Internet Programming

XML, JSON, SOAP & Web Services

Dan Brandon, Ph.D., PMP

Extensible Markup Language

- The beginning of XML goes back to 1997 where a group of developers at SUN (Jon Bosak, et al) recognized that the Internet needed more “structure” than was available with just HTML
- A problem with HTML is that it mixes together both the content and presentation of data
- Even with HTML 5 and CSS there is still a mixture of content and presentation even though the “style” of the presentation can be separated

Separation of Content

- XML separates data content from data presentation (use of the data)
- XML is a subset of SGML (Standard Generalized Markup Language – ISO 8879) which dates back to 1986
- HTML is an application of SGML (predefined tag sets) – 1992
- However, XML can be extended and customized for new and special element types, since the tags are not predefined

XML Usages

- Describing “metacontent” (information about a document’s contents) for document management, searching, etc.
- Publishing documents
- Exchange of data and documents, including contents of database systems
- As a “messaging” format between applications and systems (“SOAP”)

E-Commerce & EAI

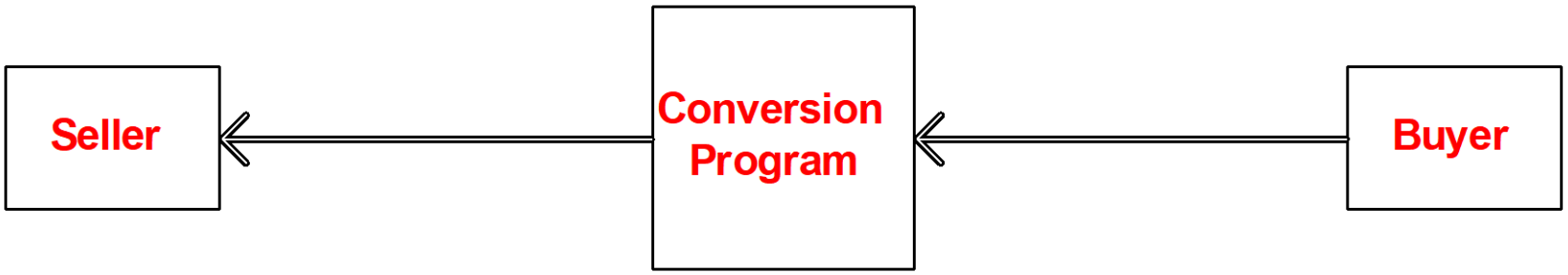
- About the same time as XML was being developed and the Internet was exploding, two other industries were forming:
  - EAI (Enterprise Application Integration) or “ERP” which had developed to help corporations link together all their data from different applications and departments
  - E-Commerce which had developed to help corporations use the Internet for commerce with their customers and suppliers
- Both of these two new industries needed a way to transfer data without having to develop “point-to-point” systems, but with a common format that was web suited and also vendor and platform independent; XML filled that need

Point to Point

[Two companies agree on a program to transform data semantics and format from one system to another]

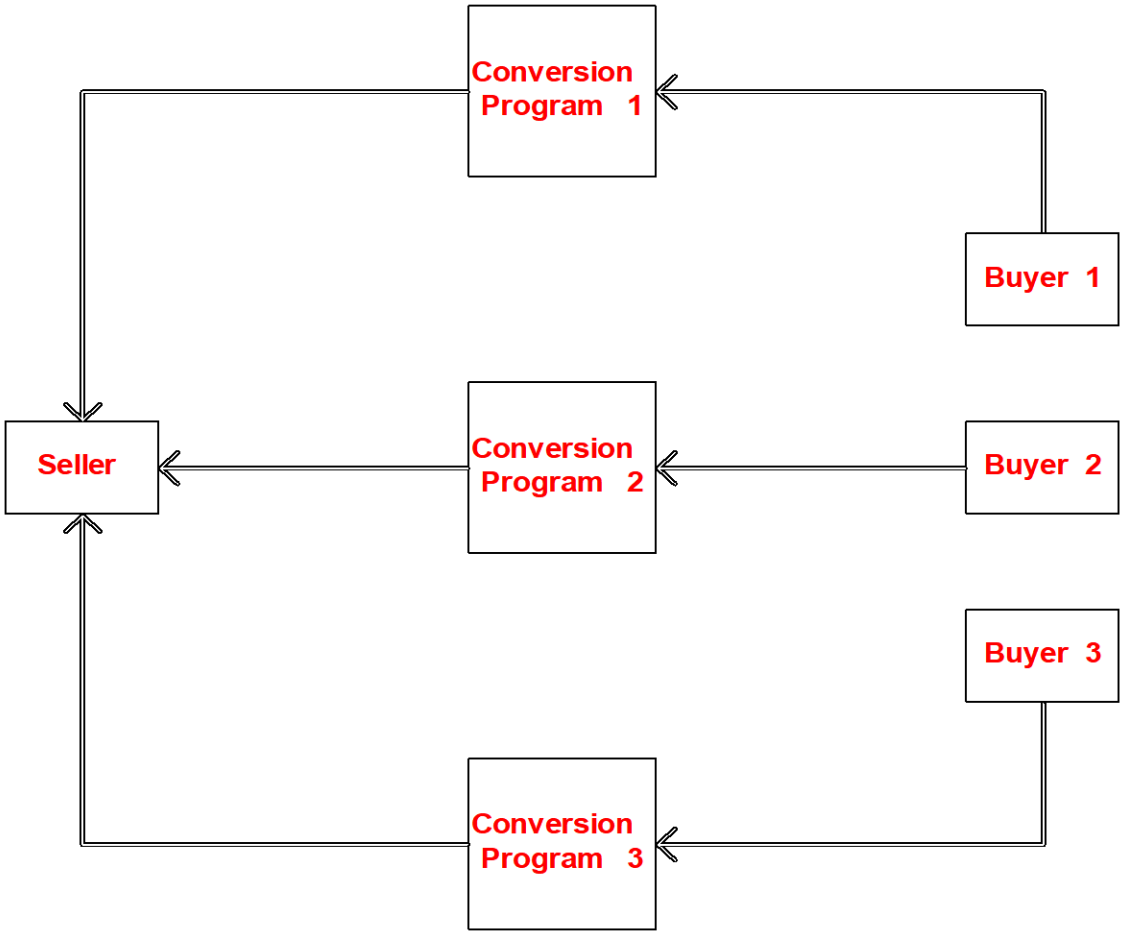
- Seller's System
  - Address 50 a/n
  - City 20 a/n
  - State – 2 a/n
  - Zip - 5 n, 4 a/n
  - Product Code – 3 a, 4 n
  - Price – 9n
- Buyer's System
  - Address line 1 – 30 a/n
  - Address line 2 – 30 a/n
  - City – 18 a/n
  - State – 2 a/n
  - Zip – 5n
  - Product Code – 7 a/n
  - Cost – 6/2 d

