  –SOAP poruka je sama po sebi XML dokument
- Nezavisan od vrste mreže, transporta i programskog jezika

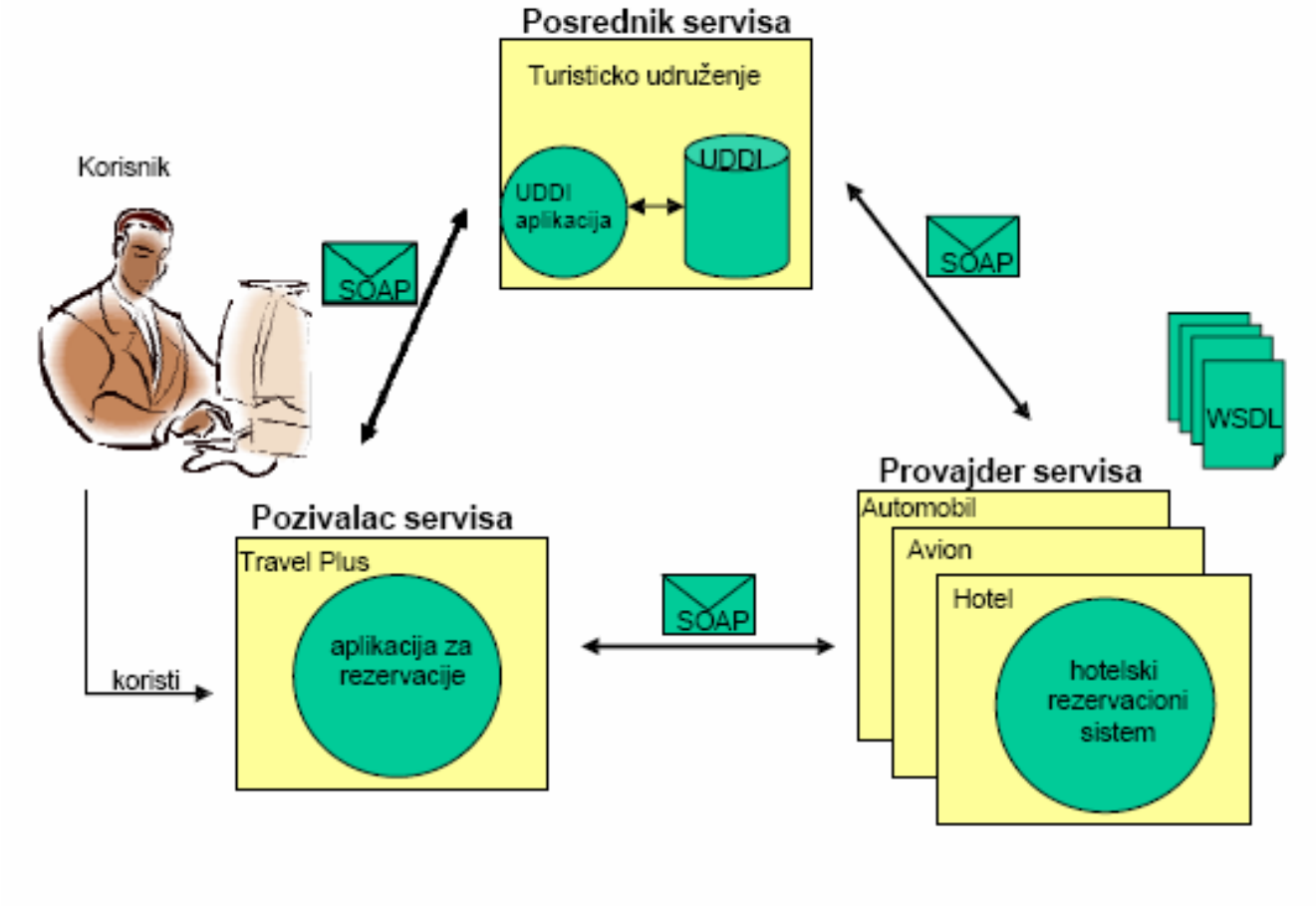
WSDL

- Web Services Definition Language (jezik za definisanje web servisa)
- Opisuje interfejs web servisa koristeci XML
  - –Povezivanja
  - –Operacije
  - –Lokacije specificne implementacije
- Apstraktno definiše operacije i poruke
- Koristi XML šemu za cuvanje informacija o tipovima

Odnosi izmedu osnovnih standarda

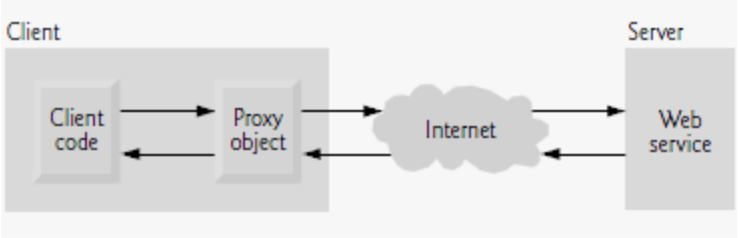


Web servis aplikacija



JAX-WS (Java Api for XML Web Services)

- Javin standard za pisanje Web servisa (deo JDK 6).
- Implementiraju ga mnogi frejmvorci kao i Apache CXF.
- Koristi anotacije za definisanje Web servisa (paket `javax.xml.ws`). Npr. `@WebService`, `@WebMethod`.
- Za protokol u komunikaciji koristi se SOAP.
- Moguće je pisati Web servise na višem i nižem nivou apstrakcije (Web Service, WebServiceProvider).
- Za prevođenje Java objekata u XML koristi se JAXB (Java Architecture for XML Binding) biblioteka.



