
CS 696

Introduction to Grid Computing:
Lecture #10

Mary Thomas

San Diego State

Tuesday, 20-Feb-07

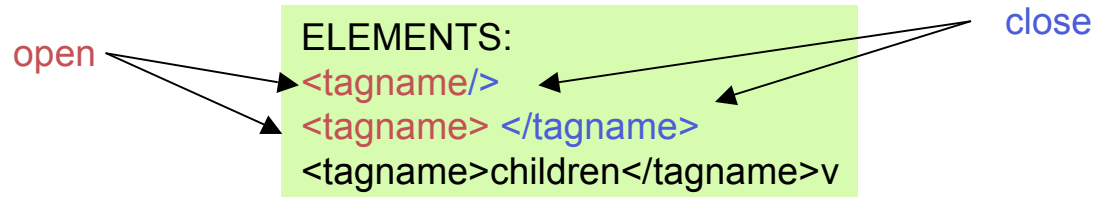
Announcements

- Today:
 - Namespaces
 - WSDL
 - <http://www.w3.org/TR/2006/CR-wsdl20-primer-20060327/>
 - WSDL2py: using python to access a WebService using the WSDL
 - <http://pywebsvcs.sourceforge.net/guide.html>

More on Namespaces: XML & SOAP

Examples adapted from: “Essential XML Quick Reference”, A. Skonnard & M. Gudgin (ISBN: 0-201-74095-8)

XML Namespaces: syntax



Namespaces appear in element start/open tag, associates elements

Have scope - set of elements {declared and children}; can override with new declaration in children

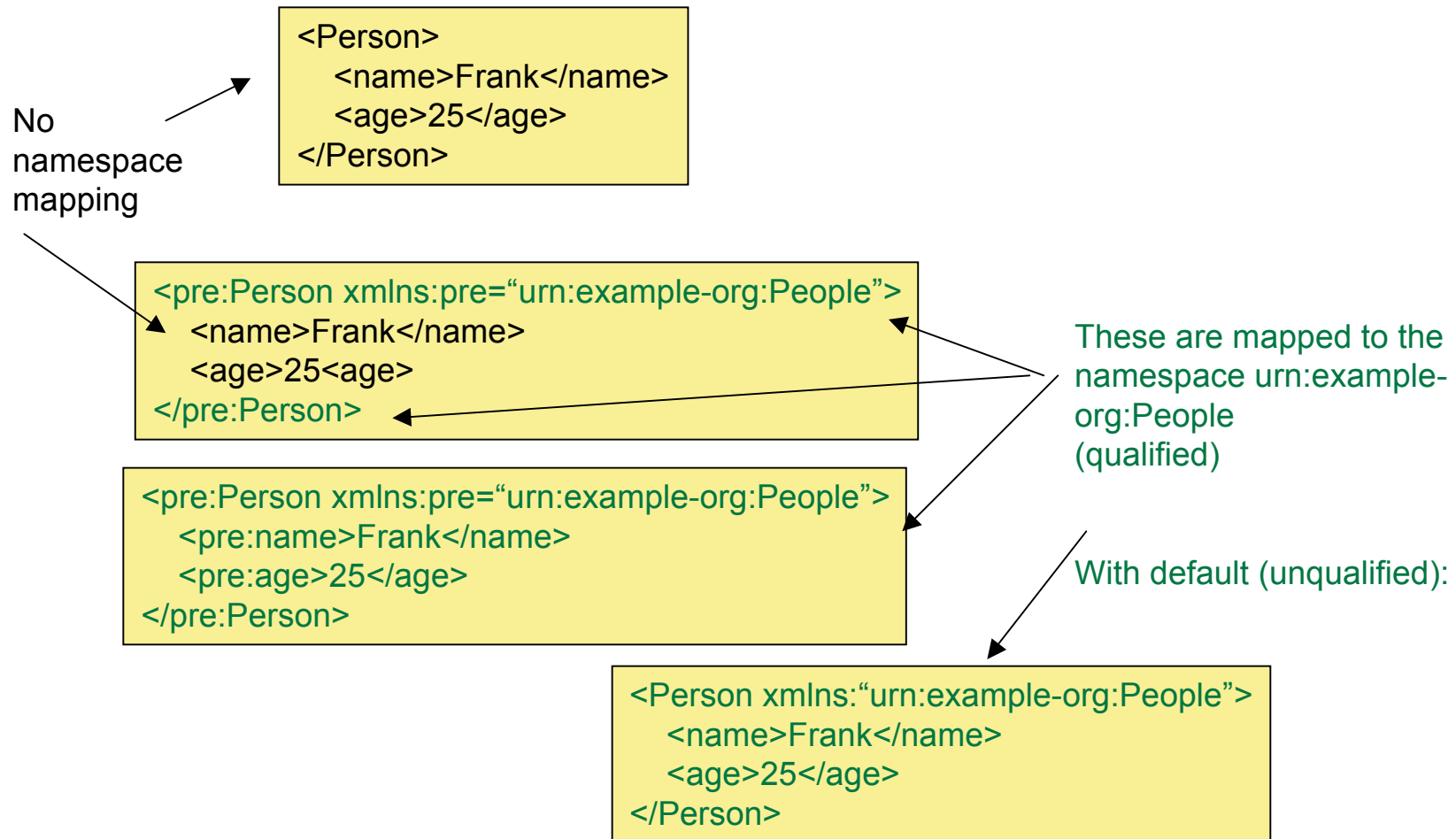
Can declare multiple namespaces

```
<prefix:localname xmlns:prefix='namespaceURI' />  
<prefix:localname xmlns:prefix='namespaceURI'></prefix:localname/>  
<prefix:localname xmlns:prefix='namespaceURI'>children</prefix:localname/>
```

