Notes on Homework and Exercises

• Each homework/exercise assignment grading is final one week after its notification date
  ▪ Contact me within the week if you have questions with my grading

• No error message from your program does not mean your homework/exercise is perfect
  ▪ The program needs also to meet all requirements of the assignment
  ▪ I highly recommend you check the requirements again after you finish programming to make sure every requirement is met
Exam Info

• Midterm: October 8th. 75 minutes. Multi-answers questions and short-answer questions. In-class, closed-book exam.

• Not every question is covered in class. Read the book and know how to do exercise/homework.

• Grading rule: (summation of partial point for each correct choice) - (summation of partial point for each incorrect choice). Minimal point is 0
  ▪ E.g.: A question has 5 choices (a, b and d are correct). If your answer is a, d and e, you get (2/3 - 1/2) of the total point.
  ▪ In this way, you will get partial credits for the parts you did right.

• Send me up to 5 good questions in your opinion, I’ll use top ones in the exam.
  ▪ Via private posts to All Instructors. Can be a group effort. You can add some explanation.

• During exam, it is difficult to discuss questions with me. Just answer the question based on your understanding. If you have questions, you can contact me after exam.

• Plagiarism will not be tolerated. Multiple submissions with the same answers will be investigated and reported.
  ▪ You have much higher chance to fail because of plagiarism than not learning well in class.
Lockdown Browser for Exam

• The exams require a **web camera** to record your activity during the exam. Make sure the computer you will use for the exam has a web camera.
  - If you really cannot find a computer with web camera, contact me ASAP (at least one week before). I may have to prepare special exams for you, such as phone camera based proctoring.

• Install the [Respondus Lockdown Browser (RLDB) software](https://wiki.umbc.edu/pages/viewpage.action?pageId=87884496) on the computer you will use for exams, open it and connect it to UMBC blackboard.
  - The software only works for Windows and Mac machines
  - You need to close all communication software like WebEx, Skype and Chrome, in order to start the lockdown browser.

• You will also need to show your campus card before taking the exam.

• I have prepared a dummy Exam, called **SystemTest1**, for you to try things out. You need to take the exam **at least one week before the actual exam date (10/08)** to make sure you will not face technical problems during the actual exam. I will monitor the exam taking records and remind the students who have not done it.

• If you have problems installing the software, connecting it to UMBC blackboard, or taking the dummy exam, report the issues you have on Piazza

• If you cannot attend the exam on 10/08 4:30-5:45 PM ET for some reason, **contact me ASAP** (at least one week before the exam date) and explain the reason why you cannot attend the exam

• More about Respondus Lockdown Browser (RLDB) software: [https://wiki.umbc.edu/pages/viewpage.action?pageId=87884496](https://wiki.umbc.edu/pages/viewpage.action?pageId=87884496)
IS 651: Distributed Systems  
Chapter 4: SOAP

Jianwu Wang  
Fall 2020
Learning Outcomes

• After learning this chapter, you should be able to
  ▪ Understand the differences between DTD and XML schema
  ▪ Write XML schema and XML documents based on XML schema
  ▪ Understand SOAP messages and how SOAP works
XMLSchema

- XMLSchema is the alternative and more modern method of validating XML documents
- It has XML syntax
- It has datatypes
  - Built-in
  - User-defined
    - simple
    - complex
- It uses namespaces
  - Namespaces are a general concept from programming to avoid name collision
  - URL for XML, package name for Java
XMLSchema Syntax

• Example
  - Note.xml
  - XML DTD for note.xml
  - XML Schema for note.xml

• Namespace
  - Defined in xmlns attribute
  - xmlns:prefix="URL"
  - xmlns="URL" for default prefix
  - Prefix used throughout schema

xs is the prefix defined here and used throughout schema
XML Example

- **Demo link**
- The xmlns attribute value does not need to be a real url
- The xsd file is in the same folder of the xml file
**XML Schema Example**

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://userpages.umbc.edu/~jianwu/po"
    xmlns="http://userpages.umbc.edu/~jianwu/po"
    elementFormDefault="qualified">
    <xs:simpleType name="inttype">
        <xs:restriction base="xs:positiveInteger"/>
    </xs:simpleType>
    <xs:complexType name="shiptotype">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="itemtype">
        <!--User-defined type -->
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="inttype"/>
            <xs:element name="price" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="shipordertype">
        <!--Built-in type -->
        <xs:sequence>
            <xs:element name="orderperson" type="xs:string"/>
            <xs:element name="shipto" type="shiptotype"/>
            <xs:element name="item" maxOccurs="unbounded" type="itemtype"/>
        </xs:sequence>
        <xs:attribute name="orderid" type="inttype" use="required"/>
    </xs:complexType>
    <xs:element name="shiporder" type="shipordertype"/>
</xs:schema>
```

- Demo link, Same schema with different prefix
- two xmlns
- targetNamespace
- xs:element for root element datatype
XML Validation Against Schema

- Command line:
  - `xmllint --noout --schema XSD_FILE XML_FILE`

- [XML Validator - XSD (XML Schema)](xml_validator_url)
SOAP

• SOAP was originally defined as the 'simple object access protocol’, but has nothing to do with objects
  ▪ It is now just a name and not an acronym
• It is the messaging protocol for XML web services
• SOAP is described in XML following SOAP schema
• SOAP is usually used with HTTP
• SOAP structure
  ▪ Envelop {(optional) header, (required) body}
• RPC-style SOAP VS. Document-style SOAP
RPC-style SOAP

- The message corresponds to a procedure call
- This xml contains three prefixes: soap, tx, m