2. Apache CXF

- Open source frejmvork za pisanje Web servisa. Uključuje širok skup funkcija.
- Podrška za Web servis standarde (SOAP, WSDL, WS-Security, WS-Security Policy...).
- Lakoća korišćenja, jednostavno pisanje i objavljivanje Web servisa, integracija sa Spring frejmvorkom, uvođenje WS-Security-ja...
- Omogućava razne načine za pisanje servisne i klijentske strane.
- Podržava pisanje REST Web servisa, implementira JAX-WS standard. Za pisanje Web servisa bez anotacija nudi Simple Frontend API.

Kreiranje servisne strane pomoću JAX-WS-a

- Moguce jekoristiti JAX-WS standard za pisanje Web servisa (i za klijentsku i za servisnu stranu), pri čemu postoje sledeći pristupi:

1. Pisanje servisa na višem i nižem nivou apstrakcije.
   - Ako želimo da sakrijemo rad sa SOAP porukama koristimo Web servise (`@WebService` anotaciju).
   - Za rad na nižem nivou apstrakcije koriste se Web servis provajderi (`@WebServiceProvider`).
2. Top-down i bottom-up pristup.
   - Top down pristup podrazumeve da se prvo napiše ugovor (WSDL) pa onda automatski generišu klase iz WSDL-a. (WSDL2Java alat).
   - Bottom-up pristup polazi od pisanja Java anotiranog koda, prilikom objavljivanja servisa generiše se automatski WSDL fajl.

- 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.

Kreiranje klijentske strane

- JAX-WS generisani klijent pomoću WSDL2Java alata,
- JAX-WS proxy,
- JAX-WS Dispatch API (za direktan rad sa SOAP porukama),
- Simple Frontend Client Proxy,
- Dinamički klijent (Dynamic Client).

Implementacija Web servisa

- Servis koji za zadatu godinu studija vraća listu studenata sa te godine.
- Koraci:
  1. Napisati WSDL fajl.
  2. Generisati sve servisne klase pomoću WSDL2Java alata (ako pišemo CXF klijenta možemo i sve klijentske klase generisati pomoću ovog alata).
  3. Napisati potrebnu logiku servisa.
  4. Objaviti servis (Tomcat, Spring).
  5. Testiranje
  6. Implementirati WS-Security (zaštita SOAP poruka koje se šalju HTTP protokolom).

WSDL (1)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns=http://studentService
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd=http://www.w3.org/2001/XMLSchema
name="StudentList" targetNamespace="http://studentService"
xmlns:tns2="http://studentService/types">

  <wsdl:types>
    <xsd:schema targetNamespace="http://studentService/types"
    xmlns:tns=http://studentService/types
    elementFormDefault="qualified">
      <xsd:element name="getStudentList">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="year" type="xsd:int" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getStudentListResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="return" maxOccurs="unbounded"
            type="tns:Student" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:complexType name="Student">
        <xsd:sequence>
          <xsd:element name="studentName"
          type="xsd:string">
          </xsd:element>
          <xsd:element name="studentID"
          type="xsd:string">
          </xsd:element>
          <xsd:element name="department"
          type="xsd:string">
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="wrongYearExceptionDetail">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorMessage" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
```

WSDL (2)

```
<wsdl:message name="getStudentListRequest">
  <wsdl:part element="tns2:getStudentList" name="parameters"
/>
  </wsdl:message>

  <wsdl:message name="getStudentListResponse">
   <wsdl:part element="tns2:getStudentListResponse"
   name="parameters" />
  </wsdl:message>
  <wsdl:message name="wrongYearException">
   <wsdl:part name="wrongYearExceptionDetail"
   element="tns2:wrongYearExceptionDetail">
   </wsdl:part>
  </wsdl:message>