Default (noprefix) namespace:

```
<prefix: localname xmlns='namespaceURI' />
```

Namespaces: qualified vs unqualified



Namespace: including attributes

```
<Person xmlns="urn:example-org:People"
  xmlns:b="urn:example-org:People:base"
  xmlns:u="urn:example-org:People:unit"
>
  <name>Frank</name>
  <age b:base='10' u:units='years'>25</age>
</Person>
```

XSLT: Extensible Stylesheet Language (XSL) Translations

- Used to translate one XML doc to another XML doc, or to HTML.
- Three programming models:
 - exemplar-based
 - procedural
 - declarative

```
<v1:emp xmlns="urn:employee:v1">  
  <fname>Frank</fname>  
  <lname>Zappa</lname>  
  <age b:base='10' u:units='years'>25</age>  
  <position>Guitarist</position>  
</v1:emp>
```



```
<v2:emp/oyee xmlns="urn:employee:v1">  
  <name>Frank Zappa</name>  
  <title>Guitarist</title>  
</v2:employee>
```

XSLT: Exemplar-based Model

```
<v2:employee
  xmlns:v1="urn:employee:v1"
  xmlns:v2="urn:employee:v2"
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
  xsl:version='1.0'
>
<name><xsl:value-of select="concat(/v1:emp/fname, ' ',
/v1:emp/lname)"/></name>
  <title><xsl:value-of select='/v1:emp/position'></title>
</v2:emp>
```

Exemplar-based is simplest model.

Allows you to take an XML doc and insert it with XSLT instructions

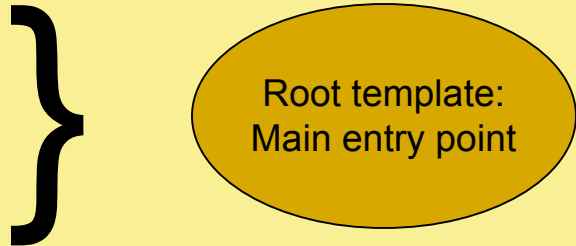
XSLT: Procedural Model

Procedural model allows you to define functions and call them from within the XML template document

```
<xsl:transform
  xmlns:v1="urn:employee:v1"
  xmlns:v2="urn:employee:v2"
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform' xsl:version='1.0'>

  <xsl:template name="outputName">
    <name><xsl:value-of select="concat(/v1:emp/fname, ' ',/v1:emp/lname)"/></name>
  </xsl:template>
  <xsl:template name="outputTitle">
    <title><xsl:value-of select='/v1:emp/position'/></title>
  </xsl:template>

  <xsl:template match="/">
    <v2:employee>
      <xsl:call-template name="outputName">
      <xsl:call-template name="outputTitle">
    </v2:employee>
  </xsl:template>
</xsl:transform>
```



