Putting Services into Practice

Massimo Mecella
mecella@dis.uniroma1.it
http://www.dis.uniroma1.it/~mecella/ricevimento.htm

Dipartimento di Informatica
e Sistemistica Antonio Ruberti
Sapienza
Università di Roma

XML BASICS

Extensible Markup Language

Extensible Markup Language (XML):

- Describes data objects called XML documents
- Is composed of markup language for structuring data
- Supports custom tags for definition, transmission, validation, and interpretation of data
- Conforms to Standard Generalized Markup Language (SGML)
- Has become a standard way to

A Simple XML Page: Example

```
<?xml version="1.0"?>
<employees>
  <employee>
    <employee_id>120</employee_id>
    <last_name>Weiss</last_name>
    <salary>8000</salary>
  </employee>
  <employee>
    <employee_id>121</employee_id>
    <last_name>Fripp</last_name>
    <salary>8200</salary>
  </employee>
</employees>
```

XML Document Structure

An XML document contains the following parts:

1. Prologue
2. Root element
3. Epilogue

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!-- this is a comment -->
<employees>
 ...
</employees>

<?gifPlayer size="100,300" ?>
```

① ② ③

The XML Declaration

XML documents must start with an XML declaration.

The XML declaration:

- Looks like a processing instruction with the `xml` name. For example:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<document-root>
...
</document-root>
```

- Must contain the `version` attribute
- May (optionally) include:
  - The `encoding` attribute
  - The `standalone` attribute
- Is optional in XML 1.0, but mandatory in XML 1.1

Components of an XML Document

XML documents comprise storage units containing:

- Parsed data, including the:
  - Markup (elements, attributes, and entities) used to describe the data they contain
  - Character data described by markup

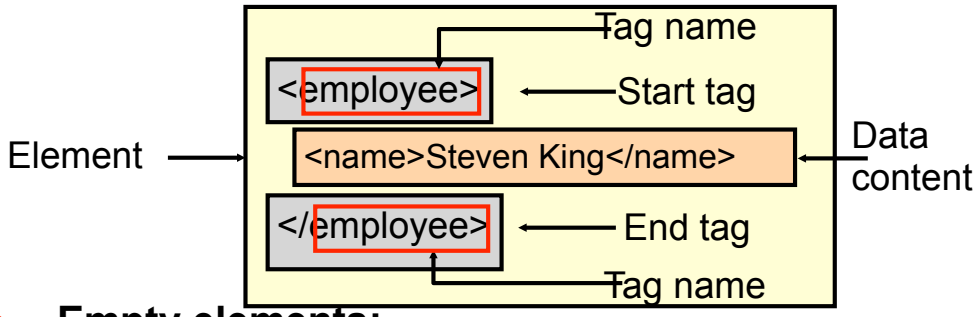
```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<employees>
  <employee id="100">
    <name>Rachael O&apos;Leary</name>
  </employee>
</employees>
```

- Unparsed data, such as textual or binary information (graphic and sound data), is left as entered.

```
<![CDATA[  ...unparsed data...  ]]>
```


XML Elements

- An XML element has:
  - A start tag, end tag, and optional data content
  - Case-sensitive tags (start and end tags must match)



- Empty elements:
  - Do not contain any data
  - May appear as a single tag

```
<initials></initial>
```

```
<initials/>
```

Markup Rules for Elements

- There is one root element, sometimes called the top-level or document element.
- All elements:
  - Must have matching start and end tags, or be a self-closing tag (that is, an empty element)
  - Can contain nested elements such that their tags do not overlap
  - Have case-sensitive tag names subject to naming conventions (that is, they must start with a letter, contain no spaces, and not start with the letters `xml`)
  - May contain white space (spaces, tabs, new lines, and combinations of them) that is considered part of the element data content

XML Attributes

An XML attribute is a name-value pair that:

- Is specified in the start tag after the tag name

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<employees>
  <employee id="100" name='Rachael O&apos;Leary'>
    <salary>1000</salary>
  </employee>
</employees>
```

- Has a case-sensitive name
- Has a case-sensitive value that must be enclosed in matching single or double quotation marks
- Provides additional information about the XML document or XML elements

Using Elements Versus Attributes

