Final

• Dec 20\textsuperscript{th}, 7PM – 9PM EST
• In this lecture hall
• No books, no notes, …
• \textbf{Everything} we covered in the class
  – Complete lectures
  – All assigned readings
  – All homework assignments
  – All topics discussed in class that aren’t on the slides
• 70-80 questions
  – Most multiple choice
  – Some short answer
  – Some text questions, some UML, some code samples, some diagrams
Final Exam

• It will start EXACTLY at 7:10 and run until 9:10 pm. Be here and in your seats, bright smiling faces, ready to go by 7pm! I will start handing out the exams at that time.

• It is 40% of your course grade.
Final Lecture

XML, Firewalls, SOA, SOAP, WebServices
And Design Exercises
<XML>
  <Lecture name="Lecture 12" location="wwh101">
    <XSL/>
    <XSLT/>
    <XPath/>
    <XPointer/>
    <WebServices>
      <HTTP/>
      <SOAP/>
      <WSDL/>
      <UDDI/>
    </WebServices>
    <SOA>Service Oriented Architecture</SOA>
    <Security/>
  </Lecture>
</XML>
Definition

- XML: eXtensible Markup Language
  - A language for defining structured documents
  - Broad concept of what a “document” is (more general)
- Who defines XML?
  - A standard defined by the W3C (World Wide Web Consortium)
  - Brings together businesses and researchers
- Documents are edited in text form combining data and tags
  - Tags are written just like in HTML

<mytag>data</mytag>
Sample Hotel Bill

HOTEL NYU SUPREME COURTYARD
1000 Broadway – Manhattan, NY 10010

Number: 99005012
Customer: Poelman
Credit Card: 12345678990  Expires 12/2006
Phone: none
Address: 43 Mercer
NY, NY 10009

Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>single room $340 * 2 nights</td>
<td>$680</td>
</tr>
<tr>
<td>single room $200 * 5 nights</td>
<td>$1000</td>
</tr>
</tbody>
</table>

$1680

Thank you for spending your hard earned cash with us!
<INVOICE number="99005012">
  <CLIENT>
    <NAME>Poelman</NAME>
    <CreditCard number="12345678990" exp-month="12" exp-year="06"/>
    <Phone></Phone>
    <ADDRESS>
      <street>43 Mercer</street>
      <city>NY</city>
      <state>NY</state>
      <zipcode>10009</zipcode>
    </ADDRESS>
  </CLIENT>
  <LINE>
    <PRODUCT>single room</PRODUCT>
    <PRICE>340</PRICE>
    <QUANTITY>2 nights</QUANTITY>
  </LINE>
  <LINE>
    <ITEM type="single room" PRICE="200" QUAN="5 nights"/>
  </LINE>
</INVOICE>
<INVOICE number="99005012">
  <CLIENT>
    <NAME>Poelman</NAME>
    <CreditCard number="12345678990" exp-month="12" exp-year="12"/>
    <Phone/>
  </CLIENT>
  <ADDRESS>
    <street>43 Mercer</street>
    <city>NY</city>
    <state>NY</state>
    <zipcode>10009</zipcode>
  </ADDRESS>
  <LINE>
    <PRODUCT>single room</PRODUCT>
    <PRICE>340</PRICE>
    <QUANTITY>2 nights</QUANTITY>
  </LINE>
  <LINE>
    <ITEM type="single room" PRICE="340" QUAN="2 nights"/>
  </LINE>
</INVOICE>
XML = SGML + Web

- SGML
  - Created at the end of the 1970s based on work by Charles Goldfarb
  - Too complex for the web
- XML respects the SGML philosophy (distinction between data and presentation)
- XML is more flexible than SGML
  - The syntax is simpler (a subset of SGML)
  - Documents can be validated, or checked for being “well constructed”
- XML is richer than SGML
  - DSSSL and HyTime (SGML links) have not had the expected following
  - Hypertext links (XLL = XLink + XPointer)
  - Style sheets (XSL)
- There is an XML-compatible version of HTML: XHTML
What is a DTD?

• Using a DTD makes it possible to check if a document is valid (that is, whether it follows its DTD)
  – Guarantees an exact structure before processing
  – Like a Data Structure Definition in C is to a Data Structure in memory. XML Doc is to a XML Doc DTD

• Explicitly describing a document’s logical structure in the form of a DTD makes it possible to:
  – Offer multiple presentations of a document
  – Analyse a document’s structure in order to “understand” it
  – Process the information it contains

• DTDs are a legacy of SGML
What is a DTD?

<?xml version='1.0'?>
<!DOCTYPE INVOICE [
<!ELEMENT INVOICE (CLIENT, (LINE)+)>
<!ELEMENT CLIENT (NAME,ADDRESS) >
<!ELEMENT NAME (#PCDATA) >
<!ELEMENT ADDRESS (#PCDATA) >
<!ELEMENT LINE (PRODUCT,QUANTITY,PRICE) >
<!ELEMENT PRODUCT (#PCDATA) >
<!ELEMENT QUANTITY (#PCDATA) >
<!ELEMENT PRICE (#PCDATA) >
]>
Inconveniences of a DTD

• Everything is a String or an untyped element i.e. #PCDATA
• Complex syntax without tags (XML)
• No concept of inheritance for enriching or redefining elements
  – Concepts dealing with external physical entities only (inclusion of text)
• DTD history linked to SGML
• No concept of predefined types
• The solution: XML Schemas!
Why a schema?

1.1 Purpose

“The purpose of XML Schema: Structures is to define the nature of XML schemas and their component parts, provide an inventory of XML markup constructs with which to represent schemas, and define the application of schemas to XML documents.

The purpose of an XML Schema: Structures schema is to define and describe a class of XML documents by using schema components to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements and their content and attributes and their values. Schemas may also provide for the specification of additional document information, such as normalization and defaulting of attribute and element values. Schemas have facilities for self-documentation. Thus, XML Schema: Structures can be used to define, describe and catalogue XML vocabularies for classes of XML documents. “