Root template:
Main entry point

XSLT: Declarative Model

NEED TO MODIFY THIS. SEE PAGE 88

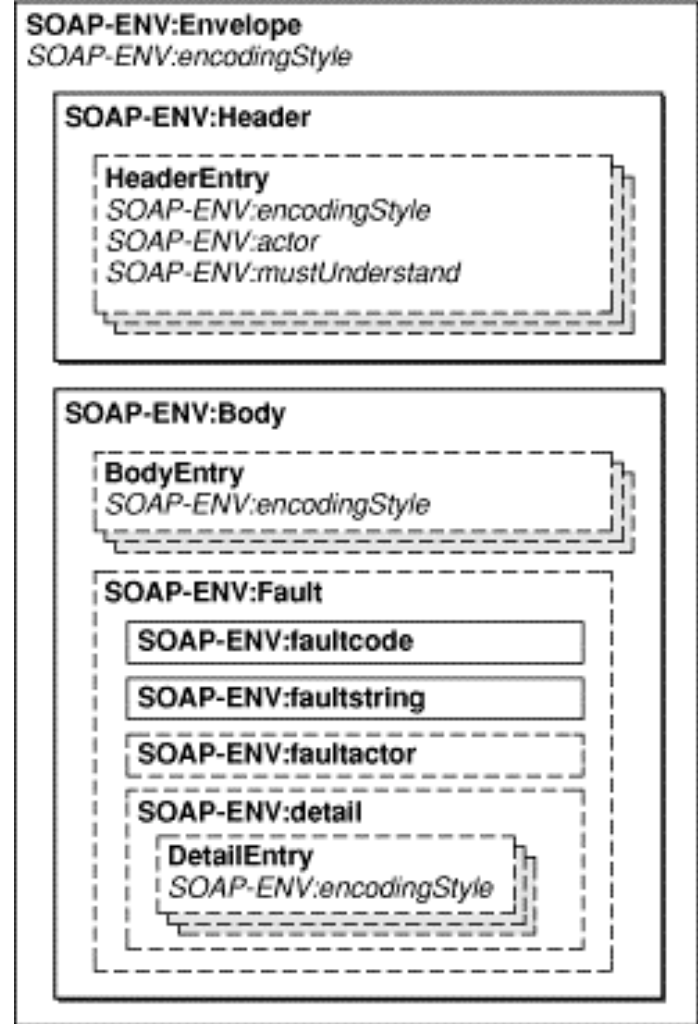
```
<xsl:transform
  xmlns:v1="urn:employee:v1"
  xmlns:v2="urn:employee:v2"
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform' xsl:version='1.0'>

  <xsl:template name="outputName">
    <name><xsl:value-of select="concat(/v1:emp/fname, ' ',/v1:emp/lname)"/></name>
  </xsl:template>
  <xsl:template name="outputTitle">
    <title><xsl:value-of select='/v1:emp/position' /></title>
  </xsl:template>

  <xsl:template match="/">
    <v2:employee>
      <xsl:call-template name="outputName">
      <xsl:call-template name="outputTitle">
    </v2:employee>
  </xsl:template>
</xsl:transform>
```

Skeleton SOAP Message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Header>
...
...
</soap:Header>
<soap:Body>
...
...
<soap:Fault>
...
...
</soap:Fault>
</soap:Body>
</soap:Envelope>
```



SOAP Envelope

- Root element of a SOAP message.
- Defines the XML document as a SOAP message
- The xmlns:soap Namespace:
 - message must have Envelope element associated with W3C namespace:
 - <http://www.w3.org/2001/12/soap-envelope>

```
<soap:Envelope  
  
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"  
  
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">  
<soap:Header>
```

SOAP Header

- Elements in SOAP Header define how a recipient should process the SOAP message
- Header elements must have attributes:
 - 3 attributes in namespace ("http://www.w3.org/2001/12/soap-envelope")
- *actor*: used to address Header element to endpoint
- *mustUnderstand true*: receiver processing Header must recognize element
- *encodingStyle*

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/appml/">
234
</m:Trans>
</soap:Header>
```

