Homework & Final!

• All Homework is Due by 5AM EST today!
• The final is next week, Dec 15th!
Final Exam

- Dec 15th 7-9pm
- WWH 109 (Here)
- Open Book – no computers
- All the books and printed material you wish to bring
- It is longer (approx 70 questions)
- It is probably a more difficult exam than the midterm
- 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 – some (not all) of what you should understand

- **Java**
  - OO features, IO, Sockets, Servlets, JSP, EJBs
  - Different libraries / specs
  - Swing, Collections, Threading, RMI, Callbacks, Serialization, Reflection, JVM concepts

- **IP, TCP, UDP, HTTP, BEEP, …**

- **XML, HTML, UML**

- **Security**
  - Concepts – IAAEDN – Id, Authen, Authorization, Encryption, Digital Signatures, Non-repudiation
  - Protocols & Tech – SSL, PK, Private Key, Digital
Final – some (not all) of what you should understand (2)

• WebServices
  – HTTP, Firewalls, Transactionality, Security
  – SOAP, WSDL, UDDI, ..
  – Tools – Glue, WSDK, …
  – Value and purpose

• OO
  – Inheritance, Encapsulation, Polymorphism, …
  – Visibility
  – Implementation and Value in design an implementation
Final – some (not all) of what you should understand (2)

• Web Concepts
  – Session, application, stateless, connectionless, cookies, URL,
  – GET, POST, …
  – CGI vs Servlets
  – Protocols – HTTP, HTTPS, TCP, FTP, SMTP, DNS, …

• Putting it all together to solve a problem i.e. creating an application to solve a real world problem.
Class Grade Calculation

• Grades will be calculated as follows:
  – \[ X = \text{number of points you received on the midterm} \]
  \[ Y = \text{number of points you received for all the homework you submitted} \]
  \[ Z = \text{number of points you scored on the final} \]
  – Grade Point Index (GPI) = \[ \frac{X}{50} \times 30 + \frac{Y}{30} \times 30 + \frac{Z}{70} \times 40 \]
• If your Grade Index is:

<table>
<thead>
<tr>
<th>Range</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 &gt;= GPI &gt;= 94</td>
<td>A</td>
</tr>
<tr>
<td>94 &gt; GPI &gt;= 90</td>
<td>A-</td>
</tr>
<tr>
<td>90 &gt; GPI &gt;= 87</td>
<td>B+</td>
</tr>
<tr>
<td>87 &gt; GPI &gt;= 84</td>
<td>B</td>
</tr>
<tr>
<td>84 &gt; GPI &gt;= 80</td>
<td>B-</td>
</tr>
<tr>
<td>80 &gt; GPI &gt;= 77</td>
<td>C+</td>
</tr>
<tr>
<td>77 &gt; GPI &gt;= 74</td>
<td>C</td>
</tr>
<tr>
<td>74 &gt; GPI &gt;= 70</td>
<td>C-</td>
</tr>
<tr>
<td>70 &gt; GPI &gt;= 64</td>
<td>D</td>
</tr>
<tr>
<td>64 &gt; GPI</td>
<td>F</td>
</tr>
</tbody>
</table>
Class grade calculation example:

- If I received a 45 of 50 on the midterm (X=45), I received 25 of 30 points on the homework (Y=25) and I received a 60 of 70 on the final (Z = 60) my grade index is:
  - \[(X / 50) \times 30 + (Y/30) \times 30 + (Z/70) \times 40\]
  - \((45/50) \times 30 + (25/30) \times 30 + (60/70) \times 40\)
  - \(27 + 25 + 34.3 = 86.3\)
  - That means I will receive a grade of “B”
I will POST [some of] those slides to the website so you don’t have to write it all down.

You may start taking notes again.

;-)
Quick Tech Demos
The Digital Pen

• The DigitalIO Pen
• www.logitech.com
• www.anoto.com
Technical data

**Camera:** Custom CMOS sensor, >50 FPS

**Processor:** Proprietary ARM-based ASIC at 70 MHz

**Communication device:** ex. Bluetooth transceiver

**Battery:** TBA

**Ink cartridge:** Standard DIN 16654

**Illumination:** IR LED

**Resolution:** 0.03 mm

**Weight:** 45g

For left and right handed use
The handwritten digital message from paper to recipient only takes 20 sec.

1. Text written with the pen is forwarded via a Bluetooth mobile phone to the Paper Look-up Service (PLS).

2. The PLS unit returns an instruction to the pen, telling it which service provider to contact.

3. The information from the pen is sent to the server at the provider supplying the service in question.

4. The message is sent from the service provider to the recipient.
Fill in the e-mail address or fax number, and tick the "Send" box. Your message is instantly transmitted to the receiver - for instance via a mobile phone.
© 2003 Anoto

- “Pidgets” pen-widgets that act like buttons on a UI
- ICR – Intelligent Character Recognition converts block letters to text – as the file name of the image

To

Ø0o. 1L1 2Z gyj
(continue from line above)

Send
How does it work?