W3C Recommendation

• DTDs were too limited.
• No datatypes, everything was a string.
• DTD weren’t written in XML format and difficult to read and understand.
What is a schema?

```
<xsd:element name = "Invoice">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Client">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name= "Name"/>
            <xsd:element name= "Address"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name = "Line" maxOccurs="15">
        <xsd:complexType>
          <xsd:sequence maxOccurs="unbounded">
            <xsd:element name="Product"/>
            <xsd:element name="Quantity"/>
            <xsd:element name="Price"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Notice – this is in XML format and describes the structure that an XML Doc must have.
<INVOICE number="99005012">
  <CLIENT>
    <NAME>Poelman</NAME>
    <CreditCard number="12345678990" exp-month="12" exp-year="06"/>
    <Phone/>
    <ADDRESS>
      <street>43 Mercer</street>
      <city>NY</city>
      <state>NY</state>
      <zipcode>10009</zipcode>
    </ADDRESS>
  </CLIENT>
  <LINE>
    <PRODUCT>single room</PRODUCT>
    <PRICE>340</PRICE>
    <QUANTITY>2 nights</QUANTITY>
  </LINE>
  <LINE>
    <ITEM type="single room" PRICE="200" QUAN="5 nights"/>
  </LINE>
</INVOICE>
The solutions schemas offer

- The XML Schema predefines various types (integer, date, URI, ...) It is possible to create custom data types
- XML schema takes complete control of namespaces i.e. name collisions are avoided e.g. two schemas may both have an element called Address without forcing a redefinition or ambiguity. Instead you can specify the namespace in front of the element. So you could have:
  - customerSchema:Address and
  - supplierSchema:Address.
Advanced XML & Schema terminology

• **Schema** – Defines the structure that an XML doc must match. Like a **class** definition in OOP or structure of a table in DBMS terminology. Files have a `.xsd` extension.

• **Document** – the **instance** of a schema. Like an **object** is to a class in OOP. Or the actual table data vs. the definition in the DB schema. Files have a `.xml` extension. Documents do not have to reference a schema (or DTD) to be well formed but must reference one or more schemas to be valid.

• **Elements** - `<Tag>value</Tag>` pair, contains one or more sub-elements and/or attributes. Like an **instance variable** in an object. Can contain 0,1, .. sub-elements. Sub elements can contain sub-sub-elements, and so on. A node in the tree that is represented by XML. Elements have a tag, a closing tag, 0,1, .. `n` attributes, 0,1, .. `n` value(s), 0,1, .. `n` sub elements.

• **Attributes** - `<Tag MyAttribute="blue">Value</Tag>` An aspect of an element that only has only one **instance** associated with that element. Ex: `<Person hair_color="brown">Mozart</person>`
The solutions schemas offer

- The ideas behind XML schema are very similar to objects (inheritance, abstract types, extension, aggregation)
  - Like a class def is to an object an XML Doc is to XML Doc Schema
- XML schema is an XML application (similar syntax)
- XML Schemas are usable as a platform and language independent IDL (Interface Definition Language) with extensive capabilities (Beyond CORBA IDL and Java IDL).
Well-formed documents and valid documents

• A well-formed XML document is a document that follows the rules for good construction listed in the specification, including:
  – One single root element in the document containing all others (an XML document is a tree)
  – Having all child elements completely enclosed in parent elements
    • A closing tag for each opening tag

• A valid XML document is a well-formed document that also respects the constraints defined in a DTD or a schema
XML schema

- Predefined datatypes include:
  - Byte
  - Short
  - Long
  - Float
  - Date
  - Time
  - String
  - Time
  - Period
  - Boolean
  - Language
  - Decimal
  - Double float
  - Others ...

- You can extend predefined datatypes or create your own!
- A much better way of defining what an XML document must be structured as that the DTD was.
- Schema = a template that defines the structure, order, and content of any XML documents that reference it.
Designing Schemas

• Schemas often define the Domain Model
  – Abstracting the entities to be represented: contract, order form, brochure, product invoice, manual, catalogue, CV
  – Defining relationships between these entities

• Schemas quality is fundamental for
  – Workflow efficiency and flexibility
  – Expandability and integration of other data sources

• Tools are needed for
  – Designing the data model:
    • UML modeling
    • Database schemas
  – generating Schemas from domain models
  – managing Schemas (sharing Schemas, version control)
Advanced XML & Schema terminology

• **simpleType** – an atomic datatype. ex: string, short, long, date, binary, ...
  Ex:
  ```xml
  <xsd:element name="USPrice" type="xsd:decimal"/>
  ```

• **complexType** – an aggregate datatype. Created in your schema, a sequence of one or more Elements (complexTypes, simpleTypes, elements and/or attributes. Reusable definition of a datastructure. Like a Value Object Definition in Java.
  ex:
  ```xml
  <xsd:complexType name="USAddress" >
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
  </xsd:complexType>
  ```

• **Sequences** – elements that must appear in that specific order in the doc.

• **Choice** – only one of the sub elements is allowed in a doc.

• **Constraints** – minOccurs (default = 1), maxOccurs, (default = 1) optional, required, prohibited, fixed, default. Apply to both elements and attributes.

• **Facets** – aspects of an element or attribute. Ex: white space treatment, precision, ...

• **Patterns** – limits the allowable values by a regular expression syntax e.g. &#XXX,

• **Enumerations** – a limited list of available choices for an element or attribute. Ex: US State abbreviations AL, TX, FL, ...
XML Spy user interface
<?xml version="1.0" encoding="UTF-8"?>

<person>
   <name>ABCD</name>
   <address>ABCD</address>
   <phone>999-999-999-9999</phone>
   <email>ABCD</email>
   <DOB>1999-12-12</DOB>
   <sex>Male</sex>
   <sex>Female</sex>
   <sex>Other</sex>
</person>
Schema generated from PersonDummy.xml by XML spy
Two different ways to structure an XML file that contains information about a person

```xml
<?xml version="1.0" encoding="UTF-8"?>
<person>
  <name>ABCD</name>
  <address>ABCD</address>
  <phone>999-999-999-9999</phone>
  <email>ABCD</email>
  <DOB>1999-12-12</DOB>
  <sex>Male</sex>
  <sex>Female</sex>
  <sex>Other</sex>
</person>
```

