Notes

• IS 651 changes from last semester
  • Bigger class size, lab classroom to lecture classroom
  • Grader helping students’ questions and separate explanation on assignments
  • More lecture time
  • Longer deadline for exercise and homework
  • More ways to get help: forum, grader, (in-class, office hour, instructor)

• More specific on questions
  • Which parts you don’t understand? What’s your understanding?

• This week’s class includes chapter 5 and 6
  ▪ No separate exercise for Chp 6. You only need to work on exercise for Chp 5
  ▪ Homework 4 includes both Ch5 Homework and Ch6 Homework
  ▪ Deadline: 3/18 for ex4 and 3/21 for hw4
Learning Outcomes

• After learning chapter 5, you should be able to
  ▪ Understand the structure of WSDL documents
  ▪ Write XSLT to transform XML documents
Critical Thinking Questions

• What is the default port for SOAP and why is it?
  ▪ 80: It is over HTTP. An advantage is that server provider side doesn’t need to open additional ports to pass firewall
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
Relationship between WSDL Elements

- Like program interface, the same abstract definition could have multiple concrete binding

Figure by Cristcost - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=7642526
WSDL Examples

• Shakespeare WSDL
  ▪ Service description link at [http://www.xmlme.com/WSShakespeare.asmx](http://www.xmlme.com/WSShakespeare.asmx)

• Weather 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:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="Request" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="GetSpeechResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetSpeechResult" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
</s:schema>
</wSDL:types>
```
WSDL Message

- There are two message elements per web service operation, which is an abstract, typed definition of the data being exchanged (for request and response)
WSDL portType

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

```xml
,portType name="ShakespeareSoap">
  <operation name="GetSpeech">
    <input message="tns:GetSpeechSoapIn"/>
    <output message="tns:GetSpeechSoapOut"/>
  </operation>
</portType>
```
WSDL Binding

• The binding element specifies a concrete protocol for transport binding and data format specification for a portType operation.
• The style element determines the RPC or Document styles.

Example WSDL binding for ShakespeareSoap service:

```xml
<binding name="ShakespeareSoap" type="tns:ShakespeareSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetSpeech">
    <soap:operation
      soapAction="http://xmlme.com/WebServices/GetSpeech"
      style="document" />
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```
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
  ▪ 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/Operation

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

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

• The service element identifies the web service, as indicated by the value of the attribute name
• It links port, binding and address
WSDL Versions

- WSDL 1.1 - our examples use this
- WSDL 1.2 - newest
UDDI

• We will not cover this in detail
• 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 (XSLT) is 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: http://www.w3schools.com/xsl/default.asp

• Examples
  ▪ For-each: http://userpages.umbc.edu/~jianwu/is651/programs/chp5/cd.xml
  ▪ Apply-template: http://userpages.umbc.edu/~jianwu/is651/programs/chp5/cd2.xml
XSLT Example

```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>
```

XML document with XSL attribute

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0">
  <xsl:import href="http://www.w3.org/1999/XSL/Transform"/>
  <xsl:import href="http://www.w3.org/1999/xhtml"/>
</xsl:stylesheet>
```

XSL document
XSLT Apply-Templates

• A better and more manageable way to do this 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>
</xsl:stylesheet>
```
XSLT in Web Services

• XSLT is also used to transform XML documents in web services from one XML format to another when the participants need it.

• In the PO example we used earlier, one participant might use a different XMLSchema for the PO natively on their system than the one that is used by the web service that processes the POs even though all the actual content is the same.