```
<?xml version="1.0"?>
<employees>
  <employee>
    <id>100</id>
    <last_name>King</last_name>
    <salary>24000</salary>
  </employee>
</employees>
```

1 Elements

```
<?xml version="1.0"?>
<employees>
  <employee id="100" last_name="King"
            salary="24000">
    <job>President</job>
  </employee>
</employees>
```

2 Attributes

XML Entities

An XML entity:

- **Is a unit of data storage**
- **Is identified by a case-sensitive name**
- **Is used as replacement text (substitute) when referencing its name between an ampersand (`&`), and a semicolon (`;`)**

```
<comment>Salaries must not be &lt; 1000</comment>
```

- **Has predefined names for special XML characters:**
  - `<` **for less than (<), and** `>` **for greater than (>)**
  - `&` **for ampersand (`&`)**
  - `"` **for double quotation mark (`"`)**
  - `'` **for single quotation mark (`'`)**

XML Comments

XML comments:

- Start with <!--
- End with -->
- May appear anywhere in the character data of a document, and before the root element
- Are not elements, and can occupy multiple lines
- May not appear inside a tag or another comment

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!-- Comment: This document has information about
     employees in the company -->
<employees>
 <name>Steven King</name> <!-- Full name -->
</employees>
```


A Well-Formed XML Document

Every XML document must be well-formed, such that:

- An XML document must have one root element
- An element must have matching start and end tag names, unless they are empty elements
- Elements can be nested, but cannot overlap
- All attribute values must be quoted
- Attribute names must be unique in the start tag of an element
- Comments and processing instructions do not appear inside tags
- The < or & special characters cannot appear in the character data of an element or attribute value

Comparing XML and HTML

- XML
  - Is a markup language for describing data
  - Contains user-defined markup elements
  - Is extensible
  - Is displayed as a document list in a Web browser
  - Conforms to rules for a well-formed document
- HTML
  - Is a markup language for formatting data in a Web browser
  - Contains predefined markup tags
  - Is not extensible
  - Does not conform to well-formed document rules

XML Development

XML documents can be developed by using:

- A simple text editor, such as Notepad
- A specialized XML Editor, such as XMLSpy

What Is a Document Type Definition?

A document type definition (DTD):

- Is the grammar for an XML document
- Contains the definitions of
  - Elements
  - Attributes
  - Entities
  - Notations
- Contains specific instructions that the XML parser interprets to check the document validity
- May be stored in a separate file (external)
- May be included within the document (internal)

Why Validate an XML Document

- Well-formed documents satisfy XML syntax rules, and not the business requirements of content and structure.
- Business rules often require validation of the content and structure of a document.
- XML documents must satisfy structural requirements imposed by the business model.
- A valid XML document can be reliably processed by XML applications.
- Validations can be performed by using a DTD or an

General DTD Rules

A DTD:

- Must provide a declaration for items used in an XML document, such as:
  - Elements
  - Attributes
  - Entities
- Is case-sensitive, but spacing and indentation are not significant
- May use XML comment syntax for documentation, but comments cannot appear inside declarations

The Contents of a DTD

A DTD contains declarations (that use the syntax shown) for:

- Elements:

```
<!ELEMENT element-name content-model>
```

- Attributes:

```
<!ATTLIST element-name attrib-name type default>
```

- Entities:

```
<!ENTITY entity-name "replacement text">
```

- Notations:

```
<!NOTATION notation_name SYSTEM "text">
```

Simple DTD Declaration: Example

Example of a simple DTD with element declarations:

```
<!ELEMENT employees (employee)>
<!ELEMENT employee (name)>
<!ELEMENT name (#PCDATA)>
```

A valid XML document based on the DTD:

```
<?xml version="1.0"?>
<employees>
  <employee>
    <name>Steven King</name>
  </employee>
</employees>
```

Note: All child elements must be defined.