- A proprietary pattern of very small dots printed on paper that is perceived by the eye as a slightly off-white color. The dots have a nominal spacing of 0.3 mm (0.01 inch). A minute portion of the total pattern uniquely defines its position in the full pattern.
- 60,000,000 square kilometres, which is equivalent to an area exceeding that of Europe and Asia combined.
- When writing with a digital pen on a paper printed with the pattern, digital snapshots of the pattern are taken with a rate of more than 50 per second.
- Every snapshot contains enough information to make a calculation of the exact position of the pen. The intelligence in the paper, derived from the pattern, makes it possible to perform operations by just ticking a box with the pen, e.g. Store, Send, To Do, Address, etc.
- IR wavelengths 800-950 nm are what the dots are read in
© 2003 Anoto
TabletPC

Gateway® Tablet PC
TabletPC

- XP Tablet Version
- Pen acts as a mouse
- Handwriting recognition
- Good for some things
- Not so good (stinks) for others
- 1024 x 768 screen and pen resolution
- Pressure sensitive
It doesn’t make you an artist!

This was written by me.
How good is the handwriting recognition?

- This was written using the write anywhere handwriting to text conversion Feature thus were actually cursive curter by me. Maybe Goop For Medical Daters?

- <<the above should have said “This was written by me using the write anywhere handwriting to text conversion features. This was actually written cursive by me. Maybe its good for medical doctros?”>>
Tools
XML, XSL, XSLT, XPath, XPointer and more
<XML> in detail
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

```html
<mytag>data</mytag>
```
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
Structured documents: definition

- A structured document consists of distinct logical elements.
- The composition of these elements defines a characteristic tree structure for the document.
- Bolstered by the distinction between:
  - The document’s logical structure: its division into elements
  - The document’s physical structure: saved in several files.
- The SGML standard was designed to allow for standardised description of logical structures.
Example of logical structure

```
<INVOICE>
  <CLIENT>
    <NAME>Poelman</NAME>
    <ADDRESS>43 Mercer NY, NY 1009</ADDRESS>
  </CLIENT>
  <LINE>
    <PRODUCT>single room</PRODUCT>
    <PRICE>340</PRICE>
    <QUANTITY>2 nights</QUANTITY>
  </LINE>
</INVOICE>
```

Invoice.xml

Hotel de la NYU

Greenwich Village, NY

Invoice n° 99005012

Addressed to:
Poelman
43 Mercer
NY, NY 1009

<table>
<thead>
<tr>
<th>description</th>
<th>PU HT</th>
<th>Quantity</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single room</td>
<td>340</td>
<td>2 nights</td>
<td>680</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td></td>
<td></td>
<td><strong>680</strong></td>
</tr>
</tbody>
</table>

Payment by cheque upon receipt
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

• 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

- Untyped elements (everything is #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: 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."

- 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?

- Example:

```xml
<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>
```

Invoice.xsd
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:
- The ideas behind XML schema are very similar to objects (inheritance, abstract types, extension, aggregation)
- 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:
  - Having all child elements completely enclosed in parent elements
    - One single root element in the document containing all others (an XML document is a tree)
    - 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
Designing Schemas

• Schemas define the information systems Data 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 analysis models
  – managing Schemas (sharing Schemas, version control)
XML schema

• A much better way of defining what an XML document must be structured as that the DTD was.
• It's an XML document itself.
• A template that defines the structure, order, and content of any XML documents that reference it.
• XML tag/value pairs can have a datatype associated with them. Thus enforcing validation of the syntax, structure and semantic content of an XML doc.
• Predefined datatypes include:
  - Byte
  - Short
  - Long
  - Float
  - Date
  - Time
  - String
  - Period,
  - Time
  - Boolean
  - Language
  - Double float
  - Others ...

• You can extend predefined datatypes and create your own!
Advanced XML & Schema terminology

From the W3C primer:

• Definition: A Schema component is the generic term for the building blocks that comprise the abstract data model of the schema. An XML Schema is a set of schema components. There are 13 kinds of component in all, falling into three groups. The primary components, which may (type definitions) or must (element and attribute declarations) have names are as follows:
  – Simple type definitions
  – Complex type definitions
  – Attribute declarations
  – Element declarations

• The secondary components, which must have names, are as follows:
  – Attribute group definitions
  – Identity-constraint definitions
  – Model group definitions
  – Notation declarations

• Finally, the "helper" components provide small parts of other components; they are not independent of their context:
  – Annotations
  – Model groups
  – Particles
  – Wildcards
  – Attribute Uses
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 a class in OOP. Can contain 0, 1, .. Inf 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 one instance associated with that element. Ex: `<Person hair_color="brown">Mozart</person>`
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 `Struct` in C.
  
