

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

	Web page	Web service
Language	HTML, css, script language	Most in XML
Data	Data process and presentation	Data process
Usage	1. for human to view 2. business to consumer (B2C) 3. e-commerce	1. for program to parse and understand results 2. business to business (B2B) interactions 3. automating business processes
Integration	Not easy to integrate multiple pages	We can compose multiple web services into one
Registration	No standard for web page	Standard and tools for registration
Commonalities	1. Both are based on HTTP protocol 2. Both are hosted by a web server 3. Both are web-based applications 4. Web technologies in this course are useful to both	

IS 651: Distributed Systems

Chapter 5: WSDL

Jianwu Wang

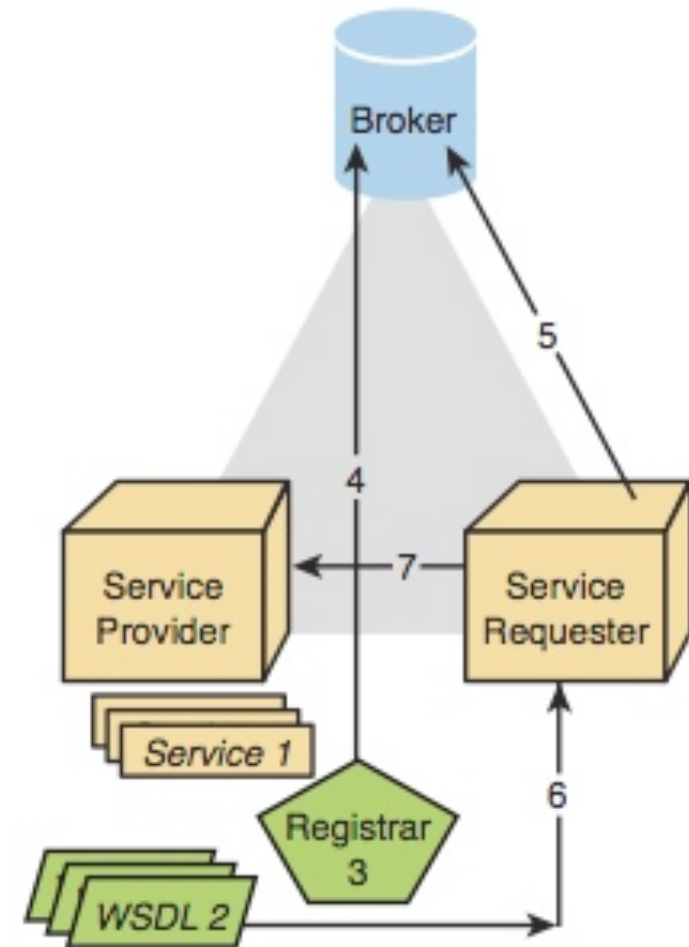
Spring 2021

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. Invoke the web service

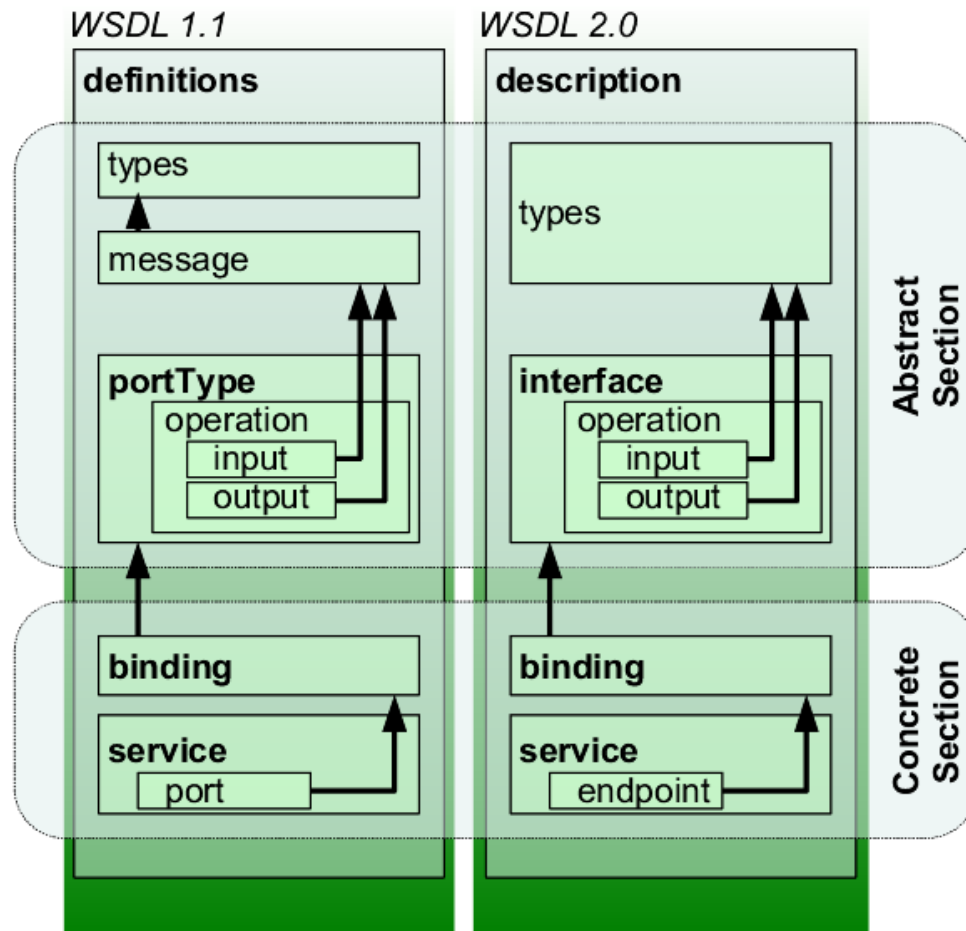


WSDL

- 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>

```
<?xml version="1.0" encoding="utf-8"?>
<definitions
  xmlns:s="http://www.w3.org/2000/10/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://xmlme.com/WebServices"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <types>...</types>
  <message>...</message>
  <message >...</message>
  <portType>...</portType>
  <binding>...</binding>
  <service>...</service>
</definitions>
```