Client Request: without namespace

*** **Outgoing SOAP msg from CLIENT to SERVER*******

POST / HTTP/1.0

Host: localhost:9905

User-agent: SOAPpy 0.12.0 (http://pywebsvcs.sf.net)

Content-type: text/xml; charset="UTF-8"

Content-length: 540

SOAPAction: "addDbI"

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"

 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"

 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

 xmlns:xsd="http://www.w3.org/1999/XMLSchema"

>

<SOAP-ENV:Body>

<**addDbI SOAP-ENC:root="1"**>

<v1 xsi:type="xsd:double">125.123456789</v1>

<v2 xsi:type="xsd:double">427.98765432099998</v2>

</**addDbI**>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

Server Response: without namespace

```
*** Incoming SOAP MSG to CLIENT from SERVER *****
HTTP/1.? 200 OK
Server: <a href="http://pywebsvcs.sf.net">SOAPpy 0.12.0</a> (Python 2.3.5)
Date: Tue, 20 Feb 2007 23:23:17 GMT
Content-type: text/xml; charset="UTF-8"
Content-length: 519
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
  <SOAP-ENV:Body>
    <addDbIResponse SOAP-ENC:root="1">
      <Result xsi:type="xsd:double">553.1111111000002</Result>
    </addDbIResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

***** In CLIENT *****
Adding 125.123457 + 427.987654 = 553.111111
```

SOAPpy Server : Declaring namespaces

Namespace is bound to the function.

```
namespace = "http://acel.sdsu.edu/ws/math"  
# Start the server  
server = SOAPServer(("localhost", port))  
  
# register the methods  
#server.registerFunction(addDbl,namespace)
```

SOAPpy Client: Declaring namespaces

```
#force it at the server level
server = SOAPProxy("http://services.xmethods.com:9090/soap",
                  namespace = 'urn:xmethods-delayed-quotes',
                  http_proxy=proxy)
print "IBM>>", server.getQuote(symbol = 'IBM')

# Do it inline ala SOAP::LITE, also specify the actual ns
server = SOAPProxy("http://services.xmethods.com:9090/soap",
                  http_proxy=proxy)
print "IBM>>", server._ns('ns1',
                          'urn:xmethods-delayed-quotes').getQuote(symbol = 'IBM')

# Create a namespaced version of your server
dq = server._ns('urn:xmethods-delayed-quotes')
print "IBM>>", dq.getQuote(symbol='IBM')
print "ORCL>>", dq.getQuote(symbol='ORCL')
print "INTC>>", dq.getQuote(symbol='INTC')
```

```
namespace = "http://acel.sdsu.edu/ws/math"
# Default: use standard http
server = SOAPProxy("http://localhost:9905", namespace)
```

Client Request: with namespace

```
*** Outgoing HTTP headers *****
POST / HTTP/1.0
Host: localhost:9905
User-agent: SOAPpy 0.12.0 (http://pywebsvcs.sf.net)
Content-type: text/xml; charset="UTF-8"
Content-length: 589
SOAPAction: "addDbI"
*****

*** Outgoing SOAP *****
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<SOAP-ENV:Body>
<ns1:addDbI xmlns:ns1="http://acel.sdsu.edu/ws/math" SOAP-ENC:root="1">
<v1 xsi:type="xsd:double">125.123456789</v1>
<v2 xsi:type="xsd:double">427.987654320999998</v2>
</ns1:addDbI>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Server Response: with namespaces

```
*** Incoming HTTP headers from SERVER *****
HTTP/1.1 200 OK
Server: <a href="http://pywebsvcs.sf.net">SOAPpy 0.12.0</a> (Python 2.3.5)
Date: Tue, 20 Feb 2007 23:32:29 GMT
Content-type: text/xml; charset="UTF-8"
Content-length: 519
*****

*** Incoming SOAP *****
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<SOAP-ENV:Body>
<addDbIResponse SOAP-ENC:root="1">
<Result xsi:type="xsd:double">553.1111111000002</Result>
</addDbIResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
*****

