Java i web servisi

JAX*

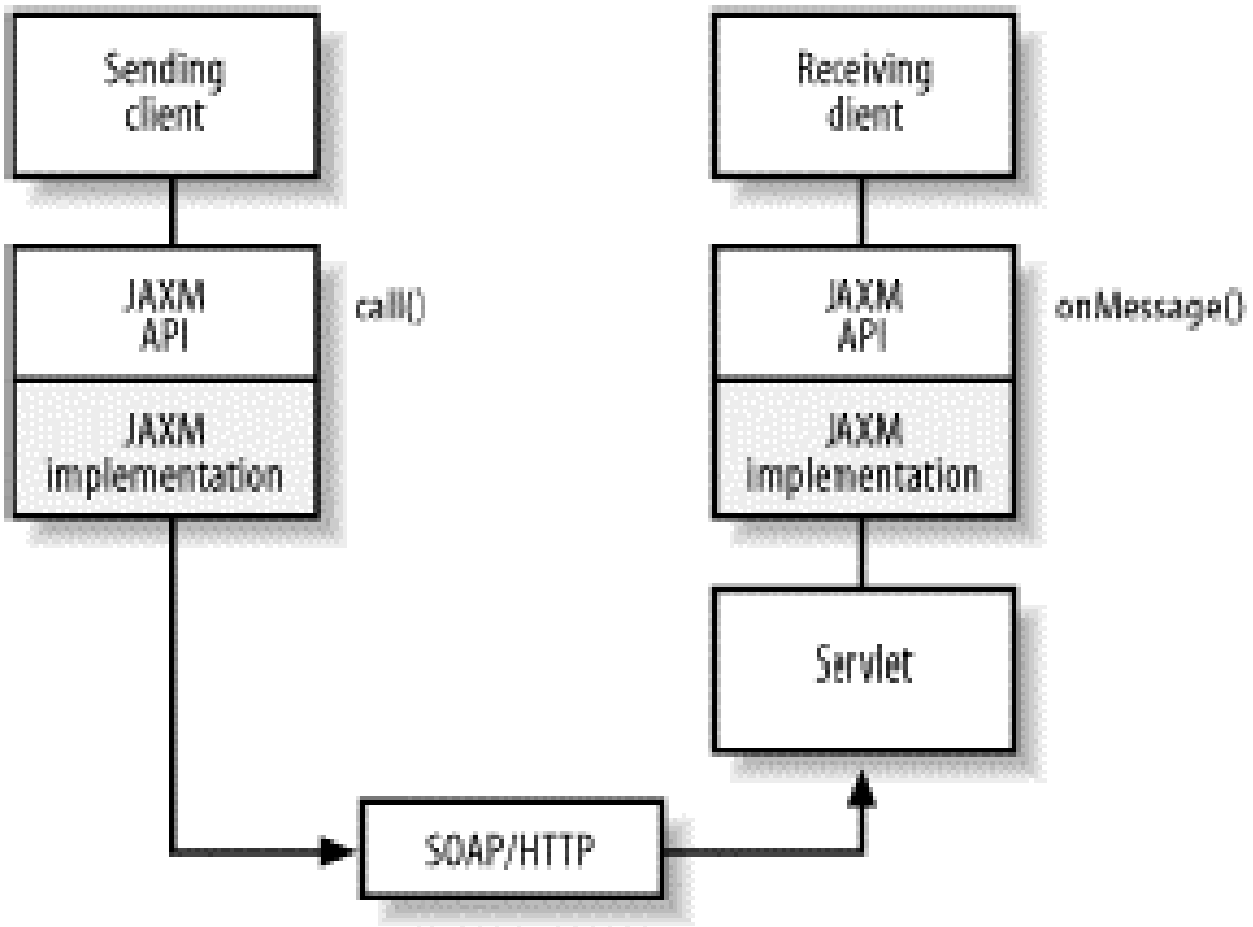
- web servisi su zasnovani na wire-level specifikacijama
- pristup web servisima iz Java okruženja je standardizovan pomoću nekoliko API-level specifikacija
  - JAXM = Java API for XML Messaging
    - pozivanje document-style web servisa
  - JAX-RPC = Java API for XML-based RPC
    - pozivanje RPC-style web servisa
    - deprecated, zamenio ga JAX-WS
  - JAXR = Java API for XML Registries
    - pristup UDDI registrima
  - JAX-WS = Java API for XML Web Services
    - pozivanje web servisa, koristi anotacije
  - JAXB = Java API for XML Binding
    - mapiranje podataka Java ↔ XML