  <wsdl:portType name="StudentList">
   <wsdl:operation name="getStudentList">
    <wsdl:input message="tns:getStudentListRequest" />
    <wsdl:output message="tns:getStudentListResponse" />
    <wsdl:fault name="wrongYearException"
    message="tns:wrongYearException">
    </wsdl:fault>
   </wsdl:operation>
  </wsdl:portType>
```

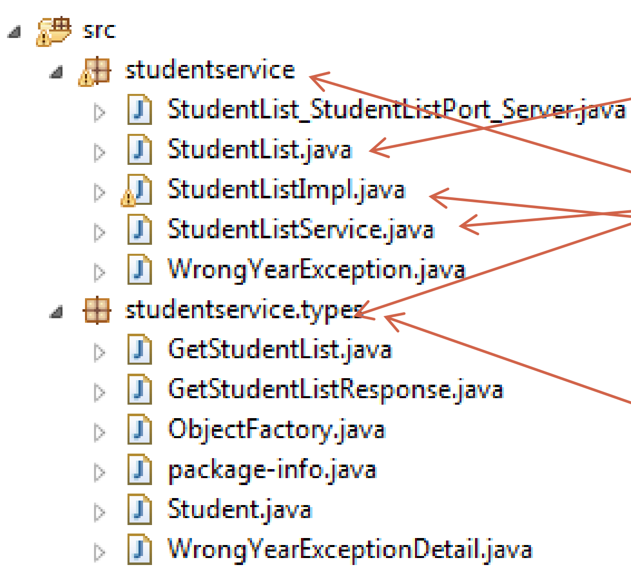
```
<wsdl:binding name="StudentList_SOAPBinding"
 type="tns:StudentList">
  <soap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="getStudentList">
   <soap:operation soapAction="" />
   <wsdl:input>
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
   <wsdl:fault name="wrongYearException">
    <soap:fault name="wrongYearException" use="literal" />
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>

  <wsdl:service name="StudentListService">
   <wsdl:port binding="tns:StudentList_SOAPBinding"
   name="StudentListPort">
    <soap:address
    location="http://localhost:9000/StudentService/StudentList" />
   </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

WSDL2 Java alat

- Pozivom WSDL2Java alata vrši se mapiranje iz korišćenog WSDL fajla u java klase. Može se koristiti i za klijentsku i za servisnu stranu.
- Opšti oblik naredbe je wsdl2java -ant -impl -server -d *outputDir* wsdlfile.wsdl



<u>StudentList.java</u>

```
package studentservice;

import javax.jws.WebMethod;
import javax.jws.WebParam;
...
@WebService(targetNamespace = "http://studentService", name = "StudentList")
@XmlSeeAlso({studentservice.types.ObjectFactory.class})
public interface StudentList {
 @WebResult(name = "return", targetNamespace =
 "http://studentService/types")
 @RequestWrapper(localName = "getStudentList", targetNamespace =
 "http://studentService/types", className =
 "studentservice.types.GetStudentList")
 @ResponseWrapper(localName = "getStudentListResponse",
 targetNamespace = "http://studentService/types", className =
 "studentservice.types.GetStudentListResponse")
 @WebMethod
 public java.util.List<studentservice.types.Student> getStudentList(
 @WebParam(name = "year", targetNamespace =
 "http://studentService/types")
 int year
 ) throws WrongYearException;
}
```

Interfejs Web servisa poklapa se sa imenom <wsdl:portType>

Klasa koja služi za instanciranje proxy-ja i za komunikaciju sa Web servisom

Imena paketa poklapaju se sa imenima targetNamespace iz WSDL fajla

generišu se metode u kojima se definiše biznis logika

U paketu se nalaze JAXB anotirani Java Bean-ovi koje se koriste za serijalizaciju objekata u XML fajl i obrnuto. Sve one nastale su na osnovu elemenata definisanih u XML šemi, predstavljaju omotače SOAP poruka

- Generisanje klasa na klijentskoj strani po sličnom principu.
- Sledeći korak je implementacija biznis logike.

Objavljivanje servisa

- Objaviti servis pomoću CXF implementacije `javax.xml.ws.Endpoint` API-ja.
- Objaviti servis na aplikativni server (Tomcat). Konfiguracija servisa u ovom slučaju je rađena pomoću Spring frejmvorka.
- Spring je open source frejmvork nastao sa idejom da olakša pisanje enterprise aplikacija.
- Spring nudi mnogo mogućnosti kao što su:
  1. Podrška za rad sa bazama podataka (JDBC)
  2. Podrška za rad sa ORM frejmovrcima (Hibernate,...)
  3. Podrška za rad sa transakcijama.
  4. AOP

Objavljvanje servisa
Spring frejmvork-Dependency Injection

- U svakoj složenijoj aplikaciji, objekat komunicira sa više drugih objekata.
- Potrebno je da objekat ima reference na druge objekte.
- Ako je objekat sam zadužen za postavljanje referenci dobija se kod sa puno zavisnosti, težak za testiranje.
- Spring omogućava da se zavisnosti inject-uju u objekat kada se on instancira pomoću Spring klasa (Spring kontejneri).
- Objekat treba da ima reference na interfejse a ne na konkretne implementacije.
- Konfigurisanje zavisnosti kroz XML.
- Spring kontejner parsira XML.
- Kada je potreban objekat traži se od Spring kontejnera i dobija sa svim postavljenim zavisnostima.
- Spring omogućava inject-ovanje kolekcija (liste,mape).