Referencing the DTD

The XML document references the DTD:

- After the XML declaration and before the root, by using:

```
<!DOCTYPE employees [ ... ]>
```

- Externally with the `SYSTEM` or `PUBLIC` keywords:

```
<!DOCTYPE employees SYSTEM "employees.dtd">
```

```
<!DOCTYPE employees PUBLIC "-//formal-public-ID">
```

- Internally in the `<!DOCTYPE root [...]>` entry:

```
<?xml version="1.0"?>
<!DOCTYPE employees [
 <!ELEMENT employees (#PCDATA)>
]>
<employees>Employee Data</employees>
```

Note: Use the root element name after `<!DOCTYPE`.

Element Declarations

- Element declaration syntax:

```
<!ELEMENT element-name content-model>
```

- Four kinds of content models:

```
<!ELEMENT job EMPTY><!-- Empty -->
```

①

```
<!-- Elements: single, ordered list, or choice -->
<!ELEMENT employees (employee)>
<!ELEMENT employee (employee_id,last_name,job_id)>
<!ELEMENT job_id (manager | worker)>
```

②

```
<!-- Mixed -->
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT hire_date (date| (day,month,year))>
```

③

```
<!ELEMENT employee_id ANY><!-- Any -->
```

④

Attribute Declarations

- The syntax for declaring an attribute is:

```
<!ATTLIST element-name attrib-name type default>
```

- Attribute declaration requires:
  - An element name
  - An attribute name
  - An attribute type, specified as:
    `CDATA`, enumerated, `ENTITY`, `ENTITIES`, `ID`, `IDREF`, `IDREFS`, `NMTOKEN`, `NMTOKENS`, and `NOTATION`
  - An attribute default, specified as:
    `#IMPLIED`, `#REQUIRED`, `#FIXED`, or a literal value
- Example:

```
<!ELEMENT employee (employee_id, last_name)>
<!ATTLIST employee manager_id CDATA #IMPLIED>
```

CDATA and Enumerated Attribute Types

- CDATA: For character data values

```
<!ELEMENT employee (employee_id, last_name)>
<!ATTLIST employee manager_id CDATA #IMPLIED>
```

```
<employee manager_id="102">          <!-- XML -->
  <employee_id>104</employee_id>
  <last_name>Ernst</last_name>
</employee>
```

- Enumerated: For a choice from a list of values

```
<!ELEMENT employee (employee_id, last_name)>
<!ATTLIST employee gender (male|female) #IMPLIED>
```

```
<employee gender="male">             <!-- XML -->
  <employee_id>104</employee_id>
  <last_name>Ernst</last_name>
</employee>
```


NOTATION Declaration and Attribute Type

- Declaring a NOTATION:

```
<!NOTATION notation_name SYSTEM "text">
```

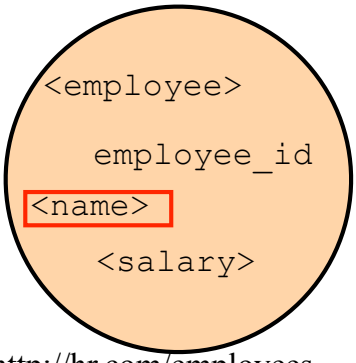
- The NOTATION attribute type represents a name of a NOTATION declared in the DTD:

```
<?xml version="1.0"?>
<!DOCTYPE photos [
<!ELEMENT photos (image+)>
<!ELEMENT image EMPTY>
<!NOTATION gif SYSTEM "image/gif">
<!NOTATION jpeg SYSTEM "image/jpeg">
<!ATTLIST image
   source CDATA #REQUIRED
   type NOTATION (gif | jpeg) #REQUIRED>
]>
<photos>
  <image source="myphoto.gif" type="gif"/>
  <image source="mypet.jpg" type="jpeg"/>
</photos>
```

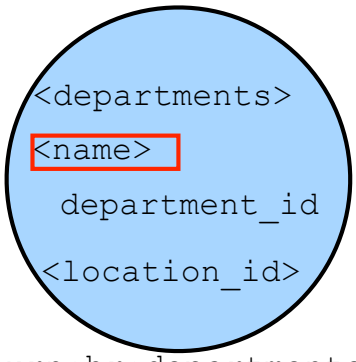
What Is an XML Namespace?