JAXM

- razmena SOAP poruka preko različitih transportnih protokola
- direktno manipulisanje sadržajem SOAP poruka
- podržava sinhronu komunikaciju
- podržava asinhronu komunikaciju samo u okviru servlet ili EJB kontejnera
- paket **`javax.xml.soap`** – formiranje i čitanje SOAP poruka
- paket **`javax.xml.messaging`** – slanje i prijem poruka

JAXM

- slanje i prijem poruka



JAXM – formiranje poruka

```
SOAPMessageFactory smf = SOAPMessageFactory.newInstance();
SOAPMessage msg = smf.createMessage();
SOAPPart soapPart = msg.getSOAPPart();
SOAPEnvelope env = soapPart.getEnvelope();
SOAPBody body = env.getBody();

Name name = envelope.createName("GetLastTradePrice", "m",
   "http://www.ztrade.com");
SOAPBodyElement bodyElement = body.addBodyElement(name);
bodyElement.addTextNode("SUNW");
---------------
<soap:Envelope>
  <soap:Body>
    <m:GetLastTradePrice xmlns:m="http://www.ztrade.com">
      SUNW
    </m:GetLastTradePrice>
  </soap:Body>
</soap:Envelope>
```

JAXM – slanje poruka

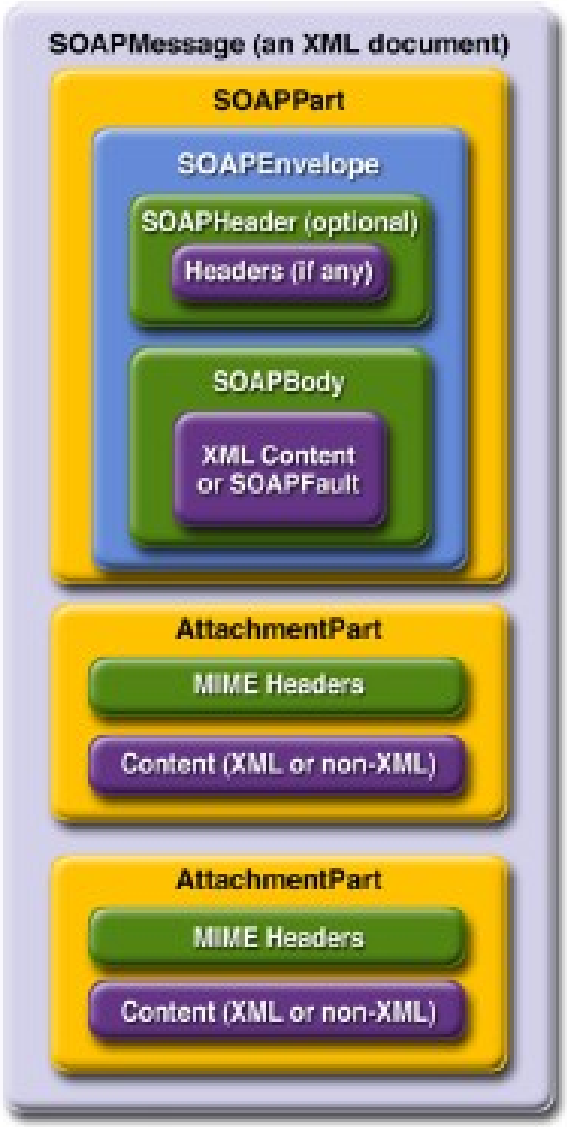
```
SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
SOAPConnection conn = scf.createConnection();

SOAPMessage reply = conn.call(new
   URLEndpoint("http://test.ns.ac.yu/Test"));

conn.close();
```