Apache CXF
Objavljivanje servisa

beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jaxws="http://cxf.apache.org/jaxws"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd">

  <import resource="classpath:META-INF/cxf/cxf.xml" />
  <import resource="classpath:META-INF/cxf/cxf-extension-
soap.xml" />
  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

  <jaxws:endpoint id="studentList" implementor="#StudentListImpl"
  address="/StudentList" />

  <bean id="StudentListImpl"
class="studentservice.StudentListImpl">
    <property name="daoStudent" ref="daoStudentDummy"/>
  </bean>

  <bean id="daoStudentDummy" class="dao.DAOStudentDummy"/>
</beans>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>WritingServiceWithSpring</display-name>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/beans.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
   org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <servlet>
    <servlet-name>CXFServlet</servlet-name>
    <servlet-class>
     org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXFServlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Testiranje servisa

- Konfiguracija klijentske strane slično, kroz Spring.
- Prilikom testiranja korišćeni su dodatni alati (soapUI, TCPMonitor).

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
<getStudentList xmlns="http://studentService/types">
 <year>1</year>
 </getStudentList>
 </S:Body>
</S:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <getStudentListResponse xmlns="http://studentService/types">
   <return>
    <studentName>Lazar Lazarević</studentName>
    <studentID>2008/3333</studentID>
    <department>RTI</department>
   </return>
   <return>
    <studentName>Petar Petrović</studentName>
    <studentID>2007/0433</studentID>
    <department>ETA</department>
   </return>
  </getStudentListResponse>
 </soap:Body>
</soap:Envelope>
```

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva(ms)
100	7
1000	16
10000	157
20000	301
30000	387
50000	633

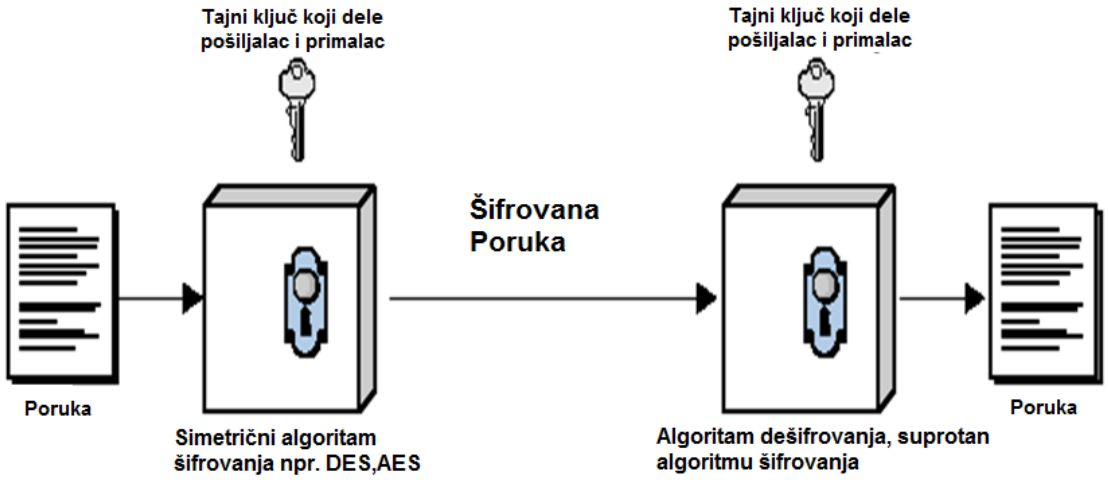
Zaštita SOAP poruka (WS-Security)
Osnovni pojmovi

- Osnovni pojmovi iz zaštite podataka:

1. *plaintext (otvoreni tekst)* - originalna poruka koja se šalje,
2. *ciphertext (šifrovana poruka)* - kodirana poruka,
3. *cipher (šifra)* - algoritam transformacije originalne u kodiranu poruku,
