Notes

• Grading statistics
  ▪ Midterm1: average 11.57 out of 15 with stdev 1.81
  ▪ Total: average 16.92 out of 21 with stdev 1.91
    ▪ A range: [18.83, 21]
    ▪ B range: [15.01, 18.83)
    ▪ C or worse range: [0, 15.01)
  ▪ The curve never exceeds a standard grading scale
  ▪ Know your position and think what you need to know to improve it
  ▪ We will not have additional assignments
  ▪ Contact me if you need help with the course

• Many thanks to students who sent me exam question suggestions. At least one question is used from each one’s suggestions.

• Any feedback/suggestions?
Notes 2

• Suggestions on study well
  ▪ Try to get full points in discussion, homework and case-study, 50 points total.
  ▪ Attend the class
  ▪ Read book chapters before class and ask questions in class
  ▪ Start homework early
  ▪ Check the requirements one more time after it is done.
  ▪ If you are experiencing any problems that affect your performance in this class, please contact me immediately
  ▪ Ask help from me and grader: exercises, homework instructions, and homework (after it is graded)
Notes 2

• Homework grading policy on late submission and resubmission
  ▪ 1 point deduction for no submission before deadline
  ▪ 2 point deduction for no submission before next week’s class
  ▪ 1.5 point deduction for resubmission within one week after next week’s class

• Reasons
  ▪ Encouraging submitting before deadline
  ▪ Encouraging learning from and correcting your mistakes
What are the main reasons for the evolution of distributed systems?

1. Hardware evolution
2. Finer work division (identify general functionalities)
3. Flexibility (hide heterogeneity)
4. Standardization

- Some old techniques/hardware die out: Mainframe, CORBA, etc.
- Some still in use (internally): Client/Server architecture, Synchronous, J2EE
IS 651: Distributed Systems
Chapter 4: SOAP

Jianwu Wang
Spring 2018
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
XML Example

- **Demo link**
- **Same XML with different schema location**
XML Schema Example

```xml
<?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`
  • DEMO

• [XML Validator - XSD (XML Schema)](XML-Validator-XSD)
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 }
- Protocol VS. API
- RPC-style SOAP VS. Document-style SOAP
RPC-style SOAP

• The message corresponds to a procedure call

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

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

```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>
    <po:shiporder po:orderid="889923" xmlns:po="http://userpages.umbc.edu/~jianwu/po">
      <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 - our examples use that
• SOAP 1.2 - newest
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 that processes SOAP</td>
</tr>
<tr>
<td>SOAP Client sends SOAP</td>
<td>SOAP Listener receives SOAP</td>
</tr>
</tbody>
</table>

HTTP ←→ HTTP

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

**request message**

```
POST /ConvertTemperature.asmx HTTP/1.1
Host: www.webservicex.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.webserviceX.NET/ConvertTemp"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ConvertTemp xmlns="http://www.webserviceX.NET/">
<Temperature>double</Temperature>
<FromUnit>degreeCelsius or degreeFahrenheit or degreeRankine or degreeReaumur or kelvin</FromUnit>
<ToUnit>degreeCelsius or degreeFahrenheit or degreeRankine or degreeReaumur or kelvin</ToUnit>
</ConvertTemp>
</soap:Body>
</soap:Envelope>
```

**response message**

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ConvertTempResponse xmlns="http://www.webserviceX.NET/">
<ConvertTempResult>double</ConvertTempResult>
</ConvertTempResponse>
</soap:Body>
</soap:Envelope>
```
Software Architecture for SOAP

• Architecture of the on-line SOAP example

• SOAP demo using curl command
  
curl -v -X POST -d @soapConvertTemp.txt
  http://www.webservicex.net/ConvertTemperature.asmx --header
  "SOAPAction:http://www.webserviceX.NET/ConvertTemp" --header "Content-Type:text/xml"

• SOAP demo using postman
Exercise #3
## Discussion #3

**• Web Pages VS Web Services**

<table>
<thead>
<tr>
<th>Difference aspect 1</th>
<th>Web page</th>
<th>Web service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference aspect 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commonalities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>