An XML namespace:

- **Is identified by a case-sensitive Internationalized Resource Identifier (IRI) reference (URL or URN)**
- **Provides universally unique names for a collection of names (elements and attributes)**



http://hr.com/employees



urn:hr:departments

Declaring XML Namespaces

Declare an XML namespace:

- **With the `xmlns` attribute in an element start tag:**
  - **Assigned an IRI (URL, URI, or URN) string value**
  - **Provided with an optional namespace prefix**
- **With a namespace prefix after `xmlns:` to form qualified element names:**

```
<dept:department
  xmlns:dept="urn:hr:department-ns">
...
</dept:department>
```

- **Without a prefix to form a "default namespace":**

```
<department xmlns="http://www.hr.com/departments">
...
</department>
```

XML Namespace Prefixes

A namespace prefix:

- May contain any XML character except a colon
- Can be declared multiple times as attributes of a single element, each with different names whose values can be the same or a different string
- Can be overridden in a child element by setting the value to a different string. For example:

```
<?xml version="1.0"?>
<emp:employee xmlns:emp="urn:hr:employee-ns">
  <emp:last_name>King</emp:last_name>
  <emp:address xmlns:emp="urn:hr:address-ns">
  500 Oracle Parkway
  </emp:address>
</emp:employee>
```


JUG Sardegna – http://www.jugsardegna.org

http://www.jugsardegna.org/vqwiki/jsp/Wiki?action=action_view_attachment&attachment=ArticoloAxis1PerJUG.pdf

http://www.jugsardegna.org/vqwiki/jsp/Wiki?action=action_view_attachment&attachment=ArticoloAxis2PerJUG.pdf

http://www.jugsardegna.org/vqwiki/jsp/Wiki?action=action_view_attachment&attachment=ArticoloAxis3PerJUG.pdf

Dispensa del corso di Architetture Software orientate ai Servizi

CONCRETE DEVELOPMENT OF A WEB SERVICE

AXIS Development

- Axis/Axis2 is an open-source Web Service development and runtime environment designed as a plug-in of common Web container
  (e.g., Apache Tomcat)
  - `http://ws.apache.org/axis/` and `http://ws.apache.org/axis2/`
  - `webapps/axis` to be installed in the servlet engine
- Basically is a servlet playing the role of SOAP Library and SOAP Listener

A Simple Service (1)

```
package miopackage;

public class SalutoWS {
   public String saluto(String name) {
   return "Ciao " + name + "!";
   }
}
```

- The class should be put in the right Axis directory, e.g., `<TOMCAT_HOME>\webapps\axis\WEB-INF\classes\miopackage\SalutoWS.class`
- Then the container should be made aware by deploying the class as a service
  - Deployment descriptor `<file>.wsdd`

A Simple Service (2)

```
<deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
   <service name="urn:SalutoWS" provider="java:RPC">
            <parameter name="className" value="miopackage.SalutoWS"/
   >
            <parameter name="allowedMethods" value="saluto"/>
            <parameter name="scope" value="Request"/>
   </service>
</deployment>
```

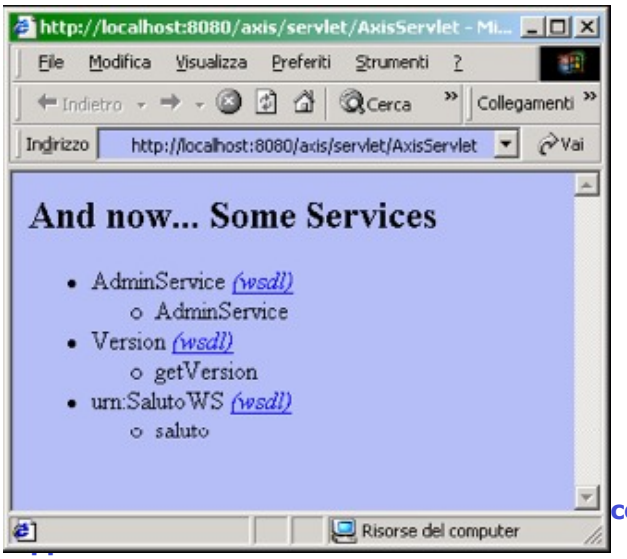
Life cycle of the service – transient (other values Application and Session)