JAXM – prijem poruka

```
public class SimpleJAXMReceive extends JAXMServlet implements ReqRespListener {
  private MessageFactory mf = null;

  public void init(ServletConfig servletConfig) throws ServletException {
    super.init(servletConfig);
    mf = MessageFactory.newInstance();
  }

  public SOAPMessage onMessage(SOAPMessage message) {
    // ispiši primljenu poruku na konzolu
    message.writeTo(System.out);
    // konstruiši odgovor i pošalji ga
    SOAPMessage msg = mf.createMessage( );
    SOAPPart part = msg.getSOAPPart( );
    SOAPEnvelope env = part.getEnvelope( );
    SOAPBody body = env.getBody( );
    Name name = env.createName("Response");
    SOAPBodyElement bodyElement = body.addBodyElement (name);
    bodyElement.addTextNode ("Text response");
    return msg;
  }
}
```

SAAJ – SOAP with Attachments API for Java

- SOAP poruke mogu imati zakačene fajlove (attachments) ... ☹
- JAXM → SAAJ



SAAJ

```
// dodavanje atačmenta u poruku
AttachmentPart attachment = msg.createAttachmentPart();
String content = "Ovo je tekst atačmenta";
attachment.setContent(content, "text/plain");
attachment.setContentId("description");
msg.addAttachmentPart(attachment);


// čitanje atačmenta iz poruke
Iterator it =  msg.getAttachments();
while (it.hasNext()) {
  AttachmentPart att = (AttachmentPart)it.next();
  String id = att.getContentId();
  String type = att.getContentType();
  Object content = att.getContent();
}
```

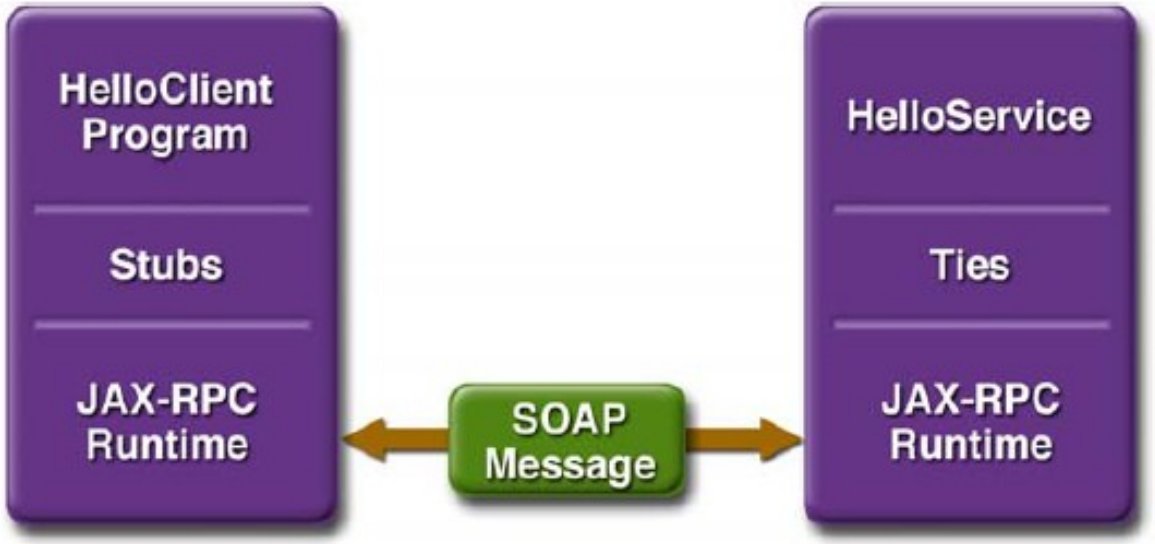
SAAJ

- primer poruke

```
--2023334682.1010158929328.JavaMail.chappell.nbchappell3
Content-Type: text/xml

<soap-env:Envelope
xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Header/><soap-env:Body><PurchaseOrder>
<shipTo country="US">
<name>Joe Smith</name>
<street>14 Oak Park</street><city>Bedford</city><state>MA</state>
<zip>01730</zip></shipTo><items><item partNum="872-AA">
<productName>Candy Canes</productName>
<quantity>444</quantity><price>1.68</price>
<comment>I want candy!</comment></item></items></PurchaseOrder>
</soap-env:Body></soap-env:Envelope>
--2023334682.1010158929328.JavaMail.chappell.nbchappell3
Content-Type: text/plain

This is an attachment.
--2023334682.1010158929328.JavaMail.chappell.nbchappell3
Content-Type: text/plain; charset=ISO-8859-1

Another attachment.
--2023334682.1010158929328.JavaMail.chappell.nbchappell3--
```