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
- The example contains 5 elements

```
<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

```
<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

```
<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>
```

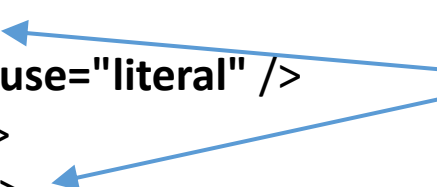
WSDL Binding

- On top of `wsdl:portType` and its `wsdl:operation` elements, WSDL Binding defines the transport protocol and data format for each port type
- Different transport protocols could be used for WSDL Binding
 - SOAP
 - HTTP POST
 - SMTP (Simple Mail Transfer Protocol)
 - ...
- Type attribute of `wsdl:binding` element uses portType defined earlier for data format
 - Example: `<wsdl:binding name="TempConvertSoap" type="tns:TempConvertSoap">`
 - With type value, we know data types of each operation's input/output

WSDL Binding to SOAP

- soap:binding element specifies a **concrete** protocol for **transport** binding
- soap:operation element specifies how to generate soap request/response message for each wsdl:operation and additional info including *soapAction*, *style* and *use*
- The *style* attribute of soap:operation element determines the RPC or Document styles

```
<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>
```



Correspond to SOAP request & response. Detailed info is at tns:TempConvertSoap

Attribute *use* in WSDL SOAP 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 style

WSDL SOAP 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

```
<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>
```

Annotations:

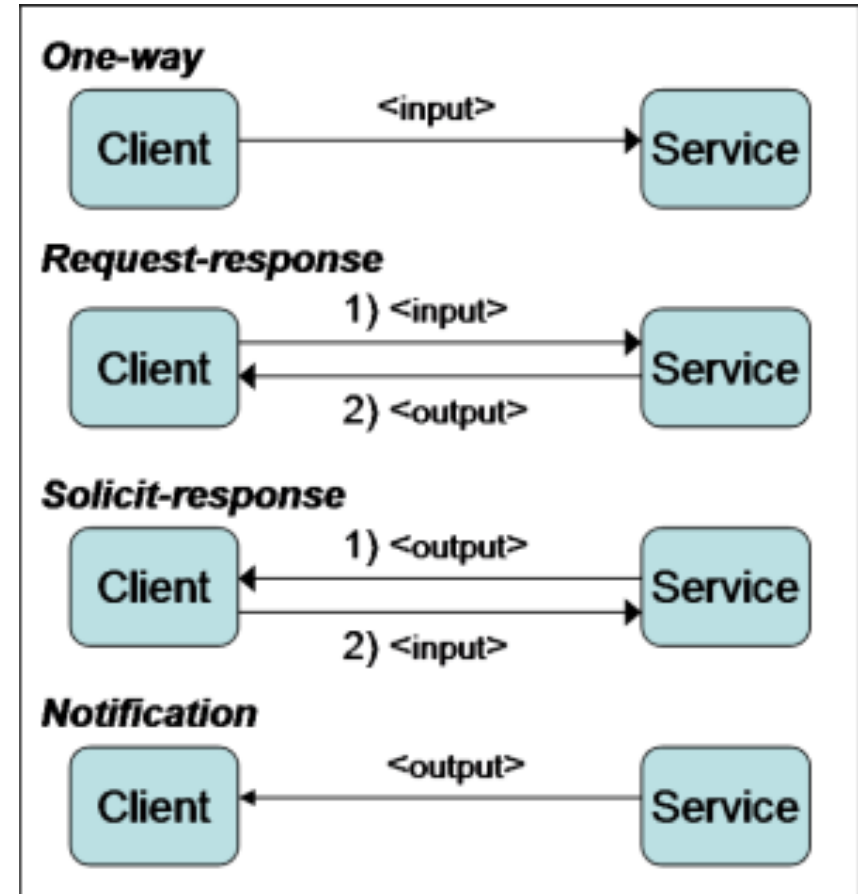
- http method
- Location to be appended after service location
- Input data is sent via www form
- Result is sent in xml, http body

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.

```
<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
- 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: <https://userpages.umbc.edu/~jianwu/is651/programs/ch5/cd.xml>
 - Apply-template:
<https://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> ...
```

XML document with XSL attribute

```
<?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="#9acd32"><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>
</xsl:stylesheet>
```

for-each

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 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>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <ul>
    <li><xsl:apply-templates select="title"/> <xsl:apply-templates select="artist"/>
    </li>
  </ul>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span> for the
</xsl:template>
```



XSLT Apply-Templates

```
<?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>
    </ul>
    <ul>
      <li>
        Title: <span style="color:#ff0000">Hide your heart</span> for the
        Artist: <span style="color:#00ff00">Bonnie Tyler</span>
      </li>
    </ul>
    ...
  </body>
</html>
```

Html results