Adding 125.123457 + 427.987654 = 553.111111
```

Server Response: bad namespace from Client

```
HTTP/1.1 500 Internal error
Server: <a href="http://pywebsvcs.sf.net">SOAPpy 0.12.0</a> (Python 2.3.5)
Date: Wed, 21 Feb 2007 00:12:07 GMT
Content-type: text/xml; charset="UTF-8"
Content-length: 709

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<SOAP-ENV:Body>
<SOAP-ENV:Fault SOAP-ENC:root="1">
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>Method Not Found</faultstring>
<detail xsi:type="xsd:string">http://junk.soapinterop.org/:quit : exceptions.KeyError u'http://junk.soapinterop.org/' &lt;traceback
object at 0x10272d8&gt;</detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
*****
<Fault SOAP-ENV:Client: Method Not Found: http://junk.soapinterop.org/:quit : exceptions.KeyError
u'http://junk.soapinterop.org/' <traceback object at 0x10272d8>>
Traceback (most recent call last):
  File "cardClient.py", line 41, in ?
    serv.quit()
  File "/sw/lib/python2.3/site-packages/SOAPpy/Client.py", line 470, in __call__
    return self.__r_call(*args, **kw)
  File "/sw/lib/python2.3/site-packages/SOAPpy/Client.py", line 492, in __r_call
    self.__hd, self.__ma)
  File "/sw/lib/python2.3/site-packages/SOAPpy/Client.py", line 406, in __call
    raise p
SOAPpy.Types.faultType: <Fault SOAP-ENV:Client: Method Not Found: http://junk.soapinterop.org/:quit : exceptions.KeyError
u'http://junk.soapinterop.org/' <traceback object at 0x10272d8>>
[gidget:cs696/soappy/card] mthomas%
```

SOAP Faults

- SOAP fault sent instead of normal response if something goes wrong