JAX-WS

- JAXM/SAAJ predstavlja API nižeg nivoa
  - omogućava direktan pristup sadržaju SOAP poruke
  - omogućava rad sa svim tipovima servisa (one-way, request-response, solicit-response, notify)
- JAX-WS je API nešto višeg nivoa
  - za mapiranje WSDL ↔ Java oslanja se na JAXB
  - rukovanje SOAP Fault mehanizmom preko Java izuzetaka



JAX-WS

- implementacija web servisa
  - napisati Java klasu
  - označiti je @WebService anotacijom
  - može se dodati *service endpoint interface* (SEI)
    - slično kao remote interfejs
  - metode koje su dostupne spolja označiti @WebMethod anotacijom
    - parametre i rezultat opisati JAXB anotacijama
  - pomoću wsgen može se generisati WSDL, stub, itd.
  - upakovati u .war

JAX-WS primer – endpoint interface

```
@WebService(
  targetNamespace="http://hello.com",
  wsdlLocation="WEB-INF/wsdl/hello.wsdl")
public interface HelloPort {

  @WebMethod
  @WebResult(
    name="Greeting",
    targetNamespace="http://hello.com")
  @RequestWrapper(
    localName="Hello",
    targetNamespace="http://hello.com"
    className="com.hello.HelloRequest")
  @ResponseWrapper(
    localName="HelloResponse",
    targetNamespace="http://hello.com"
    className="com.hello.HelloResponse")
  public String sayHello(
    @WebParam(name="name", targetNamespace="http://hello.com")
    String name);
}
```

JAX-WS primer – SOAP zahtev i odgovor

```
<soap:Body>
  <h:Hello xmlns:h="http://hello.com">
    <h:name>Mitar</h:name>
  </h:Hello>
</soap:Body>
```

```
<soap:Body>
  <h:HelloResponse xmlns:h="http://hello.com">
    <h:Greeting>Hello, Mitar.</h:Greeting>
  </h:HelloResponse>
</soap:Body>
```

JAX-WS primer – implementaciona klasa

```
@WebService(
  targetNamespace="http://hello.com",
  endpointInterface="com.hello.examples.HelloService")
public class Hello implements HelloPort {

  public void Hello() {}

  public String sayHello(String name) {
    return "Hello, " + name + ".";
  }
}
```

JAX-WS primer – klijent sa dependency injection

```
@WebServiceRef(HelloService.class)
private HelloPort helloPort;
...
String greeting = helloPort.sayHello("Mitar");

// 1) dependency injection radi samo u JEE kontejneru
// 2) HelloService je generisana klasa
```

JAX-WS primer – klijent sa generisanom Service klasom

```
HelloService helloService = new HelloService();
HelloPort helloPort = helloService.getHelloPort();

String greeting = helloPort.sayHello("Mitar");

// može da radi i izvan JEE kontejnera
```

JAX-WS primer – klijent sa dinamički kreiranom Svc klasom

```
URL wsdlUrl = new URL("http://localhost:8080/hello/Hello?wsdl");
QName svcName = new QName("http://hello.com", "HelloService");
QName portName = new QName("http://hello.com", "HelloPort");

Service svc = Service.create(wsdlUrl, svcName);
HelloPort helloPort = (HelloPort)svc.getPort(
    portName, HelloPort.class);

String greeting = helloPort.sayHello("Mitar");

// Na klijentu je i dalje potreban SEI ali nisu potrebne
// generisane klase
```

