Notes

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

• Homework 3
  ▪ It has tasks for both chapter 5 and 6. Make sure you work on all of them
  • New deadline because of spring break:
    • Monday, 03/18 for early submission and feedback
    • Saturday, 03/23 for final submission
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>Language</td>
<td>HTML, css, script language</td>
<td>Most in XML</td>
</tr>
<tr>
<td>Data</td>
<td>Data process and presentation</td>
<td>Data process</td>
</tr>
<tr>
<td>Usage</td>
<td>for human to view business to consumer (B2C)</td>
<td>for program to parse and understand results</td>
</tr>
<tr>
<td></td>
<td>e-commerce</td>
<td>business to business (B2B) interactions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>automating business processes</td>
</tr>
<tr>
<td>Integration</td>
<td>Not easy to integrate multiple pages</td>
<td>We can compose multiple web services into one</td>
</tr>
<tr>
<td>Registration</td>
<td>No standard for web page</td>
<td>Standard and tools for registration</td>
</tr>
<tr>
<td>Commonalities</td>
<td>Both are based on HTTP protocol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Both are hosted by a web server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Both are web-based applications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web technologies in this chapter are useful</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to both</td>
<td></td>
</tr>
</tbody>
</table>
Top Mistakes for Homework 2

• Results should be valid

• Wrong qualification element declared as being at least 2 occurrences. It should be
  ▪ `<xs:element name="qualifications" type="qualificationtype" minOccurs="2" maxOccurs="unbounded"/>

• 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 five 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
  ▪ https://www.w3schools.com/xml/tempconvert.asmx?WSDL

• CD store
  ▪ https://userpages.umbc.edu/~jianwu/is651/programs/ch10/cd.php?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

```xml
<wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="https://www.w3schools.com/xml/">
    <s:element name="FahrenheitToCelsius">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Fahrenheit" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="FahrenheitToCelsiusResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="FahrenheitToCelsiusResult" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="CelsiusToFahrenheit">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Celsius" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="CelsiusToFahrenheitResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="CelsiusToFahrenheitResult" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
```
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="TempConvertSoap">
  <wsdl:operation name="FahrenheitToCelsius">
    <wsdl:input message="tns:FahrenheitToCelsiusSoapIn" />
    <wsdl:output message="tns:FahrenheitToCelsiusSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="CelsiusToFahrenheit">
    <wsdl:input message="tns:CelsiusToFahrenheitSoapIn" />
    <wsdl:output message="tns:CelsiusToFahrenheitSoapOut" />
  </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

```xml
<wSDL:binding name="TempConvertSoap" type="tns:TempConvertSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="FahrenheitToCelsius">
    <soap:operation soapAction="https://www.w3schools.com/xml/FahrenheitToCelsius" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wSDL:binding>
```

Request-response
WSDL Binding (2)

• 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

• 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 deprecated SOAP encoding is used
• We will always use the literal serialization format
WSDL HTTP POST Binding

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

- **http method**: POST
- **Input data is sent via www form**
- **Result is sent in xml, http body**
- **Location to be appended after service location**
Curl Commands for Different Bindings

• curl -v -X POST -d @soapConvertTemp.txt
  https://www.w3schools.com/xml/tempconvert.asmx --header
  "soapAction:https://www.w3schools.com/xml/FahrenheitToCelsius"
  -header "Content-Type:text/xml"

• curl -v -X POST -d 'Fahrenheit=100'
  'https://www.w3schools.com/xml/tempconvert.asmx/FahrenheitToCelsius'
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.

```
<wSDL:service name="TempConvert">
  <wSDL:port name="TempConvertSoap" binding="tns:TempConvertSoap">
    <soap:address location="http://www.w3schools.com/xml/tempconvert.asmx" />
  </wSDL:port>
  <wSDL:port name="TempConvertSoap12" binding="tns:TempConvertSoap12">
    <soap12:address location="http://www.w3schools.com/xml/tempconvert.asmx" />
  </wSDL:port>
  <wSDL:port name="TempConvertHttpPost" binding="tns:TempConvertHttpPost">
    <http:address location="http://www.w3schools.com/xml/tempconvert.asmx" />
  </wSDL:port>
</wSDL:service>
```
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 version="1.0" encoding="utf-8"?>
<xml-stylesheet type="text/xsl" href="stylesheet.xsl "?>
<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>
```

XML document with XSL attribute

```
<xml version="1.0" encoding="utf-8"?>
<xml-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>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Title</th>
          <th>Artist</th>
          <th>Year</th>
        </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="year"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xml-stylesheet>
```

XSL document
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>
                    Title: <span style="color:#ff0000">Empire Burlesque</span> for the Artist: <span style="color:#00ff00">Bob Dylan</span>
                </li>
                <li>
                    Title: <span style="color:#ff0000">Hide your heart</span> for the Artist: <span style="color:#00ff00">Bonnie Tyler</span>
                </li>
                ...
            </ul>
        </body>
    </html>
</xsl:template>

<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<h2>My CD Collection</h2>
<ul>
    <li>
        Title: <span style="color:#ff0000">Empire Burlesque</span> for the Artist: <span style="color:#00ff00">Bob Dylan</span>
    </li>
    <li>
        Title: <span style="color:#ff0000">Hide your heart</span> for the Artist: <span style="color:#00ff00">Bonnie Tyler</span>
    </li>
    ...
</ul>
</body>
</html>
```