4. *key (ključ)* - informacija korišćena u šifri, poznata samo pošiljaocu/primaocu,
5. *encipher (encrypt) [šifrovanje (kriptovanje)]* - konverzija originalne poruke u kodiranu,
6. *decipher (decrypt) [dešifrovanje (dekriptovanje)]* - obnavljanje originalne poruke iz kodirane,
7. *cryptography (kriptografija)* - nauka o metodama i principima šifrovanja.

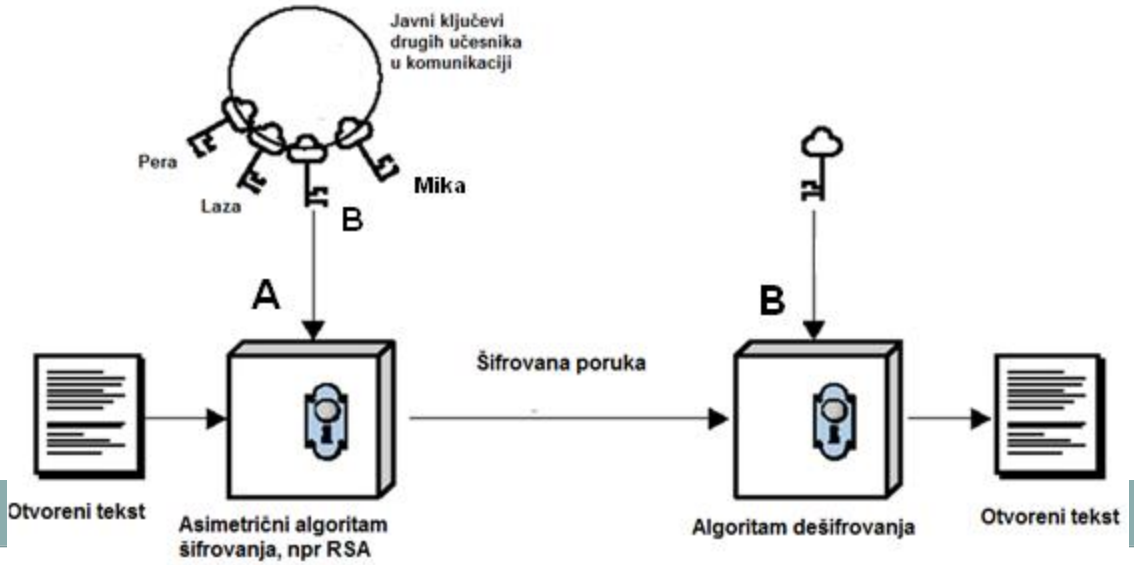
Zaštita SOAP poruka (WS-Security)
Simetrični algoritmi

- Simetrični ili klasični algoritmi šifrovanja.
- Obezbeđuju tajnost poruke.
- Postoji tajni ključ koji dele primalac i pošiljalac.
- Najpoznatiji algoritmi DES, TripleDES, AES.



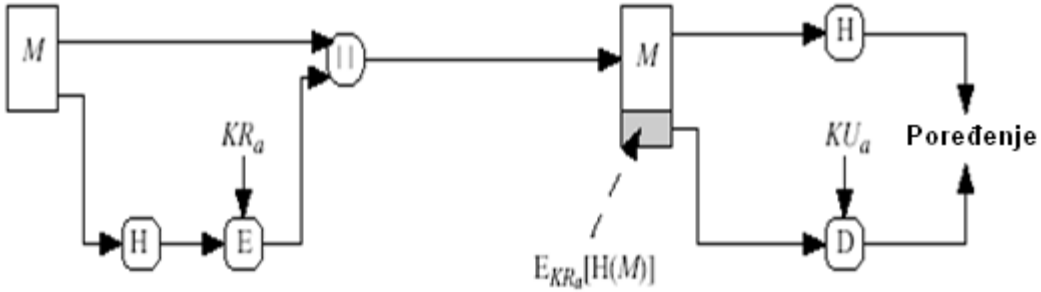
Zaštita SOAP poruka (WS-Security)
Asimetrični algoritmi

- Asimetrični algortimi, najveći napredak u istoriji kriptografije (RSA).
- Svaki učesnik u komunikaciji ima par ključeva, javni i tajni.
- Javni ključ je dostupan svakome, tajni samo vlasniku.
- Poruka šifrovana javnim ključem može da se dešifruje samo tajnim (obezbeđuje se tajnost).
- Poruka šifrovana tajnim ključem može da se dešifruje javnim (autentikacija , integritet poruke).
- Tajnost:



Zaštita SOAP poruka (WS-Security)
Autentikacija i Integritet

- Šifrovanje tajnim ključem naziva se digitalno potpisivanje.
- Svako ko ima javni ključ može da proveri potpis i autentikuje pošiljaoca.
- Obično se potpisuje samo deo poruke.
- Izračuna se heš funkcija nad određenim delom poruke (SHA1).
- Potpiše se (šifruje) tajnim ključem heš vrednost.
- Ovime se pored autentikacije obezbeđuje i integritet poslate poruke.
- Integritet je obezbeđen jer su heš algoritmi tako osmišljeni da se teško dobije ista heš vrednost za različite ulazne podatke.



Zaštita SOAP poruka (WS-Security)
CA

- Učesnik u komunikaciji generiše par javni/tajni ključ i objavljuje javni ključ.
- Kako verovati da je učesnik u komunikaciji zaista onaj za koga se predstavlja?
- U komunikaciju se uvodi proverena treća strana (TTP) ili CA (Certificate Authority) koja izdaje digitalne sertifikate.
- Digitalni sertifikat – garancija od strane CA za javni ključ učesnika u komunikaciji.
- Učesnik u komunikaciji generiše samopotpisani sertifikat i pošalje ga CA-u.
- CA potpiše svojim tajnim ključem sertifikat.
- Svi koji veruju tom CA-u (poseduju njegov javni ključ) mogu da provere sertifikat i utvrde validnost ključa.