```
<Body xmlns=http://www.w3.org/2001/12/soap-envelope>
  <Fault>
    <faultcode>Client</faultcode>
    <faultstring>Something went wrong</faultstring>
    <detail>Application specific error information</detail>
  </Fault>
</Body>
```

- Fault Code

- Client – Message incorrectly formed by client
- Server – Problem on server so message could not proceed
- VersionMismatch – Invalid namespace for SOAP envelope
- MustUnderstand – Header element not understood

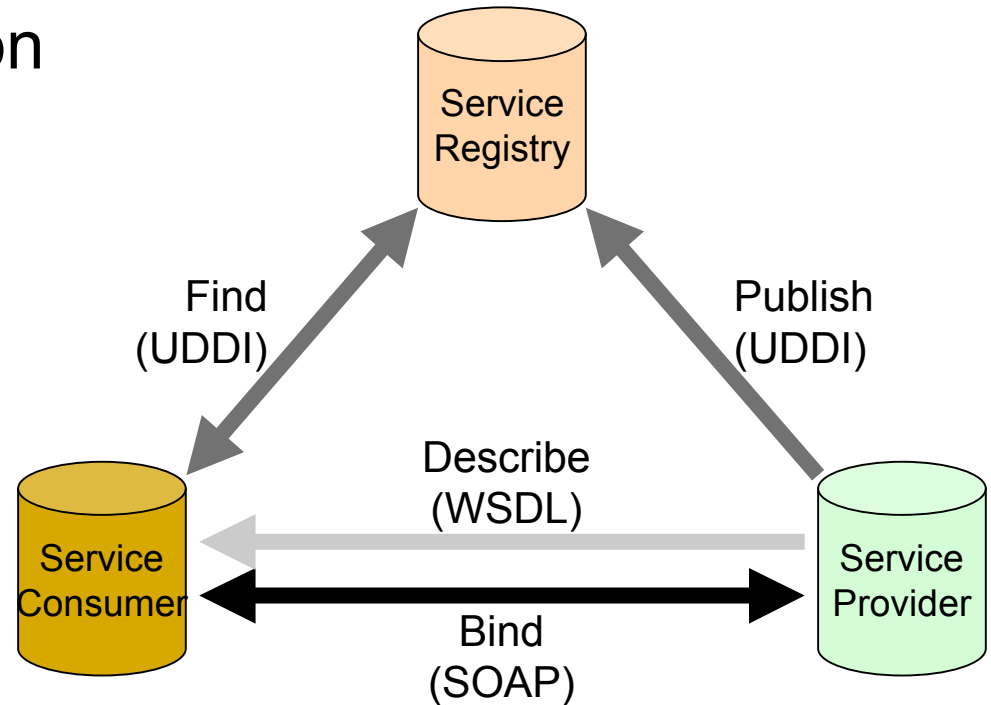
WSDL

Web Service Protocols

- Web Services based on XML Protocols

- Messaging
 - SOAP, XML-RPC
- Service Description
 - WSDL
- Service Discovery
 - UDDI
- Many Others
 - BPEL, WSRF, WS-Addressing

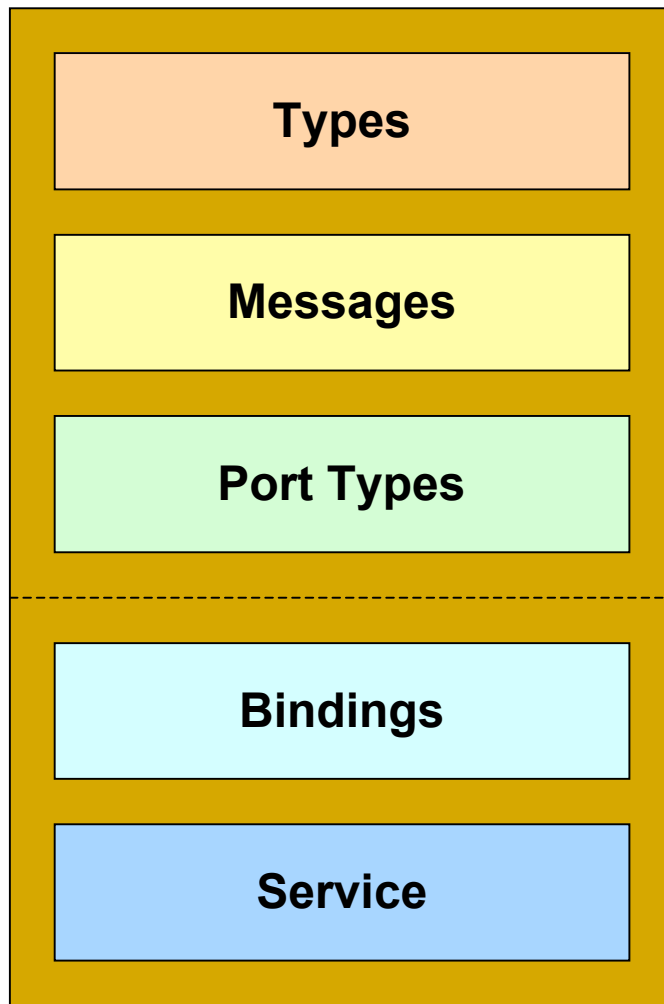
- SOAP, WSDL and UDDI are de-facto standards



WSDL

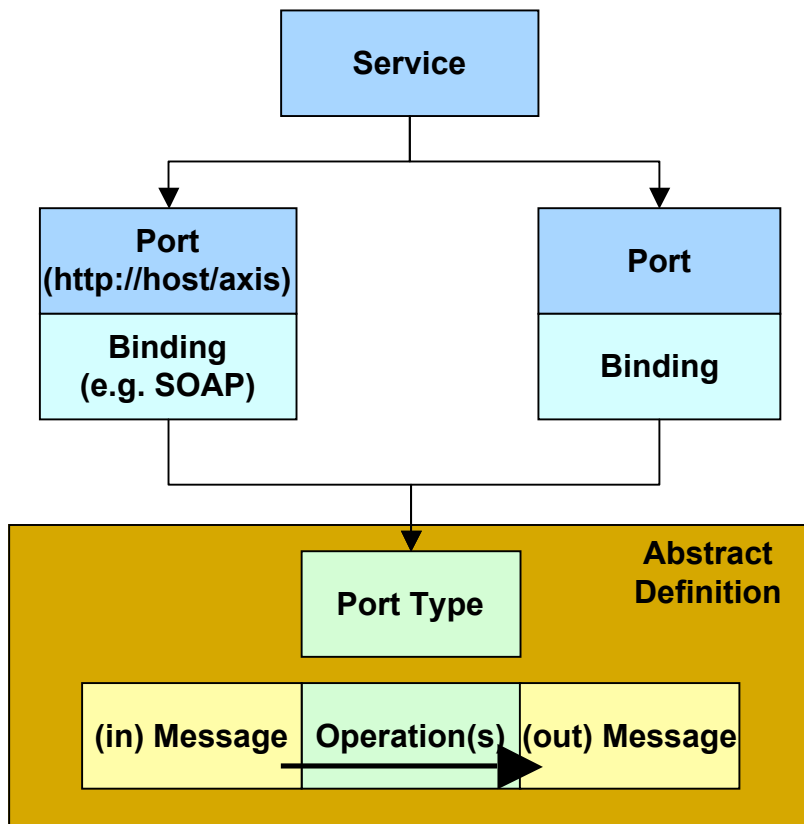
- Web Service Definition Language (WSDL)
- WSDL documents describe:
 - What the service can do,
 - Where the service resides,
and
 - How to invoke the service
- Generally available as web pages
 - e.g. <http://bouscat.cs.cf.ac.uk/someService?WSDL>
 - Location of WSDL relative to Web Service not specified
- Like CORBA Interface Definition Language (IDL) but more flexible

WSDL Documents (1)



- **Types**
 - What data types will be transmitted
- **Messages**
 - What messages will be transmitted
- **Port Types**
 - What operations (functions) will be supported
- **Bindings**
 - How will the messages be transmitted on the wire?
 - What message protocol (e.g. SOAP) specific details are there?
- **Service**
 - Where is the service located?

WSDL Documents (2)



- Service has multiple ports
 - A port is an endpoint to which messages are sent
 - e.g. `http://cs.cf.ac.uk/axis/service`
- Each port is bound to:
 - A message protocol
 - e.g. SOAP
 - A port type
- Port types specify:
 - Operation name
 - Input message type
 - Output message type
- Types defined using XML Schemas

Web Services Description Language (WSDL)

- You now know how to make an XML doc
- You now know how messages are exchanged with XML between a client and a server.
 - Client **Request**: select a **method**, submit **data**
 - Server **Response**: return XML **data**
- WSDL is an XML document that describes a Web service.
 - It specifies the location of the **service** and the **operations** (or methods) the service exposes.

WSDL Structure

ELEMENT DEFINITION

<portType>

The operations performed by the web service

<message>

The messages used by the web service

<types>

The data types used by the web service

<binding>

The communication protocols used by the web service