```xml
<?xml version="1.0" encoding="UTF-8"?><person name="ABCD" address="ABCD" phone="999-999-999-9999"
email="ABCD" DOB="1999-12-12" sex="Male | Female | Other"/>
```
simpleTypes

- Many built into the schema spec including: binary, short, long, unsignedShort, string, date, dateTime, timePeriod, ...
- You may define new simpleTypes in your schema or extend the existing types.
- Have the following facets:
  - Enumeration list ex: “FR US UK”
  - minLength
  - maxlength
  - Length
  - Required
  - Optional
  - Default
  - Prohibited
  - Many more …
complexTypes

• You may define new complexTypes in your schema or extend already defined complexTypes types.
• An aggregate datatype i.e. contains multiple values of different datatypes. (Much like a C struct).
• Ex:
  <xsd:element name="Person">
  <xsd:complexType>
    <xsd:attribute name= "Alive" type="xsd:boolean"/>
    <xsd:attribute name= "Age" type="xsd:date"/>
  </xsd:complexType>
</xsd:complexType>
• Have the following facets:
  – Enumeration list ex: “FR US UK”
  – minLength
  – maxlength
  – Length
  – Required
  – Optional
  – Default
  – Prohibited
  – Many more …
Generated XML Schema (.xsd file) for an Invoice
Part of schema file text

<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v3.5 NT (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="Address">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Dept"/>
                <xsd:element ref="Building"/>
                <xsd:element ref="Street"/>
                <xsd:element ref="City"/>
                <xsd:element ref="State"/>
                <xsd:element ref="Country"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AreaCode" type="xsd:string"/>
    <xsd:element name="BillTo">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="PartnerName"/>
                <xsd:element ref="Phone"/>
                <xsd:element ref="Address"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
Section of schema file

Shows an enumeration in the schema that limits the allowable values for an attribute (currency) of an element (Total)
<table>
<thead>
<tr>
<th>Simple Type</th>
<th>Examples (delimited by commas)</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Confirm this is electric</td>
</tr>
<tr>
<td>normalizedString</td>
<td>Confirm this is electric</td>
</tr>
<tr>
<td>token</td>
<td>Confirm this is electric</td>
</tr>
<tr>
<td>byte</td>
<td>-1, 127</td>
</tr>
<tr>
<td>unSignedByte</td>
<td>0, 126</td>
</tr>
<tr>
<td>base64Binary</td>
<td>GpM7</td>
</tr>
<tr>
<td>hexBinary</td>
<td>OF87</td>
</tr>
<tr>
<td>integer</td>
<td>-126789, -1, 0, 1, 126789</td>
</tr>
<tr>
<td>positiveInteger</td>
<td>1, 126789</td>
</tr>
<tr>
<td>negativeInteger</td>
<td>-126789, -1</td>
</tr>
<tr>
<td>nonNegativeInteger</td>
<td>0, 1, 126789</td>
</tr>
<tr>
<td>nonPositiveInteger</td>
<td>-126789, -1, 0</td>
</tr>
<tr>
<td>int</td>
<td>-1, 1, 126789, 12678964</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>0, 1, 126789, 12678964</td>
</tr>
<tr>
<td>long</td>
<td>-1, 126789, 126789641233</td>
</tr>
<tr>
<td>unsignedLong</td>
<td>0, 126789, 126789641233</td>
</tr>
<tr>
<td>short</td>
<td>-1, 12678</td>
</tr>
<tr>
<td>unsignedShort</td>
<td>0, 12678</td>
</tr>
<tr>
<td>decimal</td>
<td>-1.23, 0, 123.4, 1000.00</td>
</tr>
<tr>
<td>float</td>
<td>-INF, -1E4, -0, 0, 12.78E2, 12, INF, NaN</td>
</tr>
<tr>
<td>double</td>
<td>-INF, -1E4, -0, 0, 12.78E2, 12, INF, NaN</td>
</tr>
<tr>
<td>boolean</td>
<td>true, false1, 0</td>
</tr>
<tr>
<td>time</td>
<td>13:20:00.000, 13:20:00.000-06:00</td>
</tr>
<tr>
<td>dateTime</td>
<td>1999-06-31T13:20:00.000-06:00</td>
</tr>
<tr>
<td>duration</td>
<td>P1Y2M3DT10H30M12.36</td>
</tr>
<tr>
<td>date</td>
<td>36311</td>
</tr>
<tr>
<td>gMonth</td>
<td>0-12-</td>
</tr>
<tr>
<td>gYear</td>
<td>1999</td>
</tr>
<tr>
<td>gYearMonth</td>
<td>1999-02</td>
</tr>
<tr>
<td>gDay</td>
<td>1999-02-01</td>
</tr>
<tr>
<td>gMonthDay</td>
<td>1999-02-01</td>
</tr>
<tr>
<td>Name</td>
<td>shipTo</td>
</tr>
<tr>
<td>QName</td>
<td>po:USAAddress</td>
</tr>
<tr>
<td>NC Name</td>
<td>USAAddress</td>
</tr>
<tr>
<td>anyURI</td>
<td><a href="http://www.example.com/">http://www.example.com/</a>,</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.example.com/doc.html#DS">http://www.example.com/doc.html#DS</a></td>
</tr>
<tr>
<td>language</td>
<td>en-GB, en-US, fr</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>IDREF</td>
<td></td>
</tr>
<tr>
<td>IDREFS</td>
<td></td>
</tr>
<tr>
<td>ENTITY</td>
<td></td>
</tr>
<tr>
<td>ENTITIES</td>
<td></td>
</tr>
<tr>
<td>NOTATION</td>
<td></td>
</tr>
<tr>
<td>NMTOKEN</td>
<td>US, Brazil</td>
</tr>
<tr>
<td>NMTOKENS</td>
<td>US, UK, Brasil, Canada, Mexico</td>
</tr>
</tbody>
</table>
XML doc and the schema that defines its structure

-- top of XML Doc schema would be here --

```xml
<xsd:element name="Person" type="PersonType"/>
<xsd:complexType name="PersonType">
  <xsd:sequence>
    <xsd:element name="FirstName" type="xsd:string"/>
    <xsd:element name="Middle1Name" type="xsd:string"/>
    <xsd:element name="Middle2Name" type="xsd:string" minOccurs="0"/>
    <xsd:element name="LastName" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

-- middle of schema would be here --

```xml
<xsd:element name="Passengers">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Passenger" type="PersonType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

-- bottom of schema would be here --

```xml
-- Top of XML doc would be here --

<Passengers>
  <Passenger>
    <FirstName>John</FirstName>
    <Middle1Name>Q</Middle1Name>
    <Middle2Name>Q</Middle2Name>
    <LastName>Public</LastName>
  </Passenger>
  <Passenger>
    <FirstName>Jane</FirstName>
    <Middle1Name>I</Middle1Name>
    <LastName>Doe</LastName>
  </Passenger>
</Passengers>

-- bottom of XML doc would be here --
XML namespaces

• Prevents name collisions when including more than one schema in an XML document.
• Every XML document has an unnamed name space.
• Additional name spaces are declared in the top of the XML document by the following syntax:

<yourNameSpace :schema
    xmlns:yourNameSpace="http://www.yourwebsite.org/YourSchema"
    elementFormDefault="qualified">
## Rough analogies between different data manipulation technologies

<table>
<thead>
<tr>
<th>XML</th>
<th>OOP (Java)</th>
<th>Functional (C)</th>
<th>RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>Object Instance with runtime validation of values by mutator methods</td>
<td>Compiled and executing code with Struct and/or local variables with function(s) to set it that has validation code.</td>
<td>Table instance with stored procedure(s) to validate field values (nor integrity checks, though)</td>
</tr>
<tr>
<td>Element</td>
<td>Object Variable may be a finer grained object or a primitive datatype attribute (Int, Short, String, Bool)</td>
<td>Compiled and executing code with Struct and/or local variables with function to set it that has validation code.</td>
<td>Column/field contents and the column name.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Instance of a primitive data type object attribute</td>
<td>Primitive datatype with function to set it that has validation code.</td>
<td>Column/field contents and the column name.</td>
</tr>
<tr>
<td>Schema</td>
<td>Package or top Level Class Definition and mutator methods that validate</td>
<td>Struct definition and/or variables. A struct made up of other struct definitions. Including function(s) to set it that has validation code.</td>
<td>Table Definition in the DB schema (not the whole DB schema)</td>
</tr>
<tr>
<td>Element</td>
<td>Finer grained class definition with mutator method(s) that validate values before setting the attributes of the object.</td>
<td>Finer grained struct definition with a function to set it that has validation code.</td>
<td>Column definition including datatype ex: BirthDate:Date) with a stored procedure to set the value that has validation code.</td>
</tr>
<tr>
<td>Attribute Definition</td>
<td>Primitive type definition with mutator validation code.</td>
<td>i.e. Int MyInt; Primitive type definition with function that validates content before setting it..</td>
<td>Column definition including datatype ex: BirthDate:Date) with a stored procedure to set the value that has validation code.</td>
</tr>
<tr>
<td>simpleType</td>
<td>Primitive type definition …</td>
<td>Primitive type definition …</td>
<td>Column definition including datatype ex: BirthDate:Date) with a stored procedure to set the value that has validation code.</td>
</tr>
<tr>
<td>complexType</td>
<td>Class def that has mulitple attributes that are of different types</td>
<td>A struct that has mixed data types i.e. ints, strings, longs, and/or other struct defs</td>
<td>User defined type?</td>
</tr>
<tr>
<td>XSL file</td>
<td>A transformation and formatting class</td>
<td>A function that converts/formats/embellishes varaiables to be viewed on screen like printf();</td>
<td>A view of one or more tables</td>
</tr>
</tbody>
</table>
XML Benefits and Costs

