Notes

• Case study group setup at catme.org
  ▪ Please respond before Tuesday next week to have better group setup

• Discussion
  ▪ To boost discussion, one write-up for the whole group is fine
  ▪ Write down the names you talked with (at least one)
# Discussion #3

## Web page VS Web service

<table>
<thead>
<tr>
<th></th>
<th>Web page</th>
<th>Web service</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>HTML, css, script language</td>
<td>Most in XML</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Data process and presentation</td>
<td>Data process</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>for human to view business to consumer (B2C) e-commerce</td>
<td>for program to parse and understand results business to business (B2B) interactions automating business processes</td>
</tr>
<tr>
<td><strong>Integration</strong></td>
<td>Not easy to integrate multiple pages</td>
<td>We can compose multiple web services into one</td>
</tr>
<tr>
<td><strong>Registration</strong></td>
<td>No standard for web page</td>
<td>Standard and tools for registration</td>
</tr>
<tr>
<td><strong>Commonalities</strong></td>
<td>Both are based on HTTP protocol</td>
<td>Both are based on HTTP protocol</td>
</tr>
<tr>
<td></td>
<td>Both are hosted by a web server</td>
<td>Both are hosted by a web server</td>
</tr>
<tr>
<td></td>
<td>Both are web-based applications</td>
<td>Both are web-based applications</td>
</tr>
<tr>
<td></td>
<td>Web technologies in this chapter are useful to both</td>
<td>Web technologies in this chapter are useful to both</td>
</tr>
</tbody>
</table>
Top Mistakes for Homework 2

• Results should be valid

• Wrong qualification element declared as being either 0 or 1 occurrence. It should be
  ▪ <xs:element name="qualifications" type="qualificationtype" minOccurs="0" maxOccurs="1"/>

• No actual request value and response content
Learning Outcomes

• After learning chapter 5, you should be able to
  ▪ Understand the structure of WSDL documents
  ▪ Write XSLT to transform XML documents
Steps for Service Invocation

1. Create the service
2. Generate the web service description for the service
3. Register the web service
4. Publish the web service
5. Discover the web service
6. Understand the web service semantics
7. Invocate the web service
WSDL

- WSDL (pronounced “Wisdel”) is an XML vocabulary that describes all aspects of web services. It stands for web services description language.
- A WSDL document consists of seven basic XML elements and we will examine each one.
- WSDL schema: http://schemas.xmlsoap.org/wsdl
Like program interface, the same abstract definition could have multiple concrete binding

We mainly study WSDL 1.1
WSDL Examples

• Temperature Convert

• Weather Forecast

• Airport Checking
  ▪ [http://www.webservicex.net/airport.asmx?WSDL](http://www.webservicex.net/airport.asmx?WSDL)
WSDL Types

- The types tag contains one or more schema tags as children for all the types used by the service and defined with XMLSchema.
WSDL Message

• There are two message elements per web service operation
  ▪ Incoming message
  ▪ Outgoing message

• A message is an abstract, typed definition of the data being exchanged (for request and response)
WSDL portType

• The portType element describes one or more abstract operations
• Each operation is supported by one or more portTypes

```xml
<wsdl:portType name="ConvertTemperatureSoap">
  <wsdl:operation name="ConvertTemp">
    <wsdl:input message="tns:ConvertTempSoapIn" />
    <wsdl:output message="tns:ConvertTempSoapOut" />
  </wsdl:operation>
</wsdl:portType>
```
WSDL Binding

• The binding element specifies a **concrete** protocol for **transport** binding and data format specification for portType operations

• The style element determines the RPC or Document styles

```
<wSDL:binding name="ConvertTemperatureSoap" type="tns:ConvertTemperatureSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wSDL:operation name="ConvertTemp">
    <wSDL:input>
      <soap:body use="literal" />
    </wSDL:input>
    <wSDL:output>
      <soap:body use="literal" />
    </wSDL:output>
  </wSDL:operation>
</wSDL:binding>
```
The use="encoded" attribute refers to a SOAP encoding in WSDL message element

- It specifies how objects, structures, arrays, and object graphs should be serialized
- It is only used with RPC
- An example: http://www.herongyang.com/WSDL/WSDL-11-SOAP-12-Example-rpc-encoded.html

The use="literal" attribute refers to data that is serialized according to an XML schema

- It can be used with either RPC or document styles
WSDL Binding Style

- There are three allowed of the four logical combinations of the binding style and the soap:body use attribute:
  - RPC/literal
  - Document/literal
  - RPC/encoded
- Literal means the data is serialized according to the XMLSchema
- Encoded means the depreciated SOAP encoding is used
- We will always use the literal serialization format
WSDL HTTP GET Binding

```xml
<wsdl:binding name="ConvertTemperatureHttpGet" type="tns:ConvertTemperatureHttpGet">
  <http:binding verb="GET"/>
  <wsdl:operation name="ConvertTemp">
    <http:operation location="/ConvertTemp"/>
    <wsdl:input>
      <http:urlEncoded/>
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

- **http method**: GET
- **Location to be appended after service location**: `/ConvertTemp`
- **Input data is sent through url**: `http?urlEncoded` is used for sending data through the URL.
- **Result is sent in xml, http body**: `mime:mimeXml` is used for sending the response in XML format in the HTTP body.
WSDL HTTP POST Binding