A Simple Service (3)

Deployment through a specific tool

- `java org.apache.axis.client.AdminClient <file>.wsdd`

Development of a client

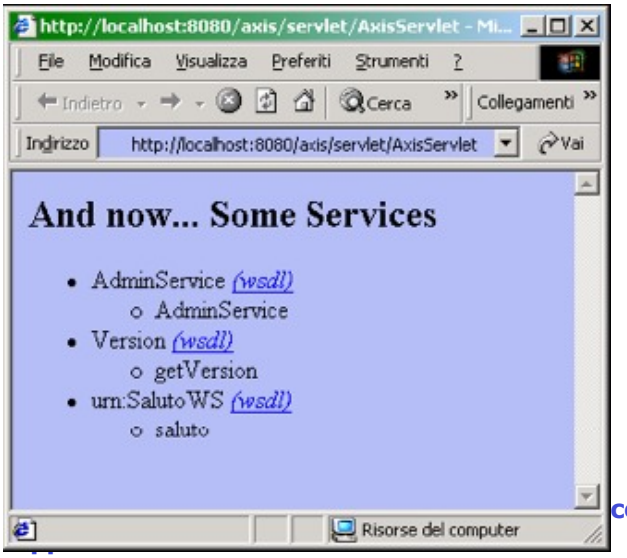


A Simple Service (3)

Deployment through a specific tool

- `java org.apache.axis.client.AdminClient <file>.wsdd`

Development of a client



A Simple Service (3)

Deploym

- java

Develop


```
import java.net.*;
import java.rmi.*;

import javax.xml.namespace.*;
import javax.xml.rpc.*;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

public class ClientSalutoWS {
  public static void main(String[] args) {
    String messaggio = "";
    try {
      Call call = (Call) new Service().createCall();
      call.setTargetEndpointAddress(new
           URL("http://localhost:8080/axis/services/"));
      call.setOperationName(new QName("urn:SalutoWS", "saluto"));
      Object rispostaWS = call.invoke(new Object[]{"Nicola"});
      messaggio = "il Web service ha risposto: "+(String) rispostaWS;
    }
    catch (MalformedURLException ex) {
      messaggio = "errore: l'url non è esatta";
    }catch (ServiceException ex) {
      messaggio = "errore: la creazione della chiamata è fallita";
    }catch (RemoteException ex) {
      messaggio = "errore: l'invocazione del WS è fallita";
    }finally{
      System.out.println(messaggio);
    }
  }
}
```

A Simple Service (4)

```
POST /axis/services/ HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 442

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>

    <ns1:saluto soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
        xmlns:ns1="urn:SalutoWS">
      <ns1:arg0 xsi:type="xsd:string">Nicola</ns1:arg0>
    </ns1:saluto>
  </soapenv:Body>
</soapenv:Envelope>
```


A Simple Service (4)

A Simple Service (4)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Connection: close
Date: Fri, 21 Nov 2003 16:10:50 GMT
Server: Apache Tomcat/4.0.4-b2 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd=http://www.w3.org/2001/XMLSchema
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:salutoResponse
        soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
        xmlns:ns1="urn:SalutoWS">
      <ns1:salutoReturn xsi:type="xsd:string">Ciao Nicola!</ns1:salutoReturn>
    </ns1:salutoResponse>
  </soapenv:Body>
</soapenv:Envelope>
```


Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

A More Complex Service (1) -- Serialization and (de)serialization

```
package miopackage.vo;

import java.io.*;

public class ProdottoVO implements Serializable {

  private int codice;
  private String nome;
  private boolean disponibile;
  private float prezzo;

  public ProdottoVO() {
  }
  public int getCodice() {
    return codice;
  }
  public void setCodice(int codice) {
    this.codice = codice;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public String getNome() {
    return nome;
  }
  public void setDisponibile(boolean disponibile) {
    this.disponibile = disponibile;
  }
  public boolean isDisponibile() {
    return disponibile;
  }
  public void setPrezzo(float prezzo) {
    this.prezzo = prezzo;
  }
  public float getPrezzo() {
    return prezzo;
  }
}
```

Application of the pattern "Data Transfer Object"

A More Complex Service (2) -- Serialization and (de)serialization