• Benefits:
  – Integration between two parties – B2B
  – Integration between two languages – EAI
  – Integration between two versions – as software evolves, allows reading earlier versions in later software
  – Easy way to describe a structure for serializing data to/from, instead of creating your own
  – Avoids agreements about persistence data structures
  – Tools and toolkits speed development
  – Standard way of describing the required structure of a data structure, with creating your own language for this – DTD & Schemas

• Costs:
  – Documents get bigger, lots bigger
  – Parsing is slower
  – Verifying is slow – against DTD/Schema
  – Vendors might not make tools support the standard completely.
  – Tools can cost money
  – Increased complexity
Related XML standards

• XSL (eXtensible Stylesheet Language)
  – Defines transformation rules for XML documents (XSLT)
  – In particular, defines how an XML document should be translated into HTML for display.

• DOM (Document Object Model) is an object model
  – Allows for the definition of APIs for accessing and manipulating documents.
  – These APIs exist for Java, Jscript, Perl …

• SAX (Simple API to XML)

• XLL = Xlink + XPointer
  – Allows for links between documents or within documents
  – Allows for navigation (like hypertext in HTML)
  – Makes it possible to go beyond the structured document hierarchical model
Related standards (continued)

- **XHTML**
  - An XML compliant version of HTML
- **XSLFO (eXtensible Stylesheet Language – Formatting Objects)**
  - A true style sheet language (like CSS)
  - Allows physical display constraints to be expressed by defining and introducing XML presentation elements
- **XQL**
  - Query language
  - Uses the Xpath pattern matching language
- **SVG (Scalable Vector Graphics)**
  - Vector graphic language (images, animations…)
  - Adopted by major players in the vector animation industry (Adobe, Macromedia, Microsoft …)
XML definitions

- **DTD**
  The "Document Type Definition" is a part of the original XML 1.0 specification that allows a developer, or standards body, to specify what elements and attributes may be used in a particular type of XML document and what their structure and nesting may be. This is also called the content model or schema of an XML document.

  If an XML document conforms with the content model defined by a DTD, it is said to be valid with respect to that DTD.

- **XSLT**
  The "eXtensible Stylesheet Language Transformation" is a programming language that allows XML documents to be transformed from one schema to another or into entirely different forms, such as HTML pages, WML cards, or PDF files.

- **XPath**
  The "XML Path Language" is a language for addressing and querying the content of XML documents.

- **XLink**
  The "XML Linking Language" describes hyperlinking in XML documents and extends the hyperlinking concepts of HTML.

- **XPointer**
  The "XML Pointer Language" is a companion standard to Xlink and describes mechanisms for addressing particular parts of a document.

- **XML Schema**
  The "XML Schema" is an ongoing effort by the W3C to supplant DTDs with a more flexible and powerful system to describe the structure of conforming XML documents, including provisions for defining datatypes.

- **XHTML**
  The "Extensible HyperText Markup Language" is the reformulation of HTML 4.0 based upon XML and will soon supplant HTML as the de-facto standard of the Internet.
XML definitions

- **WML**
  The "Wireless Markup Language" is used for WAP phone systems to enable a mobile Internet environment and is entirely based on XML - it is described by one particular DTD, which is part of the WML specification.

- **SVG**
  Scalable Vector Graphics. SVG is an XML application used to describe 2D vector graphics, text and raster images. This enables vector graphics to be defined solely in XML.

- **SMIL**
  The "Synchronized Media Integration Language" is a XML document type designed to describe multimedia presentations.

- **DOM**
  The "Document Object Model" describes how some XML parsers return the information contained in an XML document. The elements of the XML document are described as nodes of a tree that can be traversed by a programmer.

- **SAX**
  The "Simple API for XML" provides another programming model used by some parsers, which is based on events instead of a traversable tree.
• Questions
• Comments

• 7.5
XForms
XForms

• XForms are comprised of separate sections that describe what the form does, and how the form looks. This allows for flexible presentation options, including classic XHTML forms, to be attached to an XML form definition.

• Key Goals of XForms
  – Support for handheld, television, and desktop browsers, plus printers and scanners
  – Richer user interface to meet the needs of business, consumer and device control applications
  – Decoupled data, logic and presentation
  – Improved internationalization
  – Support for structured form data
  – Advanced forms logic
  – Multiple forms per page, and pages per form
  – Suspend and Resume support
  – Seamless integration with other XML tag sets

• http://www.w3.org/TR/xforms/
Sample XForm

The xform is MVC - it has a model section, a View (via a CSS / XSL) and Controller (action)

```xml
<HTML xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xforms="http://www.w3.org/2002/01/xforms"
      xmlns:ev="http://www.w3.org/2001/xml-events"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>XForms: Order Form</title>
    <link rel="stylesheet" type="text/css" href="AES.css">
  </head>
  ...
```

```
<table>
<thead>
<tr>
<th>First Name</th>
<th>Initial</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joel</td>
<td>G</td>
<td>Rivera-Gonzalez</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Street Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>253 My Street</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Town</td>
<td>NY</td>
<td>11122</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Total</th>
<th>$ 22.45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax Amount</td>
<td>$ 1.91</td>
</tr>
<tr>
<td>Total</td>
<td>$ 24.36</td>
</tr>
</tbody>
</table>

Shopping Cart

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Item Description</th>
<th>Price p/u ($)</th>
<th>Total ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>White Chocolate Bars</td>
<td>1.45</td>
<td>7.25</td>
</tr>
<tr>
<td>8</td>
<td>Dark Chocolate Bar</td>
<td>1.45</td>
<td>11.60</td>
</tr>
<tr>
<td>2</td>
<td>Caramel Filled chocolate</td>
<td>1.80</td>
<td>3.60</td>
</tr>
</tbody>
</table>

Submit  Cancel
<xforms:model id="data">
  <xforms:instance>
    <OrderInfo>
      <PersonalInfo>
        <Name>
          <First></First>
          <Middle></Middle>
          <Last></Last>
        </Name>
        <Address>
          <Street></Street>
          <City></City>
          <State></State>
          <Zip></Zip>
        </Address>
      </PersonalInfo>
      <PriceInfo>
        <SubTotal></SubTotal>
        <TaxTotal></TaxTotal>
        <TaxRate></TaxRate>
        <Total></Total>
      </PriceInfo>
      <TaxInfo>
        <CT>.060</CT>                  <NY>.085</NY>  <NJ>.083</NJ>
      </TaxInfo>
      <ShoppingCart>
        <ProductInfo name="itm1">
          <Quantity>5</Quantity>
          <Description>Wht. Chocolate Bars</Description>
          <UnitPrice>1.45</UnitPrice>
          <ItemTotal>7.25</ItemTotal>
        </ProductInfo>
        ...
      </ShoppingCart>
    </OrderInfo>
  </xforms:instance>
</xforms:model>
XForms – action (controller) fragment