```xml
<wsdl:binding name="ConvertTemperatureHttpPost" type="tns:ConvertTemperatureHttpPost">
  <http:binding verb="POST"/>
  <wsdl:operation name="ConvertTemp">
    <http:operation location="/ConvertTemp"/>
    <wsdl:input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

Input data is sent via www form
```
Curl Commands for Different Bindings

• curl -v -X POST -d @soapConvertTemp.txt
  http://www.webservicex.net/ConvertTemperature.asmx --header
  "SOAPAction:http://www.webserviceX.NET/ConvertTemp" --header
  "Content-Type:text/xml"

• curl -v -X GET
  'http://www.webservicex.net/ConvertTemperature.asmx/ConvertTemp?Temperature=100&FromUnit=degreeFahrenheit&ToUnit=degreeCelsius'

• curl -v -X POST -d
  'Temperature=100&FromUnit=degreeFahrenheit&ToUnit=degreeCelsius'
  'http://www.webservicex.net/ConvertTemperature.asmx/ConvertTemp'
WSDL Binding/Operation

• 4 Message Exchange Patterns (MEPs)
  ▪ request-response
  ▪ solicit-response
  ▪ one-way
  ▪ notification

▪ Pay attention to the order of input and output tags
WSDL Service

• The service element identifies the web service, as indicated by the value of the attribute name

• It contains ports. Each port links a binding and an address.

```xml
<wSDL:service name="ConvertTemperature">
  <wSDL:port name="ConvertTemperatureSoap" binding="tns:ConvertTemperatureSoap">
    <soap:address location="http://www.webservicex.net/ConvertTemperature.asmx" />
  </wSDL:port>
  <wSDL:port name="ConvertTemperatureSoap12" binding="tns:ConvertTemperatureSoap12">
    <soap12:address location="http://www.webservicex.net/ConvertTemperature.asmx" />
  </wSDL:port>
  <wSDL:port name="ConvertTemperatureHttpGet" binding="tns:ConvertTemperatureHttpGet">
    <http:address location="http://www.webservicex.net/ConvertTemperature.asmx" />
  </wSDL:port>
  ...
</wSDL:service>
```

Additional ports
WSDL Versions

• WSDL 1.1 - our examples use this
• WSDL 1.2 – newest
  ▪ also called WSDL 2.0
UDDI (Universal Description, Discovery, and Integration)

- Mainly used for service registration and discovery
- We will not cover this in detail
- It is not widely used
- Other options are commonly used for registries - databases, LDAP, ebXML, etc.
Web Services Interoperability (WS-I) basic profile

- XML 1.0
- XMLSchema 1.0
- SOAP 1.1
- WSDL 1.1
- UDDI 2.0
XSLT (Extensible Stylesheet Language Transformations)

• An XML-based functional language that can transform any XML document into any other XML document
• Web browser can be used to transform an XML via specified XSL file
• Tutorial link at W3Schools: https://www.w3schools.com/xml/xsl_intro.asp

• Examples
  ▪ For-each: http://userpages.umbc.edu/~jianwu/is651/programs/ch5/cd.xml
  ▪ Apply-template: http://userpages.umbc.edu/~jianwu/is651/programs/ch5/cd2.xml
XSLT Example

XML document with XSL attribute

```xml
<?xml version="1.0" encoding="utf-8"?>
<catalog>
    <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Hide your heart</title>
        <artist>Bonnie Tyler</artist>
        <company>CBS Records</company>
        <price>9.90</price>
        <year>1988</year>
    </cd>
</catalog>
```

XSL document

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml" >
    <xsl:template match="cd">
        <tr bgcolor="#9acd32">
            <td>Title</td>
            <td>Artist</td>
            <td>Company</td>
            <td>Price</td>
            <td>Year</td>
        </tr>
        <xsl:for-each select="catalog/cd">
            <tr>
                <td><xsl:value-of select="title" /></td>
                <td><xsl:value-of select="artist" /></td>
                <td><xsl:value-of select="company" /></td>
                <td><xsl:value-of select="price" /></td>
                <td><xsl:value-of select="year" /></td>
            </tr>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```
XSLT Apply-Templates

• A better and more manageable way to do iteration is to use the recursive apply-templates tag instead of for-each.
XSLT Apply-Templates Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <ul>
          <li>Empire Burlesque for the artist Bob Dylan</li>
          <li>Hide your heart for the artist Bonnie Tyler</li>
        </ul>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```