- X509 standard za sertifikate.

Apache CXF
Zaštita SOAP poruka (WS-Security)

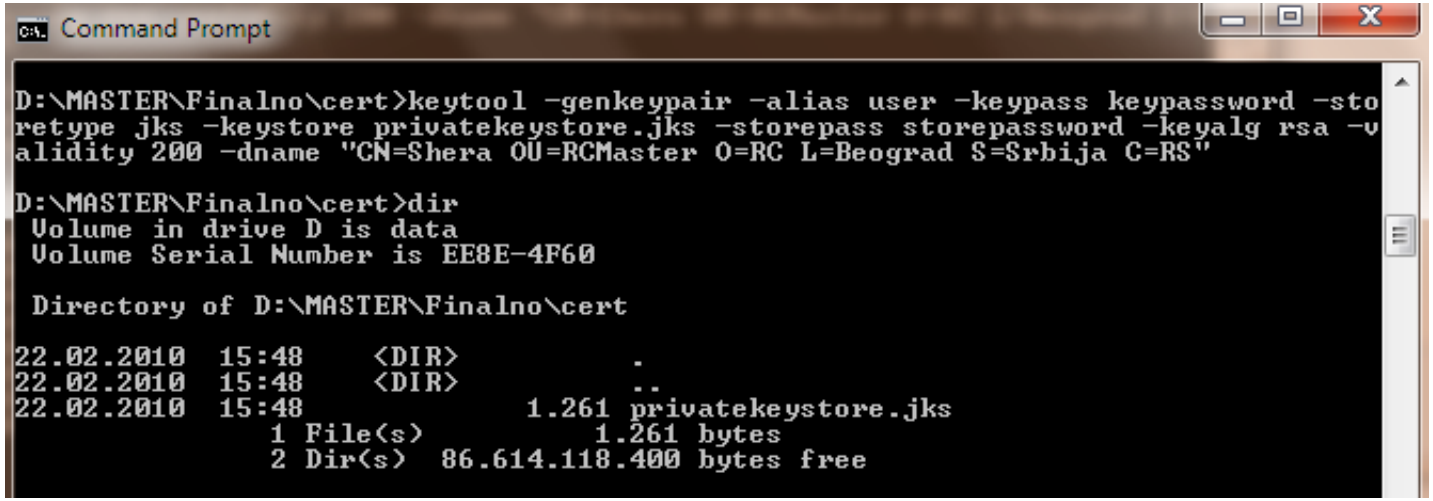
- Krajnji ciljevi implementacije vezani za zaštitu podataka:
- Klijent koji šalje poruku je prvo potpisuje. Potpisuje se neki deo poruke, npr. body deo poruke ili body i timestamp. Za računanje heša može da se koristi SHA-1 algoritam, a za šifrovanje RSA algoritam tajnim ključem klijenta. Potpis se ugrađuje u header deo poruke.
- Generiše se ključ sesije i njime se šifruje poruka. U našim primerima uvek ćemo šifrovati samo sadržaj poruke tj. body deo. Koristićemo TripleDES ili AES simetrični algoritam šifrovanja.
- Ključ sesije se šifruje javnim ključem servera, npr. RSA algoritmom, i ugrađuje u header poruke.
- U header deo poruke se ubacuje timestamp. Ako želimo da potpisujemo timestamp onda se on ubaci u poruku pre potpisivanja.
- Ovako zaštićena poruka šalje se serveru.
- Na serverskoj strani dešifruje se ključ sesije tajnim ključem servera i zatim dešifruje sadržaj poruke. Proverava se digitalni potpis tako što se dešifruje potpis javnim ključem klijenta i dobije heš vrednost koja se poredi sa ponovo sračunatim heš-om. Proverava se timestamp.
- Priprema se odgovor na isti način kao što je to radio kljient i šalje.
- Klijent proverava poruku koja je stigla.

Zaštita SOAP poruka (WS-Security)
Implementacija

- Apache CXF implementira WSS4J API.
- WSS4J je Javina biblioteka za koja se koristi za zaštitu SOAP poruka.
- Za zaštitu poruka i dodavanje svih potrebnih informacija u header deo SOAP poruke koriste se WSS4J interceptor-i.
- Konfiguracija interceptora može da se radi programski ili kroz Spring.
- Mogu se primeniti dva koncepta:
  1. Autentikacija slanjem korisničkog imena i šire (UsernameToken, za testiranje)
  2. Obostrano digitalno potpisivanje i šifrovanje SOAP poruka

Zaštita SOAP poruka (WS-Security)
Implementacija

- Pre konfiguracije interceptora potrebno je generisati parove javni/tajni ključ.
- Koristi se Javin keytool alat (sastavni deo JDK-a).
- Potrebno je generisati parove javni/tajni ključ, eksportovati javni ključ u sertifikat i instalirati ga na klijentskoj-serverskoj strani.
- Korišćeni samopotpisani sertifikati.



Zaštita SOAP poruka (WS-Security) Implementacija

- Konfiguracija interceptor-a za klijenta kroz Spring.