  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
Schema generated from PersonDummy.xml by XML spy
Two different ways to structure an XML file that contains information about a person

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Logan Poelman -->
<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 version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Logan Poelman -->
<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:

```xml
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:attribute name="Alive" type="xsd:boolean"/>
    <xsd:attribute name="Age" type="xsd:date"/>
  </xsd:complexType>
</xsd:element>
```

• 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).
### Partial list of simple types supported in of XML schema 1.00

<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, 126</td>
</tr>
<tr>
<td>unsignedByte</td>
<td>0, 126</td>
</tr>
<tr>
<td>base64Binary</td>
<td>0mP7</td>
</tr>
<tr>
<td>hexBinary</td>
<td>0FED</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</td>
</tr>
<tr>
<td>int</td>
<td>-1, 12678998576</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>0, 1267899857694</td>
</tr>
<tr>
<td>long</td>
<td>-1, 1267899857643233</td>
</tr>
<tr>
<td>unsignedLong</td>
<td>0, 1267899857643233</td>
</tr>
<tr>
<td>short</td>
<td>-1, 1267899857643233</td>
</tr>
<tr>
<td>unsigned8Short</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, 12.78E-2, 12, INF, NaN</td>
</tr>
<tr>
<td>double</td>
<td>-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN</td>
</tr>
<tr>
<td>boolean</td>
<td>true, false</td>
</tr>
<tr>
<td>time</td>
<td>13:20:00.000.000.000-05:00</td>
</tr>
<tr>
<td>dateTime</td>
<td>1999-06-31T13:20:00.000-05:00</td>
</tr>
<tr>
<td>duration</td>
<td>P1Y2M3DT10H30M12.9S</td>
</tr>
<tr>
<td>date</td>
<td>36511</td>
</tr>
<tr>
<td>gMonth</td>
<td>-06-</td>
</tr>
<tr>
<td>gYear</td>
<td>1999</td>
</tr>
<tr>
<td>gYearMonth</td>
<td>1999-02</td>
</tr>
<tr>
<td>gDay</td>
<td>-31</td>
</tr>
<tr>
<td>gMonthDay</td>
<td>-29</td>
</tr>
<tr>
<td>Name</td>
<td>shipTo</td>
</tr>
<tr>
<td>CQName</td>
<td>po: USA Address</td>
</tr>
<tr>
<td>NC Name</td>
<td>USA Address</td>
</tr>
<tr>
<td>anyURI</td>
<td><a href="http://www.example.com/">http://www.example.com/</a>, <a href="http://www.example.com/doc.htmMDE">http://www.example.com/doc.htmMDE</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>IDREF2</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>INMTOKEN</td>
<td>US, Brazil</td>
</tr>
<tr>
<td>INMTOKEN8</td>
<td>US, UK, Brésil Canada Mexico</td>
</tr>
</tbody>
</table>
XML doc and the schema that defines its structure

-- - *top of 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* - - -

```
```

-- - *top of XML doc would be here* - - -

```xml
    <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
```
XML namespaces

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

```xml
<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 Definition</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 multiple 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 variables to be viewed on screen like printf();</td>
<td>A view of one or more tables</td>
</tr>
</tbody>
</table>
Related 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
<XMLE> Presentation Layer Technologies
XSL, XSLT, XForms
XSL = Extensible Style sheet Language

- XSL is a language enabling the creation of a target document from an XML source document
- The XSL processor uses style sheets to transform XML documents
- Unlike CSS, XSL can serve to transform, filter, and format data
- XSL is based on XPATH (formalized addressing)
- http://www.w3.org/Style/XSL/
More and more businesses need to integrate new presentation modes

Appearance of new formats and the need to ensure the continuity of existing ones

- AudioTel
- WAP, PDA
- HTML browsers
- Network Computer
- Minitel (in France)
XSL for multi-channel broadcasting (2/2)

- Separating data from presentation with an XSL Formatting Object

XML Document -> XSL Processor

- WML Style Sheet
- HTML, VoxML Style Sheet
- Postscript Style Sheet
- XML Style Sheet

WAP Terminal -> HTML browser

HTML Document -> PDF Document

Specific XML Document -> Partner B2B site
XSL

- Requires that your browser have an XSL engine built in. For IE is `msxml.dll`
- Microsoft Internet Explorer 5.0 or later.
<?xml version="1.0" ?>
<sales>
  <summary>
    <heading>Greenwich Village Book Publishing</heading>
    <subhead>Regional Sales Report</subhead>
    <description>Sales report for the West Coast, Central and East Coast regions.</description>
  </summary>
  <data>
    <region>
      <name>West Coast</name>
      <quarter number="1" books_sold="24000" />
      <quarter number="2" books_sold="38600" />
      <quarter number="3" books_sold="44030" />
      <quarter number="4" books_sold="21000" />
    </region>
    <region>
      <name>Central</name>
      <quarter number="1" books_sold="11000" />
      <quarter number="2" books_sold="16080" />
      <quarter number="3" books_sold="25000" />
      <quarter number="4" books_sold="29000" />
    </region>
    <region>
      <name>East Coast</name>
      <quarter number="1" books_sold="27000" />
      <quarter number="2" books_sold="31400" />
      <quarter number="3" books_sold="40100" />
      <quarter number="4" books_sold="30000" />
    </region>
  </data>
</sales>
Sales.xml in IE