```xml
<body>
  <xforms:group>
    <div>
      <xforms:input class="First" id="first" ref="OrderInfo/PersonalInfo/Name/First">
        <div>
          <xforms:label>First Name</xforms:label>
        </div>
        <xforms:hint>Enter First Name</xforms:hint>
      </xforms:input>
      <xforms:input class="Middle" id="middle" ref="OrderInfo/PersonalInfo/Name/Name/Middle">
        <xforms:label class="label">Middle Initial</xforms:label>
        <xforms:hint>Enter Middle Initial</xforms:hint>
      </xforms:input>
      <xforms:input class="Last" id="last" ref="OrderInfo/PersonalInfo/Name/Name/Last">
        <xforms:label class="label">Last Name</xforms:label>
        <xforms:hint>Enter Last Name</xforms:hint>
      </xforms:input>
    </div>
  </xforms:group>
</body>
```
SOA & Web Services

Firewalls, Interoperability, HTTP, SOAP, WSDL, UDDI
The Problems

• Provide a mechanism for integrating together systems built on:
  – Different languages
    • Java, Perl, Python, Cobol, C, C++, VB, Delphi, …
  – On different hardware & OSes
    • Win, Linux, Solaris, Mainframe (OS/390), Mac, …
  – Different LANs even between businesses behind firewalls
    • What exactly is a firewall? (in a minute)
The Problems (2)

• How to find services provided by software components
  – Search for them, Like Yahoo!/Google for software applications/components/services
• That is independent of the component model used by the applications communicating
  – RMI, J2EE/EJB, COM, COM+, CORBA, …
SOA (sort of a Design Pattern)

• SOA = Service Oriented Architecture
  – Like the CreditBureau Service we created
• High level “services” that are exposed to many applications in the enterprise
• Should avoid dev language “coupling” i.e. dependence to use the service.
  – Ex: a java service that can be invoked by a PERL script, COBOL, C++
• Services are NOT object oriented (hence the name – “service” oriented)
  – Exchange “data documents” – text based data structures
  – Data structures
  – Messages
  – Value object (just getters/setters - no real behavior)
    • Often exchange major parts of the domain model
    • Don’t exchange complex functionality objects
  – High level / large granularity services – not lower level/functions
SOA (2)

- **Examples services:**
  - GetCreditScore(ApplicantInfo)
  - CalculateTax(AmountOfPurchase)
  - GetProductCatalog()
  - TransferFunds(FundsXferRequestDoc)

- Generally a single parameter (msg or document) is passed in and a single parameter is returned.

- Usually named with a VerbNoun structure to it (as opposed to most OO naming which is Noun.Verb)
  - Ex: GetCreditScore(), UpdateCreditScore(), TransferMoney(), GetUserProfile(), UpdateUserProfile(), SearchForListOfCustomers(), SubmitPurchaseOrder(), GetOrderStatus(), CancelPurchaseOrder(), AddNewUser(), DeleteUser(), FindUser(), AddUserExamGrade()

- Often CRUD(S) operations – Create, Read, Update, Delete, Search

- 2 parts of a service being executed
  - Service Requestor – code making the request (“call”)
  - Service Provider – code handling the request
SOA (3)

• Originally came to prominence in the field of EAI
  – Enterprise Application Integration
• Often, in the past, implemented using asynchronous messaging technologies like:
  – Messaging - IBM MQ Series, TIBCo, JMS
  – Email – SMTP/POP3/IMAP
  – Advantage – requester and provider don’t have to be running at the same time – request/response can be queued!
  – Disadvantages – firewalls, lookup, complexity
• Or File IO, FTP
  – Advantage – simple protocol that’s widely available, kinda queued
  – Disadvantages – firewalls, lookup, management and code you need to write
• Or Sockets
  – Advantage – simple protocol that’s widely available
  – Disadvantages – not queued, firewalls, lookup
• RPC could be used, too
  – RMI, CORBA, COM, EXEC CICS
  – Disadvantages – language coupling and can’t be queued
SOA (3) – asynchronous messaging based
• Questions
• Comments

• 7.75
Firewall

- A firewall regulates access to internal systems, generally HTTP servers.
- Types include:
  - Protocol filtering
  - Address filtering
  - Proxy based
Firewall Types

• Address filtering
  – Allow access to only certain addresses or addresses and ports

• Protocol filtering
  – Allow only the protocols on the address/ports you choose:
    • HTTP, SMTP, FTP, TCP, UDP and/or IP
  – Direction - outside in, inside out or allow both

• NAT – (Native Address Translating)
  – a sort of firewall
  – Maps public IP addresses to internal addresses
  – Allows on public IP to map to multiple internal IPs
Firewall Types

• Proxy
  – An application (proxy) runs between the clients and the servers. The proxy can interrogate every packet that comes in or out and limit the types of packets and content of packets that flow through the firewall.
  – It “fakes” the clients into thinking they are directly talking to the server application. The client requests are “proxy-ed” through this application.
Multi-Firewall DMZ

**FW1**
- **Filter:**
  - Translate IPs
  - Inbound
  - ONLY HTTP
  - ONLY Certain IPs
  - ONLY Port 80
- HTTP Servers
- Each has 2 NICs one for each LAN

**FW2**
- **Filter:**
  - Inbound
  - ONLY TCP
  - ONLY Certain IP Addresses
  - ONLY Port 8080

**FW3**
- Allow only internal HTTP requests to outside servers
- on port 80
- on any address

**DMZ**

- **B2B App**
- **Web Users**
- **Internet**
- **Corp LAN**
- **Mainframe**
- **RDBMS Server**
- **App Server**
- **Internal Users**
- **Behind the firewall**
Firewalls in Corporations

- Most do NOT allow TCP, UDP, or IP through (unless the packets are HTTP over TCP)!
- They allow HTTP on Port 80.
- Often only allowing HTTP clients to be outside and HTTP servers to be inside the firewall.
- How can you have a complex application in one company, access services hosted in a different corporate environment?
  - EJBs?
  - CORBA? COM+? RMI?
  - TCP? UDP? IP?
  - HTTP?
• Questions
• Comments

• 7.75
HTTP to the rescue

• If you package up your request in a HTTP request and pass it to an HTTP server that knows you don’t just want a HTML page returned.

• This is the basic idea of a **WebService**