```
<jaxws:client id="studentListClient"
 serviceClass="studentservice.StudentList"
address="http://localhost:8080/CXFSignEncrSpringServer/StudentList">

  <jaxws:outInterceptors>
    <bean
    class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor">
      <constructor-arg>
        <map>
          <entry key="action" value="Signature Encrypt Timestamp" />

          <entry key="signaturePropFile"value="client_sign.properties" />
          <entry key="user" value="user"/>
          <entry key="signatureKeyIdentifier" value="IssuerSerial"/>
          <entry key="passwordCallbackRef">
            <ref bean="myPasswordCallback" />
          </entry>

          <entry key="encryptionPropFile" value="client_enc.properties" />
          <entry key="encryptionUser" value="serverShera" />
          <entry key="encryptionKeyIdentifier"value="IssuerSerial" />
        </map>
      </constructor-arg>
    </bean>
  </jaxws:outInterceptors>
```

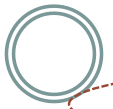
```
<jaxws:inInterceptors>
    <bean
    class="org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor">
      <constructor-arg>
        <map>
          <entry key="action" value="Signature Encrypt Timestamp" />

          <entry key="signaturePropFile" value="client_enc.properties"/>
          <entry key="signatureKeyIdentifier" value="IssuerSerial"/>

          <entry key="decryptionPropFile" value="client_sign.properties"/>
          <entry key="encryptionKeyIdentifier"value="IssuerSerial" />
          <entry key="passwordCallbackRef">
          <ref bean="myPasswordCallback" />
          </entry>
        </map>
      </constructor-arg>
    </bean>
  </jaxws:inInterceptors>
</jaxws:client>

  <bean id="myPasswordCallback"
  class="handlers.ClientCallbackHandler" />
```

Zaštita SOAP poruka (WS-Security)
Testiranje



```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
    1.0.xsd" soap:mustUnderstand="1">

     <wsu:Timestamp xmlns:wsu="http://docs.oasis-
  open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
  1.0.xsd" wsu:Id="Timestamp-4">
    <wsu:Created>2010-02-23T11:05:38.206Z</wsu:Created>
    <wsu:Expires>2010-02-23T11:10:38.206Z</wsu:Expires>
  </wsu:Timestamp>

  <xenc:EncryptedKey xmlns:xenc=http://www.w3.org/2001/04/xmlenc#
  Id="EncKeyId-EDBF37562A75AB88AA12669231381715">
    <xenc:EncryptionMethod Algorithm=
    "http://www.w3.org/2001/04/xmlenc#rsa-1_5" />

  <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
    1.0.xsd">
          <ds:X509Data>
    <ds:X509IssuerSerial>
            <ds:X509IssuerName>CN=SheraServis OU\=RCMaster O\=RC
            L\=Beograd S\=Srbija C\=RS</ds:X509IssuerName>
        <ds:X509SerialNumber>1266851056</ds:X509SerialNumber>
    </ds:X509IssuerSerial>
  </ds:X509Data>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>

    <xenc:CipherData>
     <xenc:CipherValue>XfXpV22miVm6cuhqDLhKZx7LLWEM4bRWCjjA3lHE
     68/IeNfe3c/4ckLrXuZESLYS2wJfJGQ2JzCblfKvKax6tyfg/JasJVNh8TW
     b4fPOE8eaKe1pS9ZWKDOBa+vUL26rvp7dvovbrFhilSGnwAZm9d9VpMmSgL
     sg3MhdHVeWAr0=</xenc:CipherValue>
       </xenc:CipherData>
       <xenc:ReferenceList>
        <xenc:DataReference URI="#EncDataId-3" />
       </xenc:ReferenceList>
     </xenc:EncryptedKey>
```

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-1">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm=
    "http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod Algorithm=
        "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <ds:Reference URI="#id-2">
          <ds:Transforms>
   <ds:Transform Algorithm=
   "http://www.w3.org/2001/10/xml-exc-c14n#" />
  </ds:Transforms>
  <ds:DigestMethod Algorithm=
  "http://www.w3.org/2000/09/xmldsig#sha1" />
  <ds:DigestValue>i2HBEgaW40flpW9OPEFrsX7esUk=
  </ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>

      <ds:SignatureValue>nA7YMNO3Hyp4g6s5K2SeEUPhjmWXeuEmJ7RBRCL8
   +t8I3tptlPZu1e0Jw3SqK2kYBM3hzIUcmJP8Bma6F6oPJ6IqzVTgg+H5jKY
   Dg3dzXtNwbPoscDNCgc1wdxQTsMUtmnao3Nc0UXh1DcJTqWgIWFSqw+mFwu
   WX18E7Q7GC7K0=</ds:SignatureValue>

   <ds:KeyInfo Id="KeyId-EDBF37562A75AB88AA12669231379192">
    <wsse:SecurityTokenReference
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
    200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
    200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="STRId-
    EDBF37562A75AB88AA12669231379223">
          <ds:X509Data>
   <ds:X509IssuerSerial>
    <ds:X509IssuerName>CN=Shera OU\=RCMaster O\=RC
     L\=Beograd S\=Srbija C\=RS</ds:X509IssuerName>
        <ds:X509SerialNumber>1266849905
        </ds:X509SerialNumber>
   </ds:X509IssuerSerial>
  </ds:X509Data>
    </wsse:SecurityTokenReference>
   </ds:KeyInfo>
  </ds:Signature>
    </wsse:Security>
  </soap:Header>
```