```xml
<?xml version="1.0" ?>
<sales>
  <summary>
    <heading>Greenwich Village Book Publishing</heading>
    <subhead>Regional Sales Report</subhead>
    <description>Sales report for the West Coast, Central and East Coast regions.</description>
  </summary>
  <data>
    <region>
      <name>West Coast</name>
      <quarter number="1" books_sold="24000" />
      <quarter number="2" books_sold="38600" />
      <quarter number="3" books_sold="44030" />
      <quarter number="4" books_sold="21000" />
    </region>
    <region>
      <name>Central</name>
      <quarter number="1" books_sold="11000" />
      <quarter number="2" books_sold="16080" />
      <quarter number="3" books_sold="25000" />
      <quarter number="4" books_sold="29000" />
    </region>
    <region>
      <name>East Coast</name>
      <quarter number="1" books_sold="27000" />
      <quarter number="2" books_sold="31400" />
      <quarter number="3" books_sold="40100" />
      <quarter number="4" books_sold="30000" />
    </region>
  </data>
</sales>
```
Adding an XSL statement

```xml
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="Transform.xsl"?>
  <sales>
    <summary>
      <heading>Greenwich Village Book Publishing</heading>
      <subhead>Regional Sales Report</subhead>
      <description>Sales report for the West Coast, Central and East Coast regions.</description>
    </summary>
    <data>
      <region>
        ...
        //rest of file would be here
      </region>
    </data>
  </sales>
```
Transform.xsl

<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="html"/>
    <xsl:template match="/">
        <HTML>
            <HEAD>
                <TITLE><xsl:value-of select="//summary/heading"/></TITLE>
            </HEAD>
            <BODY>
                <h1><xsl:value-of select="//summary/heading"/></h1>
                <h2><xsl:value-of select="//summary/subhead"/></h2>
                <p><xsl:value-of select="//summary/description"/></p>
            </BODY>
        </HTML>
    </xsl:template>
</xsl:stylesheet>
Greenwich Village Book Publishing

Regional Sales Report

Sales report for the West Coast, Central and East Coast regions.
Greenwich Village Book Publishing

Regional Sales Report

Sales report for the West Coast, Central and East Coast regions.

<table>
<thead>
<tr>
<th>Region/Quarter</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>West Coast</td>
<td>24,000</td>
<td>38,600</td>
<td>44,030</td>
<td>21,000</td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>11,000</td>
<td>16,080</td>
<td>25,000</td>
<td>29,000</td>
<td></td>
</tr>
<tr>
<td>East Coast</td>
<td>27,000</td>
<td>31,400</td>
<td>40,100</td>
<td>30,000</td>
<td></td>
</tr>
</tbody>
</table>
# Regional Sales Report

Sales report for the West Coast, Central and East Coast regions.

<table>
<thead>
<tr>
<th>Region/Quarter</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>West Coast</td>
<td>24,000</td>
<td>38,600</td>
<td>44,030</td>
<td>21,000</td>
<td><strong>127,630</strong></td>
</tr>
<tr>
<td>Central</td>
<td>11,000</td>
<td>16,080</td>
<td>25,000</td>
<td>29,000</td>
<td><strong>81,080</strong></td>
</tr>
<tr>
<td>East Coast</td>
<td>27,000</td>
<td>31,400</td>
<td>40,100</td>
<td>30,000</td>
<td><strong>128,500</strong></td>
</tr>
</tbody>
</table>
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>
<xsl:template match="/">

<HTML>
<HEAD>
<TITLE><xsl:value-of select="//summary/heading"/></TITLE>
</HEAD>
<BODY>
<h1><xsl:value-of select="//summary/heading"/></h1>
<h2><xsl:value-of select="//summary/subhead"/></h2>
<p><xsl:value-of select="//summary/description"/></p>

<!--Part 2-->
<table>
<tr>
<th>Region\Quarter</th>
<xsl:for-each select="//data/region[1]/quarter">
<th>Q<xsl:value-of select="@number"/></th>
</xsl:for-each>
<th>Total</th>
</tr>
<xsl:for-each select="//data/region">
<tr>
<th style="text-align:left"><xsl:value-of select="name"/></th>
<xsl:for-each select="quarter">
<td style="text-align:right">
<xsl:value-of select="format-number(@books_sold,'###.###')"/>
</td>
</xsl:for-each>
</tr>
</xsl:for-each>
</table>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
XPath

• “Xpath - is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. In addition to its use for addressing, XPath is also designed so that it has a natural subset that can be used for matching (testing whether or not a node matches a pattern); this use of XPath is described in XSLT.

XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string-value for each type of node. “

- W3C
XPath Examples

- ./author
  author
  Find all author elements within the current context (both are equivalent).

- first.name
  first.name
  Find all first.name elements.

- /bookstore
  Find the bookstore element at the root of this document.

- */
  Find the root element of this document.

- //author
  Find all author elements anywhere within the current document.

- /bookstore[@specialty = "textbooks"]
  Find all bookstores where the value of the specialty attribute is equal to "textbooks".

- book[@style = /bookstore/@specialty]
  Find all books where the value of the style attribute on the book is equal to the value of the specialty attribute of the bookstore element at the root of the document.

- book[@style = /bookstore/@specialty]
  Find all books where the value of the style attribute on the book is equal to the value of the specialty attribute of the bookstore element at the root of the document.
XML to HTML via an XSL
Sample XSL style sheet

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" omit-xml-declaration="no" indent="no" media-type="text/html"/>
  <!--XSL Stylesheet for generating Datasheet-->
  <xsl:template match="datasheet">
    <html>
      <head>
        <title>
          <xsl:value-of select="manufacturer"/>
          <xsl:value-of select="productline"/>
          Product Line - Data Sheet
        </title>
      </head>
      <body>
        <p align="right">
          <img src="Altova_right_300.gif" alt="Altova logo"/>
        </p>
        <h1>
          <xsl:value-of select="manufacturer"/>
          <xsl:value-of select="productline"/>
          Product Line - Data Sheet
        </h1>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

- - - rest of XSL would be here - - -
XSLT

• XML Stylesheet Language Transformation
• Subset of XSL.
• Designed for transforming one XML doc to a different format.
• Specialty language targeted at transformations used to format documents for presentation including: HTML, WML, PDF, etc.
• An XSLT file is an XML file.
• Patterns match source tree (source XML file) are transformed and appended to the output tree (target XML file).
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/
The xform is MVC it has a model section, a View (Data via a CSS / XSL) and Controller (action)

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