• Its like the Browser is for humans, it is for application. It allows applications to make GET and POST requests over HTTP to invoke services remotely.
How does WebService (WS) work?

• Like RMI uses serialization to send data, WS uses a marshalling of data from a client data structure/object into an XML format that can be sent over the wire and through a firewall to a HTTP server.

• This is called a SOAP document

• SOAP is an XML document format for marshalling & unmarshalling

• SOAP
<?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/'
    xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
    soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding'/>

<soap:Body>
    <n:getRate xmlns:n='urn:xmethods-CurrencyExchange'>
        <country1 xsi:type='xsd:string'>usa</country1>
        <country2 xsi:type='xsd:string'>japan</country2>
    </n:getRate>
</soap:Body>
</soap:Envelope>
SOAP is:

• Platform, dev language and component model independent
  – you can make a request from a java EJB to a WS that is written in COBOL, if you wanted.
• The serialized form of an data / value object
• Interesting but you as a dev never need directly manipulate it (think about RMI, lots of the work is done for you, same with SOAP.
• You have tools to get your request/response to/from java and SOAP.
• Don’t worry about the details of SOAP, leave that to the WS “plumbers”
Equivalent of RMI Stubs & Skeletons in WebServices (WS)

- Called a **WSDL** file
  - WS Description Language
- Tells clients of a service what the **interface** to the service “looks like”
  - How you can invoke it, what’s it called, what parameters you must send.
- It's another type of XML file
- Generally created via vendor/JDK like java2wsdl.exe that creates a WSDL to match invoking a java file.
- More “plumbing” for WS
<?xml version='1.0' encoding='UTF-8'?>
<!--generated by GLUE on Tue May 21 10:39:36 CDT 2002-->  
<wsdl:definitions name='examples.publish.Exchange'  
targetNamespace='http://www.themindelectric.com/wsdl/examples.publish.Exchange/' 
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' 
xmlns:http='http://schemas.xmlsoap.org/wsdl/http/' 
xmlns:mime='http://schemas.xmlsoap.org/wsdl/mime/' 
xmlns:xsd='http://www.w3.org/2001/XMLSchema' 
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/' 
xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/' 
xmlns:tme='http://www.themindelectric.com/'>

<wsdl:message name='getRate0In'>  
<wsdl:part name='country1' type='xsd:string'> 
  <wsdl:documentation>The country to convert from</wsdl:documentation>  
</wsdl:part>  
<wsdl:part name='country2' type='xsd:string'>  
  <wsdl:documentation>The country to convert to</wsdl:documentation>  
</wsdl:part>  
</wsdl:message>

(Continued …)
<wsdl:message name='getRate0Out'>
  <wsdl:part name='Result' type='xsd:float'>
    <wsdl:documentation>The exchange rate</wsdl:documentation>
  </wsdl:part>
</wsdl:message>

<wsdl:portType name='examples.publish.Exchange'>
  <wsdl:operation name='getRate' parameterOrder='country1 country2'>
    <wsdl:documentation>Return the exchange rate between two countries</wsdl:documentation>
    <wsdl:input name='getRate0In' message='tns:getRate0In'/>
    <wsdl:output name='getRate0Out' message='tns:getRate0Out'/>
  </wsdl:operation>
</wsdl:portType>
WSDL (CONTINUED...)

....
<wsd1:binding name='examples.publish.Exchange' type='tns:examples.publish.Exchange'>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
  <wsdl:operation name='getRate'>
    <soap:operation soapAction='getRate' style='rpc'/>
    <wsdl:input name='getRate0In'>
      <soap:body use='encoded'
        namespace='http://tempuri.org/examples.publish.Exchange'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding'/>
    </wsdl:input><wsdl:output name='getRate0Out'>
      <soap:body use='encoded'
        namespace='http://tempuri.org/examples.publish.Exchange'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding'/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding><wsdl:service name='examples.publish.Exchange'>
  <wsdl:port name='examples.publish.Exchange' binding='tns:examples.publish.Exchange'>
    <soap:address location='http://192.168.1.107:8004/glue/exchange'/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
WSDL is …

- Platform, dev language and component model independent
  - you can have the same WSDL generated by a WS implemented in java and implemented in COBOL if you wanted.
  - It defines the interface NOT the implementation

- Like the Stub & Skeleton created by RMIC
- Like a CORBA IDL or COM typelib/IDL
- You shouldn’t EVER have to manually edit these. Tools should manipulate them for you.
- Ignore the details in most cases. Its just plumbing like RMI is.
How do you “find” a WS?

• Somebody gives you the WSDL for it or …
• Look up its WSDL in a UDDI directory!
• UDDI – Universal Description Discovery & Integration
• Like a phone book for WS WSDLs.
  – White Pages – by service name
  – Yellow Pages – by service provider
  – Green Pages – by service type
• Like yahoo! Or Google for WS WSDLs – used by requestors.
• Like JNDI for Java or CosNaming for CORBA or the Registry / AD for COMCOM+
• UDDI can be public (on the Internet) or private.
UDDI Directory

• A HTTP based process that is published on a well known port and address
• Gets a SOAP request to look up a service(s), returns a response with the information about services (WSDLs)
• Not required. You can just point directly to a WSDL file to get the info needed to invoke a WS.
Creating web services in java. You have many choices:

- The SUN Web Services Developer Pack
  - Built into the JDK
  - The kitchen sink and more
  - Complex to just run a web service

- Thirdparty tool sets
  - WASP
  - Glue
  - ...
• Questions
• Comments

• 8.0
WS Examples
A CreditBureauService that is published as a webservice

• I created the following files:
  - ICreditBureauService
  - CreditBureauService
  - PublishAndStartCreditBureauService
  - InvokeCreaditBureauService
Invoking a webservice
If the service isn’t running here’s what the client does
Web Service invoked via the servlet
• Questions
• Comments

• 8.25
Selected code from the servlet