JAXB – konverzija podataka Java ↔ XML

- zasnovana na anotacijama
- postoje defaults

JAXB primer

```
// za mapiranje se koriste atributi, a ne setter/getter metode
@XmlAccessorType(AccessType.FIELD)

// anonimni složeni tip, definisan redosled propertija
@XmlType(name="", propOrder={"billTo","items"})

// naziv elementa za objekte ove klase
@XmlRootElement(name="simpleOrder")
public class SimpleOrder {

  // mapira se na element iz šeme
  @XmlElement(namespace="http://www.example.com/oms")
  protected BillTo billTo;

  // mapira se na element iz šeme
  @XmlElement(namespace="http://www.example.com/oms")
  protected Items items;
  ...
```

JAXB primer

```
@XmlAccessorType(AccessType.FIELD)
@XmlType(name="", propOrder={"name","street","city","state",
  "zip","phone"})
public static class BillTo {
  @XmlElement(namespace="http://www.example.com/oms",
    required=true)
  protected String name;

  @XmlElement(namespace="http://www.example.com/oms")
  protected String street;

  @XmlElement(namespace="http://www.example.com/oms")
  protected String city;

  @XmlElement(namespace="http://www.example.com/oms")
  protected String state;

  @XmlElement(namespace="http://www.example.com/oms")
  protected String zip;

  @XmlElement(namespace="http://www.example.com/oms")
  protected String phone;
```

JAX-WS web servisi bez JAXB mapiranja

- koristi se @**WebServiceProvider** anotacija
- implementira se interfejs **javax.xml.ws.Provider<T>**
  - metoda **T invoke(T request)**
- može da radi u dva režima:
  - message: dobija se kompletna SOAP poruka
  - message payload: dobija se sadržaj SOAP Body

JAX-WS web servisi bez JAXB mapiranja

```
@WebServiceProvider(
  serviceName="RequestOrderService",
  portName="RequestOrderPort",
  targetNamespace="http://www.example.com/req",
  wsdlLocation="WEB-INF/wsdl/RequestOrder.wsdl")
@ServiceMode(Service.Mode.PAYLOAD)
public class RequestOrderEndpoint implements Provider<Source> {

  @Resource
  WebServiceContext webServiceContext;

  public Source invoke(Source payload) {...}
}
```

JAXR

- API-level specifikacija za pristup registrima web servisa
  - UDDI registri
  - ebXML registri
- paket **`javax.xml.registry`**

JAXR

- standalone JAXR klijent

```
Context ctx = new InitialContext();
ConnectionFactory cf = ConnectionFactory.newInstance();

Properties props = new Properties();
// adrese gde se nalaze UDDI servisi
props.setProperty("javax.xml.registry.queryManagerURL",
  "http://uddi.ibm.com/testregistry/inquiryapi");
props.setProperty("javax.xml.registry.lifeCycleManagerURL",
  "http://uddi.ibm.com/testregistry/publishapi");

// adresa mog HTTP i HTTPS proxy-ja
props.setProperty("com.sun.xml.registry.http.proxyHost", "proxy.uns.ac.rs");
props.setProperty("com.sun.xml.registry.http.proxyPort", "8080");
props.setProperty("com.sun.xml.registry.https.proxyHost", "proxy.uns.ac.rs");
props.setProperty("com.sun.xml.registry.https.proxyPort", "8080");

// otvori konekciju
cf.setProperties(props);
Connection conn = cf.createConnection();
```

JAXR

- standalone JAXR klijent

```
// reference na servise
RegistryService rs = conn.getRegistryService();
BusinessQueryManager bqm = rs.getBusinessQueryManager();
BusinessLifeCycleManager blm = rs.getBusinessLifeCycleManager();

// parametri upita
Collection findQualifiers = new ArrayList();
findQualifiers.add(FindQualifier.SORT_BY_NAME_DESC);
findQualifiers.add(FindQualifier.CASE_INSENSITIVE_MATCH);
Collection namePatterns = new ArrayList();
namePatterns.add("%trade%");

// postavi upit
BulkResponse response = bqm.findOrganizations(findQualifiers, namePatterns,
  null, null, null, null);
Collection orgs = response.getCollection();
```