```
package miopackage;

import java.util.*;
import miopackage.vo.*;

public class CatalogoWS {

  public CatalogoWS() { }

  public Collection getListaProdotti() {
    Vector lista = new Vector();

    //prodotto1
    ProdottoVO prodotto1 = new ProdottoVO();
    prodotto1.setCodice(1);
    prodotto1.setNome("Lettore MP3");
    prodotto1.setDisponibile(true);
    prodotto1.setPrezzo(25.99f);
    lista.add(prodotto1);

    //prodotto2
    ProdottoVO prodotto2 = new ProdottoVO();
    prodotto2.setCodice(2);
    prodotto2.setNome("Display LCD");
    prodotto2.setDisponibile(true);
    prodotto2.setPrezzo(199.99f);
    lista.add(prodotto2);

    return lista;
  }

  public ProdottoVO getProdotto(int codice) {
    ProdottoVO prodotto = new ProdottoVO();
    prodotto.setCodice(codice);
    prodotto.setNome("Sub Woofer attivo");
    prodotto.setDisponibile(true);
    prodotto.setPrezzo(154.99f);
    return prodotto;
  }
}
```


A More Complex Service (3) -- Serialization and (de)serialization

```
<deployment
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="urn:catalogoWS" provider="java:RPC" style="rpc" use="encoded">
    <parameter name="className" value="miopackage.CatalogoWS"/>
    <parameter name="allowedMethods" value="getProdotto, getListaProdotti"/>
    <parameter name="scope" value="Session"/>
    <beanMapping qname="myNS:ProdottoVO" xmlns:myNS="urn:catalogoWS"
                 languageSpecificType="java:miopackage.vo.ProdottoVO"/>
  </service>
</deployment>
```

Compact form for.
Use the general form if you want to personally manage the serialization step

```
<typeMapping
  xmlns:ns=" urn:catalogoWS "
  qname=" myNS:ProdottoVO "
  type="java:miopackage.vo.ProdottoVO "
  serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
  deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
```

A More Complex Service (4) -- Serialization and (de)serialization

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions    targetNamespace="http://localhost:8080/axis/services/urn:catalogoWS"
xmlns="http://schemas.xmlsoap.org/wsdl/"     xmlns:apachesoap="http://xml.apache.org/xml-
soap"                          xmlns:impl="http://localhost:8080/axis/services/urn:catalogoWS"
xmlns:intf="http://localhost:8080/axis/services/urn:catalogoWS"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"     xmlns:tns1="urn:catalogoWS"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <schema targetNamespace="urn:catalogoWS" xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
        <complexType name="ProdottoVO">
          <sequence>
            <element name="codice" type="xsd:int"/>
            <element name="disponibile" type="xsd:boolean"/>
            <element name="nome" nillable="true" type="xsd:string"/>
            <element name="prezzo" type="xsd:float"/>
            </sequence>
          </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="getListaProdottiRequest">
  </wsdl:message>
  <wsdl:message name="getProdottoResponse">
    <wsdl:part name="getProdottoReturn" type="tns1:ProdottoVO"/>
  </wsdl:message>
  <wsdl:message name="getListaProdottiResponse">
    <wsdl:part name="getListaProdottiReturn" type="soapenc:Array"/>
  </wsdl:message>
  <wsdl:message name="getProdottoRequest">
    <wsdl:part name="codice" type="xsd:int"/>
  </wsdl:message>
```

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

A More Complex Service (5) -- Serialization and (de)serialization

```
<wsdl:portType name="CatalogoWS">
  <wsdl:operation name="getListaProdotti">
    <wsdl:input message="impl:getListaProdottiRequest" name="getListaProdottiRequest"/>
    <wsdl:output message="impl:getListaProdottiResponse"
                 name="getListaProdottiResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getProdotto" parameterOrder="codice">
    <wsdl:input message="impl:getProdottoRequest" name="getProdottoRequest"/>
    <wsdl:output message="impl:getProdottoResponse" name="getProdottoResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="urn:catalogoWSSoapBinding" type="impl:CatalogoWS">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getListaProdotti">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="getListaProdottiRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="http://miopackage" use="encoded"/>
    </wsdl:input>
    <wsdl:output name="getListaProdottiResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="http://localhost:8080/axis/services/urn:catalogoWS"
                 use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getProdotto">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="getProdottoRequest">
```

