Quick Notes

• Make sure you try the dummy exam *SystemTest1* before our next class
  ▪ If you try it too late, you might not have enough time to fix the problems on time
  ▪ If you do not try it, you might spend a lot of exam time learning how to use it, or you might not be able to take the exam at all
# 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>1. for human to view</td>
<td>1. for program to parse and understand results</td>
</tr>
<tr>
<td></td>
<td>2. business to consumer (B2C)</td>
<td>2. business to business (B2B) interactions</td>
</tr>
<tr>
<td></td>
<td>3. e-commerce</td>
<td>3. 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>1. Both are based on HTTP protocol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Both are hosted by a web server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Both are web-based applications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Web technologies in this course are useful to both</td>
<td></td>
</tr>
</tbody>
</table>
IS 651: Distributed Systems
Chapter 5: WSDL

Jianwu Wang
Fall 2020
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
Relationship between WSDL Elements

- Like program interface, the same abstract definition could have multiple concrete binding
- We mainly study WSDL 1.1

Figure by Cristcost - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=7642526
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:element name="string" nillable="true" type="s:string"/>
  </s:schema>
</wsdl:types>
```

This element can be null
WSDL Message

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

```xml
<wsdl:message name="FahrenheitToCelsiusSoapIn">
  <wsdl:part name="parameters" element="tns:FahrenheitToCelsius" />
</wsdl:message>

<wsdl:message name="FahrenheitToCelsiusSoapOut">
  <wsdl:part name="parameters" element="tns:FahrenheitToCelsiusResponse" />
</wsdl: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">
      <soap:input use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```
Attribute *use* in 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

- The use="literal" attribute refers to data that is serialized according to an XML schema
  - It can be used with either RPC or document style
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 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
- **Location to be appended after service location**: `/FahrenheitToCelsius`
- **Input data is sent via www form**: application/x-www-form-urlencoded
- **Result is sent in xml, http body**: mimeXml part="Body"
Curl Commands for Different Bindings

- curl -v -X POST -d @soapConvertTemp.xml  
  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.

```xml
<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>
```
More on WSDL

• WSDL Version
  ▪ WSDL 1.1 - our examples use this
  ▪ WSDL 1.2 – newest, also called WSDL 2.0

▪ You will know how to call a web service by just checking its WSDL
▪ Very often, WSDL files are generated automatically based on your service implementation
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
- XML Schema 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"?>
<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>
```

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="/">
  <html>
    <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr bgcolor="#f0f0f0">
          <th>Title</th>
          <th>Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
```
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="/">
        <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>
    </xsl:template>
</xsl:stylesheet>
```