```xml
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Header>
    <tx:Trans
        xmlns:tx="http://www.example.org/transaction/
        soap:mustUnderstand="1">
      234
    </tx:Trans>
  </soap:Header>
  <soap:Body
      xmlns:m="http://www.example.org/product-prices">
    <m:GetProductPrice>
      <m:productId>450R</m:productId>
    </m:GetProductPrice>
  </soap:Body>
</soap:Envelope>
```

An example for RPC-style SOAP request message.
RPC-style SOAP (2)

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
 <soap:Body xmlns:m="http://www.example.org/product-prices">
   <m:ProductPrice>
     <m:price>20</m:price>
     <m:unit>dollar</m:unit>
   </m:ProductPrice>
 </soap:Body>
</soap:Envelope>

An example for RPC-style SOAP response message.
Document-style SOAP

• Client just sends XML documents
• Service knows what to do

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Header>
    <tx:Trans xmlns:tx="http://www.example.org/transaction/" soap:mustUnderstand="1">234</tx:Trans>
  </soap:Header>
  <soap:Body>
    <po:shiporder po:orderid="889923" xmlns:po="http://userpages.umbc.edu/~jianwu/">
      <po:orderperson>John Smith</po:orderperson>
      <po:shipTo>
        <po:name>Ola Nordmann</po:name>
        <po:address>Langgt 23</po:address>
        <po:city>4000 Stavanger</po:city>
        <po:country>Norway</po:country>
      </po:shipTo>
      <po:item>
        <po:title>Empire Burlesque</po:title>
        <po:note>&lt; Special Edition &gt;</po:note>
        <po:quantity>1</po:quantity>
        <po:price>10.90</po:price>
      </po:item>
    </po:shiporder>
  </soap:Body>
</soap:Envelope>
SOAP Fault

- Return message for error
- Corresponds to throwing exception in normal programming

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <soap:Body>
        <soap:Fault>
            <faultcode xsi:type="xsd:string">soap:Client</faultcode>
            <faultstring xsi:type="xsd:string">Failed to locate method.</faultstring>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

An example for SOAP fault message.
Intermediaries

- Initial Senders
- Intermediaries
- Ultimate Receivers

Unlike the Initial Sender and Ultimate Receiver applications, intermediaries do not act on the Body content of the SOAP message.

Intermediaries essentially act upon the information encoded in the appropriate Header element.
SOAP Versions

• SOAP 1.1
• SOAP 1.2 – newest
  ▪ Also known as 2.0
### SOAP Operation

<table>
<thead>
<tr>
<th>Requestor</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requestor creates SOAP</td>
<td>Provider parses SOAP for J2EE programs in a servlet container processes SOAP</td>
</tr>
<tr>
<td>SOAP Client sends SOAP</td>
<td>SOAP Listener receives SOAP</td>
</tr>
<tr>
<td>HTTP</td>
<td>HTTP</td>
</tr>
</tbody>
</table>

Implementation architecture based on J2EE framework
On-line SOAP Example: ConvertTemp Demo

POST /xml/tempconvert.asmx HTTP/2
Host: www.w3schools.com
User-Agent: curl/7.59.0
SOAPAction: https://www.w3schools.com/xml/FahrenheitToCelsius
Content-Type: text/xml
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:tns="https://www.w3schools.com/xml/">
 <SOAP-ENV:Body>
  <FahrenheitToCelsius xmlns="https://www.w3schools.com/xml/">
   <Fahrenheit>double</Fahrenheit>
  </FahrenheitToCelsius>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

HTTP/2 200
Content-Type: text/xml; charset=utf-8
Date: Mon, 04 Mar 2019 14:54:17 GMT

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <FahrenheitToCelsiusResponse xmlns="https://www.w3schools.com/xml/">
   <FahrenheitToCelsiusResult>
    <double></double>
   </FahrenheitToCelsiusResult>
  </FahrenheitToCelsiusResponse>
 </soap:Body>
</soap:Envelope>

request message

response message

IS 651: Distributed Systems
Software Architecture for SOAP

• Architecture of the on-line SOAP example

• SOAP demo using curl command
  ```
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"
  ```

• NOTE: The Shakespeare Service example in the book chapter does not work any more.