```java
import electric.registry.Registry;

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException {
    response.setContentType("text/html");
    java.io.PrintWriter out = response.getWriter();
    out.println("<html>");

    try {
    // bind to web service whose WSDL is at the specified URL
    String url = "http://localhost:8004/myWebServices/CreditBureauService.wsdl";
    ICreditBureauService creditBureauService =
        (ICreditBureauService) Registry.bind( url, CreditBureauService.class );

    LName = "Chang";  FName = "Isiu";
    creditScore = cBS2.getCreditScore(FName,LName);
    out.println( "Credit score by Name through CBS2 [ " + FName + "" + LName + "] = " + creditScore +"<br>");
    creditHistory = cBS2.getCreditHistory(FName,LName);
    out.println( "Credit history = " + creditHistory);
    }
    catch(Exception e)
    {
    out.println("Exception = " + e);
    }
```
<wsdl:message name='getCreditScore0In'>
  <wsdl:part name='SSNumber' type='xsd:string'/>
</wsdl:message>

<wsdl:message name='getCreditScore0Out'>
  <wsdl:part name='Result' type='xsd:float'/>
</wsdl:message>

<wsdl:message name='getCreditScore1In'>
  <wsdl:part name='FName' type='xsd:string'>
    <wsdl:documentation>The persons first name</wsdl:documentation>
  </wsdl:part>
  <wsdl:part name='LName' type='xsd:string'>
    <wsdl:documentation>The persons last name</wsdl:documentation>
  </wsdl:part>
</wsdl:message>

<wsdl:message name='getCreditScore1Out'>
  <wsdl:part name='Result' type='xsd:float'>
    <wsdl:documentation>The credit score. A float between 0 and 800. If -1 no credit score is available</wsdl:documentation>
  </wsdl:part>
</wsdl:message>

<wsdl:message name='getCreditHistory2In'>
  <wsdl:part name='SSNumber' type='xsd:string'/>
</wsdl:message>

<wsdl:message name='getCreditHistory2Out'>
  <wsdl:part name='Result' type='xsd:string'/>
</wsdl:message>
<wsdl:portType name='CreditBureauService'>
  <wsdl:operation name='getCreditScore' parameterOrder='SSNumber'>
    <wsdl:input name='getCreditScore0In' message='tns:getCreditScore0In'/>
    <wsdl:output name='getCreditScore0Out' message='tns:getCreditScore0Out'/>
  </wsdl:operation>
  <wsdl:operation name='getCreditScore' parameterOrder='FName LName'>
    <wsdl:documentation>Return the score for an individual.</wsdl:documentation>
    <wsdl:input name='getCreditScore1In' message='tns:getCreditScore1In'/>
    <wsdl:output name='getCreditScore1Out' message='tns:getCreditScore1Out'/>
  </wsdl:operation>
  <wsdl:operation name='getCreditHistory' parameterOrder='SSNumber'>
    <wsdl:input name='getCreditHistory2In' message='tns:getCreditHistory2In'/>
    <wsdl:output name='getCreditHistory2Out' message='tns:getCreditHistory2Out'/>
  </wsdl:operation>
  <wsdl:operation name='getCreditHistory' parameterOrder='FName LName'>
    <wsdl:documentation>Return the history for an individual.</wsdl:documentation>
    <wsdl:input name='getCreditHistory3In' message='tns:getCreditHistory3In'/>
    <wsdl:output name='getCreditHistory3Out' message='tns:getCreditHistory3Out'/>
  </wsdl:operation>
</wsdl:portType>
The WSDL created ...(3)

```xml
<wsdl:operation name='getCreditScore'>
    <soap:operation soapAction='getCreditScore' style='rpc'/>
    <wsdl:input name='getCreditScore1In'>
        <soap:body use='encoded' namespace='http://tempuri.org/CreditBureauService' encodingStyle='http://schemas.xmlsoap.org/soap/encoding'/>
    </wsdl:input>
    <wsdl:output name='getCreditScore1Out'>
        <soap:body use='encoded' namespace='http://tempuri.org/CreditBureauService' encodingStyle='http://schemas.xmlsoap.org/soap/encoding'/>
    </wsdl:output>
</wsdl:operation>
```
Types of WS invocations

- **RPC** (Remote Procedure Call)
  1. Client requests
  2. sends SOAP request via HTTP
  3. service gets SOAP request
  4. processes it and returns a SOAP response to client
  5. client receives the SOAP response and continues execution.
     - Blocks waiting for a response
     - Not very scalable
Types of WS invocations

- **Doc** (document)
  - Client requests
  - sends SOAP request via HTTP and doesn’t block.
  - Client also has a portion of code that acts as a WS service provider but to receive responses. It waits for a response to come in and processes it asynchronously.
  - More scalable
    - Harder to get through a firewall because clients and servers exist on both sides of the firewall.
WS info

- A set of web standards currently
- Not all vendor implementations are interoperable!
- Over hyped!!!
- Not a replacement within apps for RMI/COM+/CORBA
- No management technology of WS (unlike EJBs yet in place) Need a Web Service Server! Or WS Manager.
- WS are really interfaces to services hosted as EJBs, Servlets, java processes, COM objects, etc.
- Most EJB servers have added WS features
- Not required to be HTTP!
  - JMS, Email (SMTP), TCP, ?
www.grandcentral.com

a site for webservices integration and security, ...
XMethods (www.xmethods.com)

a web site devoted to publicly published web services
Current Limitations of WS

• Very limited security capabilities – See WS-Security standards
• Non-transactional – HTTP can’t be transactional by its very nature
• Waiting for a “killer app”
• Doc version and firewall configurations are still incompatible
• Not many really understand the value and power – most sending objects over WS, should just send data!
• HTTP only for the most part
WSDP is …

- JAXM and SAAJ
- JAXP
- JAXR
- JAX-RPC
- JSTL
- Tomcat
- Ant
- Registry Server
Summary

- WebService =
  - HTTP
  - SOAP (which is XML)
  - WSDL (which is XML)
  - UDDI (which is a webservice based interface to a directory of web services)

- Gets through firewalls
- RPC vs DOC
- Independent of platform and dev language!
- Non transactional and weak security
- Web Services are “da bomb diggity” of technology right now! (Very over hyped)
• Questions
• Comments

• 8.50
Other Java Items of Interest

• Applets
• JNLP – Java Network Launch Protocol
• Portal
Other Java Items of Interest
Applets
Applet - upside

- Downloaded via web and runs in the browser
- Runs in a “sandbox”
  - No access to client machines file system, IO
  - No ability to open sockets to another other IP but server it was downloaded from
  - Very safe from hacking
- Zero install of app on client
- Can be a Swing UI – more feature rich than standard web apps
- Always gets the latest version of the app
- Applet has a lifecycle
- Applets can start threads that keep running even when not visible.
Applet - downside

- Depends on the JVM version in the browser – problems with IE by default – blame microsoft
  - Java browser plugin from Sun can fix this but requires install
- Could get downloaded every time – if big, could be slow
- Runs in a “sandbox”
  - You can download signed applets and allow access but extra work.
- No used much anymore because of incompatibilities
- Might be suitable for internal only applications when you have control of the users installed browser
http://www.timeanddate.com/worldclock/personalapplet.html
Special HTML commands

<applet code="personalclock.class" width=610 height=320 alt="Personal Clock Applet" align=center>
<param name="cities" value="438,179,104,136">
<param name="fhr" value="0">
<param name="ftz" value="0">
<param name="fhm" value="">
</applet>
Applet has a lifecycle controlled by the browser

```java
public class MyApplet extends Applet
{
    // page is loaded with applet - like the constructor
    void init() { . . . }