```xml
<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>
```

Copyright Poelman & Associates, Inc. 2002 - All Rights Reserved
XForms – action (controller) fragment

<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/Middle">

    <div>
        <xforms:label class="label">Middle Initial</xforms:label>
    </div>

    <xforms:hint>Enter Middle Initial</xforms:hint>

</xforms:input>

<xforms:input class="Last" id="last"
              ref="OrderInfo/PersonalInfo/Name/Last">

    <div>
        <xforms:label class="label">Last Name</xforms:label>
    </div>

    <xforms:hint>Enter Last Name</xforms:hint>

</xforms:input>

</div>

</xforms:group>

</body>
“The XML Query Language (XQL) is a notation for addressing and filtering the elements and text of XML documents. XQL is a natural extension to the XSL pattern syntax. It provides a concise, understandable notation for pointing to specific elements and for searching for nodes with particular characteristics.

The XSL pattern language provides an extremely understandable way to describe a class of nodes to process. It is declarative rather than procedural. One simply describes the types of nodes to look for using a simple pattern modeled after directory notation. For example, book/author means find author elements contained in book elements.

XQL provides a natural extension to the XSL pattern language. It builds upon the capabilities XSL provides for identifying classes of nodes, by adding Boolean logic, filters, indexing into collections of nodes, and more.

XQL is designed specifically for XML documents. It is a general purpose query language, providing a single syntax that can be used for queries, addressing, and patterns. XQL is concise, simple, and powerful.”

- W3C
• Questions
• Comments

• 7.75
Managing `<XML>` Persistence
Relational Theory

Tables can be joined through the key (ID) to create a one to many relationship

Customer Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0012</td>
<td>Poelman</td>
<td>43 Mercer</td>
</tr>
<tr>
<td>0013</td>
<td>Smith</td>
<td>2000 Main</td>
</tr>
<tr>
<td>0324</td>
<td>Chang</td>
<td>55 Broadway</td>
</tr>
</tbody>
</table>

Invoice Table

<table>
<thead>
<tr>
<th>ID</th>
<th>InvoiceNum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0012</td>
<td>12324</td>
<td>344.55</td>
</tr>
<tr>
<td>0012</td>
<td>44424</td>
<td>120.00</td>
</tr>
<tr>
<td>0324</td>
<td>55554</td>
<td>2344.00</td>
</tr>
<tr>
<td>0013</td>
<td>43533</td>
<td>0.99</td>
</tr>
</tbody>
</table>
A Join between both tables where ID = 0012 yields the table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>InvoiceNum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0012</td>
<td>Poelman</td>
<td>43 Mercer</td>
<td>12324</td>
<td>344.55</td>
</tr>
<tr>
<td>0012</td>
<td>Poelman</td>
<td>43 Mercer</td>
<td>44424</td>
<td>120.00</td>
</tr>
</tbody>
</table>
As XML it could be represented as:

```xml
<CustomerInvoices>
  <Customer>
    <ID>0012</ID>
    <Name>Poelman</Name>
    <Address>43 Mercer</Address>
    <Invoice>
      <Number>12324</Number>
      <Total>344.55</Total>
    </Invoice>
    <Invoice>
      <Number>44424</Number>
      <Total>120.00</Total>
    </Invoice>
  </Customer>
  <Customer>
    <ID>0013</ID>
    <Name>Smith</Name>
    <Address>2000 Main</Address>
    <Invoice>
      <Number>43533</Number>
      <Total>0.99</Total>
    </Invoice>
  </Customer>
</CustomerInvoices>
```
XML DataBase Management System (XML DBMS / XDBMS)

- XML is NOT relational its hierarchical!
- Storage of XML is inefficient in:
  - Files
  - RDBMS
  - Object DB (maybe but XML is just data no methods)
- XML database
  - XML data is parsed and accessible through a DOM-like API and SQL-type queries
  - High-performance, multi-user processing
  - Stored in a form that is more efficient for hierarchical data
- XML front end permitting access to heterogeneous data, relational data in particular
- Supports XQL
XML DBMS

- 4Suite, 4Suite Server
- Birdstep RDM XML
- Centor Interaction Server
- Cerisent XQE
- Coherity XML Database
- DBDOM
- dbXML
- DOM-Safe
- eXist
- eXtc
- eXtensible Information Server (XIS)
- GoXML DB
- Infonyte DB
- Ipedo XML Database
- Lore
- Lucid XML Data Manager
- MindSuite XDB
- Natix
- Neocore XML Management System
- Ozone
- Socrates XML
- Tamino
- Tendara Mobile XML Database
- TeraText DBS
- TEXTML Server
- Virtuoso
- XDBM
- XDB
- Xfinity Server
- X-Hive/DB
- Xindice
- Xyleme Zone Server
- XYZFind Server

Good overview http://www.rpbourret.com/xml/XMLAndDatabases.htm
Using **<XML>** in the enterprise
Integration of company data

- XML database as a front end:
  - Data durability and accessibility
  - Architecture independence
XML for application integration

• Electronic business must confront the growing number of information systems products (ERP, Business Directories, SGBD …)

• EAI (Enterprise Application Integration) offers a unified vision of heterogeneous data sources

• Modeling business processes in a standard way to mask the complexity of technical implementations
  – As a workflow graph
    • Works by sending asynchronous (or synchronous) messages

• Workflow interacts with products through event-linked connectors (new order, etc.)

The products must provide some EAI connectors!
XML for application integration (EAI)
• Questions
• Comments

• 8.0
Microsoft .net & C#

A better Java than Java!
"Quite simply, .NET is Microsoft's platform for XML Web services. XML Web services allow applications to communicate and share data over the Internet, regardless of operating system or programming language.

The Microsoft® .NET Platform includes a comprehensive family of products, built on XML and Internet industry standards, that provide for each aspect of developing, managing, using, and experiencing XML Web services. XML Web services will become part of the Microsoft applications, tools, and servers you already use today and will be built into new products to meet all of your business needs. “

- Microsoft website September 25, 2001
.NET Distilled - What

• OO layer that runs on top of the OS
  – Windows 9x, 2000, XP, PocketPC, Linux?
  – Simplify distributed object apps
  – Gives a path to migrate
  – CLR is a Virtual Machine (VM) + Libraries
  – Managed code model

• Language independent

• Better, Web-friendly ASP.NET architecture
  – XCOPY deployment
  – Lazy instantiation

• New security

• New UI model
.NET Framework

Common Language Specification (CLS)

- ADO.NET: Data and XML
- ASP.NET: Web Services & Web Forms
- Windows Forms

Common Type System (CTS)

Common Language Runtime (CLR)

Operating Systems
(9x, ME, 2000, XP, PocketPC, Linux?)
.NET Distilled - How

- Framework and tools
- Distributed application platform based on mostly industry standards
- A set of languages
- A common inheritance method across languages (CTS)
- Shared classes (CLS)
- Common exception model across languages
- Server products
- VisualStudio.NET tool
- Wizards/classes for the average developer
.NET Distilled - Why

• Technology leadership mindshare
• A better mousetrap
• Interoperability
• Potential instability of poorly written COM components
• Windows wasn’t OO
• Java – language, platform, ideas
• DNA / ASP requirement multiple technologies
• Piracy of Microsoft software
Microsoft and 3\textsuperscript{rd} party support for .NET development includes (announced):

- APL
- C++
- C#
- COBOL
- Component Pascal
- Curriculum
- Eiffel
- Fortran
- Haskell
- Java Language
- Microsoft JScript
- Mercury
- Mondrian
- Oberon
- Oz
- Pascal
- Perl
- Python
- RPG
- Scheme
- Smalltalk
- Standard ML
- Microsoft Visual Basic
Vision

• Platform Independence
• Common Libraries Across Languages
• Embracing Open Standards
• No Need For Java