Zaštita SOAP poruka (WS-Security)
Testiranje

```
<soap:Body xmlns:wsu="http://docs.oasis-
 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 wsu:Id="id-2">

  <xenc:EncryptedData
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EncDataId-0"
  Type="http://www.w3.org

<xenc:EncryptionMethod Al
"http://www.w3.org/2001/0

<ds:KeyInfo xmlns:ds="http
 <wsse:SecurityTokenRefer
 open.org/wss/2004/01/oas
 1.0.xsd">
  <wsse:Reference xmlns:w
  open.org/wss/2004/01/oa
  1.0.xsd" URI="#EncKeyId
  />
 </wsse:SecurityTokenRefe
</ds:KeyInfo>

<xenc:CipherData>
 <xenc:CipherValue>gtDLJ
 EOmcmjPMTczvr63lUYhkl
 dTEQWKkOKZKwFG1fMg
 gSA/gI7AMVoIlWIT9KSm
 7h43tRBJP8Mo6FGSpEVh
 8NhlDg8WQbs8TUG8Yzsv
 i/WxHX+DqcM2cD55CbC
 /SlPmr6Uxsg==</xenc:Ci
</xenc:CipherData>
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva
100	84ms
1000	250ms
10000	1.4s
20000	2.5s
30000	3.3s
50000	5.5s

Tabela 10. Prosečno vreme odziva metode `getStudentList` kada se obe strane potpisuju i šifruju poruku (AES)

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva
100	71ms
1000	187ms
10000	1.7s
20000	3.3s
30000	4.6s
50000	7.7s

Tabela 11. Prosečno vreme odziva metode `getStudentList` kada se obe strane potpisuju i šifruju poruku (TripleDES)

3.Interoperabilnost sa C#

- Testovi bez zaštite.
- Klijenske aplikacije – Windows Forms
- Automatsko generisanje klasa na dva načina:
  1. wsdl.exe alat (.net 1.1 pa nadalje) – Automatski se poziva dodavanjem Web Service Reference
  2. svcutil.exe (.net 3.0) – Automatski se poziva dodavanjem Service reference
- Generiše se proxy klasa za slanje zahteva.
- Generisane SOAP poruke potpuno razumljive Java serveru i C# klijentu.
- Rezultati testova performansi kada je klijent dobijen pomoću svcutil.exe.

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva(ms)
100	6.4
1000	14.3
10000	100
20000	214
30000	309
50000	512

Tabela 17. Prosečno vreme odziva metode `getStudentList` za različite količine podataka, C# klijent - CXF server

Interoperabilnost sa C# - Zaštita podataka

- Zaštita SOAP poruka rešena pomoću dve tehnologije:
  1. WSE 3.0 (Web Service Enhacement)
  2. WCF (Windows Communication Foundation), podrazumevano kada se koristi svcutil.exe
- Za generisanje parova javni tajni ključ makecert alat.
- Konfiguracija potpuno deklarativna, kroz XML.
- Obe tehnologije nude vizuelne editore za konfiguraciju.
- Potrebno dosta vremena da se usklade Java – C# standardi, naročito zbog nedostatka literature.
- Npr. WCF nudi više klasa za WS-Security mod ali se koristi SOAP 1.2 protokol.
- Java servisi implementirani pomoću CXF su fleksibilniji (npr. kod podrazumevanog potpisivanja.)
- Šta ako se uvede i npr. PHP klijent?

Dodatni testovi

- Definisane nove usluge Web servisa kroz WSDL fajl.
- Testirano prenošenje datetime podataka, null vrednosti...

Form1

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva()
100	32.3 ms
1000	253.9ms
10000	2.4 s
20000	4.9 s
30000	7.3 s
50000	12.2 s

Tabela 20. Prosečno vreme odziva metode `testMethod` za različite količine podataka, C# klijent - CXF server

Broj studenata koji se prenosi SOAP porukom	Prosečno vreme odziva()
100	11 ms
1000	25.6ms
10000	206.8 ms
20000	398 ms
30000	591ms
50000	978.6 ms

Tabela 21. Prosečno vreme odziva metode `testMethod` za različite količine podataka ali uz instanciranje samo jednog datuma na serveru, C# klijent - CXF server