A More Complex Service (6) -- Serialization and (de)serialization

```
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://miopackage" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="getProdottoResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://localhost:8080/axis/services/urn:catalogoWS"
                       use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CatalogoWSService">
    <wsdl:port binding="impl:urn:catalogoWSSoapBinding" name="urn:catalogoWS">
      <wsdlsoap:address location="http://localhost:8080/axis/services/urn:catalogoWS"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```


A More Complex Service (7) -- Serialization and (de)serialization

```
package miopackage;

import java.net.*;
import java.rmi.*;
import java.util.*;

import javax.xml.namespace.*;
import javax.xml.rpc.*;

import miopackage.vo.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.ser.*;

public class ClientCatalogoWS {

  public static void main(String[] args) {
    try {
      Call call = (Call) new Service().createCall();
      call.setTargetEndpointAddress(new URL("http://localhost:8080/axis/services/"));

      QName qnameProd = new QName("urn:catalogoWS", "ProdottoVO");
      Class classeProd = ProdottoVO.class;
      call.registerTypeMapping(classeProd, qnameProd, BeanSerializerFactory.class,
                                                      BeanDeserializerFactory.class);

      //richiamo il metodo getProdotto
      call.setOperationName(new QName("urn:catalogoWS", "getProdotto"));
      Object rispostaWS = call.invoke(new Object[]{new Integer(1)});
      ProdottoVO prodotto = (ProdottoVO) rispostaWS;
      System.out.println("Il prodoto scelto è:");
      visualizza(prodotto);
```

Cfr. with the wsdd

A More Complex Service (8) -- Serialization and (de)serialization

```
  //richiamo il metodo getListaProdotti
  call.setOperationName(new QName("urn:catalogoWS", "getListaProdotti"));
  rispostaWS = call.invoke(new Object[]{});
  Collection lista = (Collection) rispostaWS;
  System.out.println("Il catalogo comprende:");
  Iterator iter = lista.iterator();
  while (iter.hasNext()) {
    ProdottoVO item = (ProdottoVO) iter.next();


        visualizza(item);
      }
    } catch (Exception ex) {
      System.out.println("Si è verificata l'eccezione: " + ex.toString());
    }
  }

  public static void visualizza(ProdottoVO prodotto) {
    if (prodotto == null) {
      return;
    }
    System.out.println("nome: " + prodotto.getNome());
    System.out.println("codice: " + prodotto.getCodice());
    String disponibile = (prodotto.isDisponibile()) ? "si" : "no";
    System.out.println("disponibile: " + disponibile);
    System.out.println("prezzo: " + prodotto.getPrezzo());
  }
}
```


A More Complex Service (9) -- Serialization and (de)serialization

Richiesta

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getProdotto
          soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:ns1="urn:catalogoWS">
      <ns1:arg0 xsi:type="xsd:int">1</ns1:arg0>
    </ns1:getProdotto>
  </soapenv:Body>
</soapenv:Envelope>
```

Risposta

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getProdottoResponse
          soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:ns1="urn:catalogoWS">
      <ns1:getProdottoReturn href="#id0"/>
    </ns1:getProdottoResponse>
    <multiRef id="id0" soapenc:root="0"
          soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          xsi:type="ns2:ProdottoVO"
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:ns2="urn:catalogoWS">
      <codice xsi:type="xsd:int">1</codice>
      <disponibile xsi:type="xsd:boolean">true</disponibile>
      <nome xsi:type="xsd:string">Sub Woofer attivo</nome>
      <prezzo xsi:type="xsd:float">154.99</prezzo>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