    // page is becomes visible or revisible
    public void start() { . . . }

    // page becomes non-visible
    public void stop() { . . . }

    // browser window closed
    public void destroy() { . . . }
}
```

Sample applets

JNLP & Java Web Start
JNLP – Java Network Launch Protocol & Java Web Start

• Gets around some problems with the applet approach.
• Allows zero install applications that get downloaded and executed
• Java Web Start allows full-featured applications to be launched with just a single-click of a mouse.
• You will need to have Java Web Start installed in order for the applications to launch. If it is not installed, you will be redirected to the Java Web Start setup page. (part of the JRE install)
  • launchLink("images/player.jpg", "apps/player.jnlp", 134, 125);
  • http://java.sun.com/products/javawebstart/demos.html
Example Web Start

Desktop Integration provides a quick and easy way to access your application. Would you like to have "Draw 4 App" integrated into your desktop environment?

Yes  No  Ask Later  Configure...

Copyright (c) Poelman All Rights Reserved
Portal
Portal Features

• Aggregated components create a single desktop – like a standard windows desktop that is running multiple applications

• User can choose what components are shown on their desktop and where. Pick them from a list of components.

• Portlets (special versions of servlets) can communicate with each other on the page to do special things – portlets are like UI building blocks
Regular Desktop With Non-overlapping Windows vs. Portal Desktop
Example Portal User Interface
Example Portal User Interface
Define a Portal

Features that may/may not define a portal:

- **Aggregation** – two or more web pages on a single page
- **Customization** - user can change content sources (web pages & applications) and layout of their personal view.
- **Standardized but changeable “look & feel”**
- **Searching**
  - Federated Search - search across multiple databases.
  - Mixed content type search - images, structured, semistructured and structured data.
  - Meta data enabled search.
- **Personalization** - site changes content and layout to match rules set up by administrator. These rules can take into account the specific user’s preferences, previous behavior, date, special offers, etc.
- **Portlet to Portlet communication** - portlets are the parts of the portal view that contains a single content source/application.
How can you build a portal

• What if you have one servlet that invokes multiple other servlets via a RequestDispatcher include() method. Each of those included servlets just returns a portion of the page – an HTML page fragment – the part they “own”

• You’d create a single page out of smaller view components to create and aggregated view
Portal “desktop” view
Particular “parts” of a portal desktop
• Questions
• Comments

• 8.75
Sample Design Problems
Sample design problems

- Instant Messenger
- VOIP
- Distance Learning Tools
- Money Transfer System
- Bookstore
- Dating Site
- Travel / Hotel booking site
- Language Translation for Documents
- Wordprocessor – desktop versus web based
Instant Messenger

• Create an application that allows users to text chat with each other via their computers
• User to User communication 1:1
• Behind a Firewall
• Find People and see when they are online/offline
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, ...
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, …
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
VoIP – (Voice over IP) – Internet Phone

• Create a system that allows one user to talk to another user, from their machine using speakers and a microphone as though it’s a telephone (like Skype)

• Lookup a number, know when your friends are online, block calls, handle firewalls
Quick Analysis - Design Criteria

• Major Entities in Domain Model (Classes)
• Users
  – Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
• Security Requirements
  – Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
• Value of operations
  – Zero, a few cents, a few dollars, many dollars, …
• Availability and Scalability
  – 24 x 7 x 365?, 10x5x250,
  – User population growth over time?
• Application Structure
• Dev Language
  – Java, PERL, C#, PHP, …
• UI
  – Command Line, Swing App, WebUI, Swing Applet
• Communications Protocols
  – IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
• Design Patterns
Distance Learning

- Professor wants to teach a course over the internet
  - Share the “slides” with each student
  - See the Professor via Video window
  - Allow prof to write on the slides
  - Hear the prof on their speakers
  - Students raise hand to ask questions via text chat
  - Professor would like to be able to quiz students in a popup dialog that appears on the students screens and automatically grade the results of those answers to see whether the class is getting it/or not
  - Max 50 registered users (login and password)
  - Web based UI!
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, …
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, …
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
Money Transfer System

• Transfer money from banks accounts in different banks
• Used by bank employees
• Dedicated network (not the internet)
• Transfers up to $10,000,000
• 500 transfers a day, average
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, ...
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, ...
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
Bookstore

- Browse Books
- Select, Add, Remove from Cart
- Purchase
- Check on order see if books are shipped yet
- Allow using credit cards for purchase
- Purchases between $10.00 to $1000 per customer
- Potentially >1000 customers per day
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, …
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, …
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
Dating Site

- Create a profile
- Find matching profiles
- Get automated matches weekly
- Send anonymous emails between you and other users
- Keep out “kooks”
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, …
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, …
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
Travel / Hotel booking site
Quick Analysis - Design Criteria

• Major Entities in Domain Model (Classes)
• Users
  – Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
• Security Requirements
  – Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
• Value of operations
  – Zero, a few cents, a few dollars, many dollars, …
• Availability and Scalability
  – 24 x 7 x 365?, 10x5x250,
  – User population growth over time?
• Application Structure
• Dev Language
  – Java, PERL, C#, PHP, …
• UI
  – Command Line, Swing App, WebUI, Swing Applet
• Communications Protocols
  – IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
• Design Patterns
Language Translation for Documents
Quick Analysis - Design Criteria

• Major Entities in Domain Model (Classes)
• Users
  – Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
• Security Requirements
  – Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
• Value of operations
  – Zero, a few cents, a few dollars, many dollars, …
• Availability and Scalability
  – 24 x 7 x 365?, 10x5x250,
  – User population growth over time?
• Application Structure
• Dev Language
  – Java, PERL, C#, PHP, …
• UI
  – Command Line, Swing App, WebUI, Swing Applet
• Communications Protocols
  – IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
• Design Patterns
Wordprocessor – desktop versus web based

• Edit text, change fonts, colors, embed images, sounds, video
• Save on a central server not local to client desktop
• Get at my docs no matter what computer I am on
Quick Analysis - Design Criteria

- Major Entities in Domain Model (Classes)
- Users
  - Location, Trust Level, Training on App, Frequency of use, Total, Simultaneous, Request per Minute
- Security Requirements
  - Id, Authentication, Authorization, Encryption, Signatures, Non-repudiation
- Value of operations
  - Zero, a few cents, a few dollars, many dollars, …
- Availability and Scalability
  - 24 x 7 x 365?, 10x5x250,
  - User population growth over time?
- Application Structure
  - Single App, Client/Server, P2P, WebApp, Combo?
- Dev Language
  - Java, PERL, C#, PHP, …
- UI
  - Command Line, Swing App, WebUI, Swing Applet
- Communications Protocols
  - IP, TCP, UDP, FTP, SMTP/POP/IMAP, HTTP, RMI, WebServices
- Design Patterns
The End

• Thanks for your questions, comments and efforts
• Please feel free to send me updates about how you use what you have learned in this class in your future endeavors
• Feel free to send me feedback – good and bad
resources

• WebServices Blueprints
  • http://java.sun.com/blueprints/webservices/using/webservbp.html
  • http://java.sun.com/webservices/webservicepack.html
  • http://www.soapclient.com/uddisearch.html
  • http://www.xmethods.com/