```
<definitions>
  <types>
    definition of types.....
  </types>
  <message>
    definition of a message....
  </message>
  <portType>
    definition of a port.....
  </portType>
  <binding>
    definition of a binding....
  </binding>
  <other elements>
</definitions>
```

WSDL: <portType>

- Most important WSDL element
 - Most used
- Defines:
 - the web service
 - the operations that can be performed
 - the messages (methods) used
- Similar to a function library (or a module, or a class) in a traditional programming language

WSDL: <message>

- The <message> element defines the data elements of an operation
- Each messages can consist of one or more parts.
 - Like <param> in XML-RPC
 - Similar to the arguments or parameters of a function call in a traditional programming language.

WSDL: <types> and <binding>

■ WSDL Types

- ❑ <types> element defines the data type used by the web service.
- ❑ WSDL uses XML Schema syntax to define data types.
 - More portable

■ WSDL Bindings

- ❑ <binding> element defines the message format and protocol details for each port.

WSDL <portType> operation types

- The request-response type is the most common operation type, but WSDL defines four types:

TYPE	DEFINTION
One-Way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

WSDL: one-way operation

```
<message name="newTermValues">  
  <part name="term" type="xs:string"/>  
  <part name="value" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="setTerm">  
    <input name="newTerm" message="newTermValues"/>  
  </operation>  
</portType >
```

WSDL Binding: SOAP

- The **binding** element has two attributes - **name** and **type**
 - Name: (you can use any name you want) defines name of binding
 - Type: attribute points to port for the binding
- The **soap:binding** element has two attributes - **style** and **transport**.
 - Style: can be "rpc" or "document"
 - Transport: defines the SOAP protocol to use (eg HTTP)
- The **operation** element defines each operation that the port exposes:
 - For each op the corresponding SOAP action has to be defined.
 - Must specify how the input and output are encoded

SOAPpy: WSDL Proxy

- SOAPProxy wrapper
 - parses method names, namespaces, soap actions from the web service description language (WSDL) file passed into the constructor.
- The WSDL reference can be passed in as
 - a stream, a url, a file name, or a string.
- Loads info into self.methods
 - a dictionary with methodname keys and values of WSDLTools.SOAPCallinfo.

WSDL Client: Babelfish WSDL

```
# Check for a web proxy definition in environment
try:
    proxy_url=os.environ['http_proxy']
    phost, pport = re.search('http://(?:[^\:]+):([0-9]+)', proxy_url).group(1,2)
    proxy = "%s:%s" % (phost, pport)
except:
    proxy = None

server = WSDL.Proxy('http://www.xmethods.net/sd/2001/BabelFishService.wsdl',
                   http_proxy=proxy)
print len(server.methods)
print server.methods.keys()

english = "Hi Friend!"

print "Babelfish Translations"
print "-----"
print "English: '%s'" % english
print "French: '%s'" % server.BabelFish('en_fr',english)
print "Spanish: '%s'" % server.BabelFish('en_es',english)
print "Italian: '%s'" % server.BabelFish('en_it',english)
print "German: '%s'" % server.BabelFish('en_de',english)
```

Babelfish WSDL:

<http://www.xmethods.net/sd/2001/BabelFishService.wsdl>

```
<?xml version="1.0"?>
<definitions name="BabelFishService" xmlns:tns="http://www.xmethods.net/sd/BabelFishService.wsdl"
targetNamespace="http://www.xmethods.net/sd/BabelFishService.wsdl" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="BabelFishRequest">
    <part name="translationmode" type="xsd:string"/>
    <part name="sourcedata" type="xsd:string"/>
  </message>
  <message name="BabelFishResponse">
    <part name="return" type="xsd:string"/>
  </message>
  <portType name="BabelFishPortType">
    <operation name="BabelFish">
      <input message="tns:BabelFishRequest" />
      <output message="tns:BabelFishResponse" />
    </operation>
  </portType>
  <binding name="BabelFishBinding" type="tns:BabelFishPortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="BabelFish">
      <soap:operation soapAction="urn:xmethodsBabelFish#BabelFish"/>
      <input>
        <soap:body use="encoded" namespace="urn:xmethodsBabelFish" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </input>
      <output>
        <soap:body use="encoded" namespace="urn:xmethodsBabelFish" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </output>
    </operation>
  </binding>
  <service name="BabelFishService">
    <documentation>Translates text of up to 5k in length, between a variety of languages.</documentation>
    <port name="BabelFishPort" binding="tns:BabelFishBinding">
      <soap:address location="http://services.xmethods.net:80/perl/soaplite.cgi"/>
    </port>
  </service>
</definitions>
```

Babelfish Client Output

```
HTTP/1.? 200 OK
Date: Wed, 21 Feb 2007 02:15:19 GMT
Server: Apache/1.3.26 (Unix) Enhydra-Director/3 PHP/4.0.6 DAV/1.0.3 AuthNuSphere/1.0.0
SOAPServer: SOAP::Lite/Perl/0.52
Content-Length: 546
Connection: close
Content-Type: text/xml; charset=utf-8
X-Pad: avoid browser bug
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
<SOAP-ENV:Body>
<namespace1:BabelFishResponse xmlns:namespace1="urn:xmethodsBabelFish">
<return xsi:type="xsd:string">Salutations, Classe ! </return>
</namespace1:BabelFishResponse>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
*****
French: 'Salutations, Classe !'
```

Babelfish Translations

```
-----
English: 'Greetings, Class!'
French: 'Salutations, Classe !'
Spanish: 'Saludos, Clase!'
Italian: 'Saluti, Codice categoria!'
```