A More Complex Service (10) -- Serialization and (de)serialization

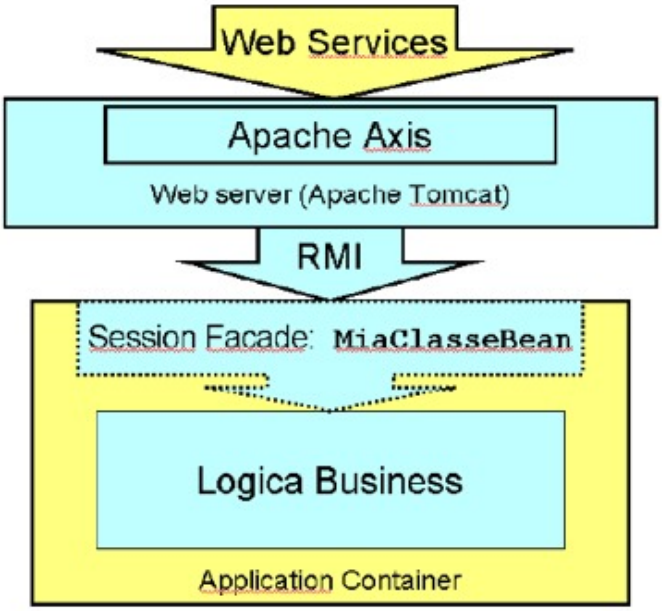
Richiesta

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getListaProdotti
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns1="urn:catalogoWS"/>
  </soapenv:Body>
</soapenv:Envelope>
```

Risposta

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getListaProdottiResponse
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns1="urn:catalogoWS">
      <ns1:getListaProdottiReturn href="#id0"/>
    </ns1:getListaProdottiResponse>
    <multiRef id="id0" soapenc:root="0"
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns2:Vector"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns2="http://xml.apache.org/xml-soap">
      <item href="#id1"/>
      <item href="#id2"/>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns3:ProdottoVO" xmlns:ns3="urn:catalogoWS"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
      <codice xsi:type="xsd:int">1</codice>
      <disponibile xsi:type="xsd:boolean">true</disponibile>
      <nome xsi:type="xsd:string">Lettore MP3</nome>
      <prezzo xsi:type="xsd:float">25.99</prezzo>
    </multiRef>
    <multiRef id="id2" soapenc:root="0"
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns4:ProdottoVO" xmlns:ns4="urn:catalogoWS"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
      <codice xsi:type="xsd:int">2</codice>
      <disponibile xsi:type="xsd:boolean">true</disponibile>
      <nome xsi:type="xsd:string">Display LCD</nome>
      <prezzo xsi:type="xsd:float">199.99</prezzo>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```


Services and Application Servers (1)



Services and Application Servers (2)


```
package miopackage;

import java.rmi.*;
import javax.ejb.*;
import miopackage.vo.* ;

public interface MiaClasse extends EJBObject{

  public RisultatoVO getRisultato(MioVO  vo);

}
```

```
package miopackage;

import java.rmi.*;
import javax.ejb.*;

public interface MiaClasseHome extends EJBHome {

  public MiaClasse create() throws RemoteException, CreateException;

}
```


Services and Application Servers (3)


```
<deployment
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="urn:nomeServizio" provider="java:EJB">
    <parameter name="jndiURL" value="jnp://nomehost:1099"/>
    <parameter name="jndiContextClass" value="org.jnp.interfaces.NamingContextFactory"/>
    <parameter name="beanJndiName" value="nomeJNDImiaEjb"/>
    <parameter name="homeInterfaceName" value="miopackage.MiaClasseHome"/>
    <parameter name="remoteInterfaceName" value="miopackage.MiaClasse"/>
    <parameter name="allowedMethods" value="getRisultato"/>
    <parameter name="scope" value="Session"/>
    <beanMapping qname="myNS1:MioVO" xmlns:myNS1=" urn:nomeServizio"
                 languageSpecificType="java:miopackage.vo.MioVO"/>
    <beanMapping qname="myNS2: RisultatoVO " xmlns:myNS2=" urn:nomeServizio"
                 languageSpecificType="java:miopackage.vo. RisultatoVO "/>

  </service>
</deployment>
```

To deploy
- `MiaClasseHome`, `MiaClasse` and `RisultatoVO` into `WEB-INF\classes`
- `Jboss-client.jar`, `jboss-j2ee.jar`, etc. into `WEB-INF\lib`