• Software As Services
• Enterprise Software Infrastructure
• .NET Servers - Off The Shelf Solutions
• Lower Time To Market
• Lower TCO
Benefits

• Enabling technology – what CORBA / COM+ / RMI promised may now be delivered

• Faster time to market

• Lower development costs

• Buy vs. Build

• Lower footprint so lower TCO

• Integration between packaged software (MS OfficeXP) and custom apps
Benefits

• OO
• Distributed
• Enterprise scalable
• Better security
• Monitorable
• More “plumbing” is provided by the OS
• Simpler integration - A2A, EAI, B2B
• Lower skills needed by developers
C#.NET Upside

- **Objected-Oriented** – Encapsulation, Polymorphism, Inheritance
- **Versioning Support** – Eliminate DLL hell to prevent code incompatibility when newer code are made.
- **Type Safety** – All variable casts are required to be safe. Can’t create an invalid cast.
- **Rich API** - Pre-built libraries allows programmer to be more productive.
- **Leverage the CLR environment** – Automatic Garbage Collection, Security, Language Interoperability, and Legacy Integration.
C# .NET Downside

• **Lack of Portability** - Only supported on the Window platform. It is currently not portable across other operating systems.

• **New syntax and API** - Applying the new class libraries and constructs effectively may require a slight learning curve.

• **YALL** - Yet another language to learn and support.

• **Unproven** - no track record or its ease of use, power, productivity,
C# and Java Key Similarities

• Code compiled to interpreted language (IL/byte-code).
• Automatic garbage collection of objects.
• All values are initialized before use.
• Interfaces, with multiple inheritance of interfaces, single inheritance of implementation.
• All code scoped to packages (or assemblies in .NET)
Creating a WebForms app in .net
Design “canvas”

Click and it generates a handler in C#
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace WebApplication1
{
    /// <summary>
    /// Summary description for WebForm1.
    /// </summary>
    public class WebForm1 : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Label Label1;
        protected System.Web.UI.WebControls.TextBox txtYourName;
        protected System.Web.UI.WebControls.TextBox txtTargetArea;

        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            // CODEGEN: This call is required by the ASP.NET Web Form Designer.
            InitializeComponent();
            base.OnInit(e);
        }

        private void btnMoveText_Click(object sender, System.EventArgs e)
        {
        }
    }
}
Add one line of code

- `txtTargetArea.Text = txtYourName.Text;`
- And you have a web application!
.NET Recap

- Great New Development Languages and IDE
- Extremely Powerful VM and Container Model (CLR)
- Language Interoperability A Big Plus
- XML everywhere!
- Fast Dev for Small - Medium Web/Desktop Apps
- Not Enterprise Scale Yet but will be soon
- Better Security
- Mixed COM+/.NET highly likely on next 12-24 months
- TCO could be higher over next 24 months but lower in a future CLR / OS without COM+
## .NET vs. J2EE

<table>
<thead>
<tr>
<th></th>
<th>.NET</th>
<th>J2EE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OO Languages</strong></td>
<td>C#, VB, …</td>
<td>Java</td>
</tr>
<tr>
<td><strong>Virtual Machine</strong></td>
<td>CLR</td>
<td>JVM</td>
</tr>
<tr>
<td><strong>Container</strong></td>
<td>ServicedComponents in the COM+</td>
<td>EJB</td>
</tr>
<tr>
<td><strong>Pseudo ops</strong></td>
<td>IL</td>
<td>JavaByteCode</td>
</tr>
<tr>
<td><strong>Platforms</strong></td>
<td>Windows</td>
<td>Windows, Unix, OSS/390</td>
</tr>
<tr>
<td></td>
<td>Mono on Linux</td>
<td></td>
</tr>
<tr>
<td><strong>XML</strong></td>
<td>Everywhere</td>
<td>Lots</td>
</tr>
<tr>
<td><strong>WebServices</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>UI</strong></td>
<td>WinForms, WebForms</td>
<td>Swing, JSP</td>
</tr>
<tr>
<td><strong>Maturity</strong></td>
<td>Beta but reused stuff and good quality now</td>
<td>3-4 years</td>
</tr>
</tbody>
</table>
• Questions
• Comments

• 8.20
Putting it All together

What tools do you use to solve what problems when programming for the www?
<<Note to students>>

• You should try and answer these questions on the next slides your self.
• I will present my answers in class.
• You should try ahead of time to see if you choose the same answers and why. If you disagree, why?
Think you are building Amazon.com

- Existing Inventory Management System (IMS)
  - built in COBOL on IBM mainframe
  - Tracks what books are in inventory, reserved, and back ordered
  - IBM MQ Enabled
- Existing customer relationship management system (CRMS)
  - Java EJB Based
  - Keeps track of customer info NOT orders, though.
  - Purchased off the shelf – not built by your team
- You need to build a New order management system (OMS)
  - Keeps track of orders for customers of items in the inventory
  - Will need to communicate with Users via UI, with CRMS and IMS
- Users will access via the Internet
- Administrators of the Amazon bookstore need a rich user Interface to customer, inventory and order mgmnt systems.
What should this Arch look like?

Users

Admins

What goes here?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?

Protocol?
What About Security?

Users

Browser

Internet

Outside

DMZ

Admins

Inside

CRMS

OMS

IMS
Now, AMAZON decides to sell Starbucks Coffee Stuff through its web Site

• SB has a its OWN ORS&IMS
• Its Built in C# & MS .net!
• It is hosted somewhere else.
• Amazon wants to Front-end SB’s back end system.
• What do you do?
Add in SB back-ends

Users
Browser

Internet
Outside

DMZ

Biz Logic in C#

DMZ

IMS
OMS
CRMS

Inside

Amazon

Starbucks

IMs
OMS
What tech do you use

• Users UI? Why?
• Biz Logic? Why?
• Connection to the Mainframe IMS? Why?
• Connection to the CRMS? Why?
• Built the OMS? Why?
• Connect to the OMS? Why?
• Admin console? Why?
• Starbuck’s backend? Why?
Security

• Between User UI and Amazon? Why?
• Between DMZ and Amazon’s
  – IMS?
  – OMS?
  – CRMS?
• Between Amazon and Starbucks? Why?
• Questions
• Comments

• 8.75
Review of WWW Technologies
Major WWW Technologies

- IP
- UDP
- TCP
- FTP
- SMTP / POP / IMAP
- HTML
- HTTP / HTTPS
- XML
- URL
- DNS
- MIME
- CGI
- LDAP
- SOAP
Major WWW Concepts

- Packet vs. Circuit Based
- Connectionless vs. Connection-based Protocol
- Stateless vs. Stateful Protocol
- IP Address vs. URL
- Localhost
- Protocol, Address, Port
- Cookies, URL Rewrites
- Browser vs. Server
- Client / Server vs. Peer to Peer
- Maintainability
- Scalability
- Fault Tolerance
- Integration
- Time to market

- OO
  - Object = Data + Behavior
  - Maintain State
  - Inheritance
  - Encapsulation
  - Overriding
  - Polymorphism
  - Classes & Objects
  - Messages
  - Object ID
  - Genericity
- Class vs. Instance
  - Methods
  - Attributes
- Overloading vs. overriding
- Tiers
  - Presentation
  - Biz Logic
  - Data
- MVC
WWW Servers

- Socket
- HTTP/Web Server
- Servlet Engine
- Application Server
- Mail Server
- DNS
- Firewall
- WebService Server
WWW and Java

- OO
- Platform Independent
- C++ Syntax
- Interpreted
- JVM
- Bytecodes & Verification
- Single Inheritance
- Garbage Collection
- References
- Casting
- Shadowing
- Serialization

- Super
- This
- Classes vs. Interfaces
- Multithreading
- Heap vs. Stack
- Primitives vs. Objects
- Call By Value vs. Call By Reference
- Constructors vs. Methods
- Exceptions
- J2SE vs. J2EE vs. J2ME
- Javac.exe, java.exe, rmic.exe, javadoc.exe
WWW and Java

- Operators
- Primatives
- Class Object
- Class Class
- Equality vs. Identity
- String vs. StringBuffer
- Arrays
- Collections
- Wrapper Classes
- Private, Protected, Public and package

- Classes vs. Interfaces
- Multithreading
- Heap vs. Stack
- Primitives vs. Objects
- Call By Value vs. Call By Reference
- Constructors vs. Methods
- Exceptions
- Reflection
- J2SE vs. J2EE vs. J2ME
- Javac.exe, java.exe, rmic.exe, javadoc.exe
Major WWW Java

- Swing
- I/O
- Servlet
  - Class vs. Instance vs. Method Variables
  - Lifecycle
- JSP
  - Directives
  - Scriptlets
  - Actions
- JavaBean
  - Getters/Setters
  - Events
- RMI
  - Data vs. Behavior
- EJB
  - Session Bean
    - Stateful
    - Stateless
    - Entity
    - MDB
- JMS
WWW and Design Patterns

- Stateless
- Connectionless
- Session
- Singleton
- Class Factory
- Value Object
- Command Pattern
  - HTTP Get, Post, Delete, Put …
- Callback
- MVC

- Object Lifetimes/Scopes
  - Request
  - Page
  - Session
  - Application
- Transaction
- Container Managed
- Service Oriented Architecture (SOA)
- Synchronous vs. Asynchronous
Major WWW Security Choices

- Encryption
  - Private Key
  - Public Key
- What is / How do you do it?:
  - Identification
  - Authentication
  - Authorization
  - Integrity
  - Non-repudiation
  - Digital Signatures
- SSL
- Firewalls
  - Address Filtering
  - Protocol Filtering
  - Proxy Based
  - Address Translation

- WebServices
  - HTTP
  - XML
  - SOAP
  - WSDL
  - UDDI
• Questions
• Comments

• 9.00
The end!

<<this is only the tip of the iceberg for the Web, the Internet and Java>>
Find out more...

• Interesting sites:
  – http://w3c.org
  – http://www.w3.org/TR/
  – http://www.oasis-open.org
  – http://www.schema.net
XML web links

• http://www.w3.org/TR/SOAP/
• http://www.xml.org
• http://www.UDDI.org
• http://xml.com/xml/pub
• http://java.sun.com/xml
• http://www.xmlsoftware.com
• http://www.xml.org/xml/resources_cover.shtml
• http://www-106.ibm.com/developerworks/xml/
• http://www.microsoft.com/net/
• Interesting tutorials http://www.w3schools.com
• http://www.gotdotnet.com/