Two companies agree on a program to transform data semantics and format from one system to another:

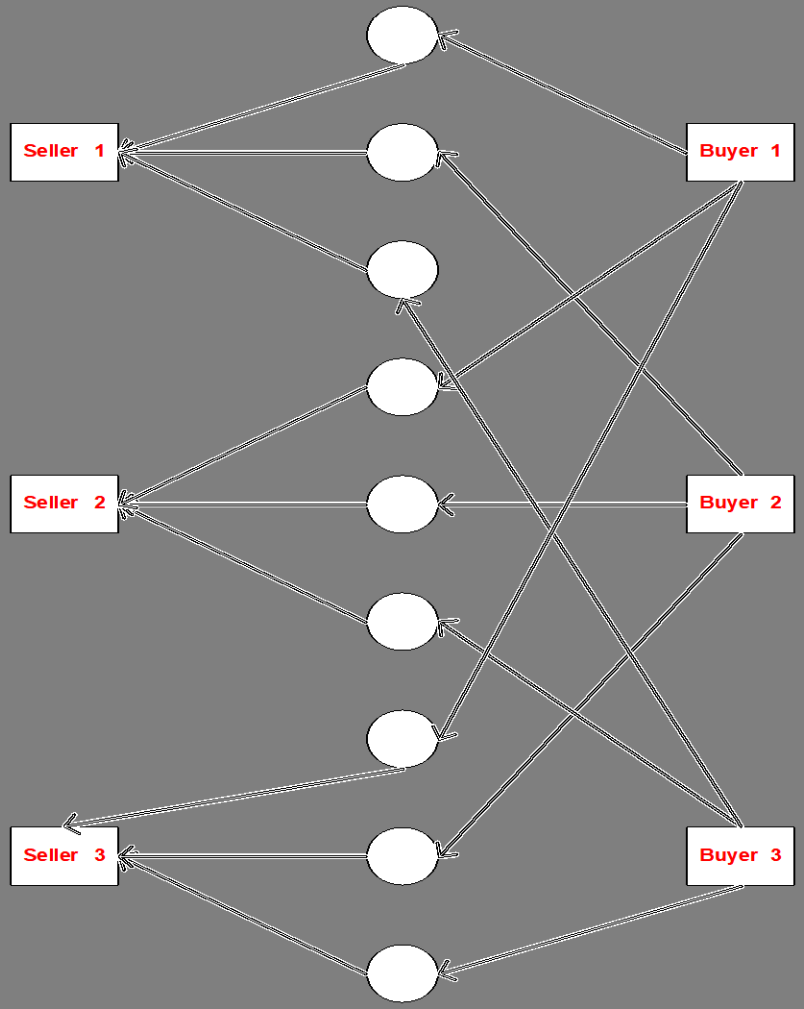


Seller typically writes and maintains conversion software.

Now instead of two parties, many more buyers trade electronically; this causes the seller to make a different conversion program for each buyer:



Now each buyer wants to get prices and trade with many sellers; many middleware conversion programs must be implemented:



EDI/XML (con't)

- Add to this the fact that all the companies have different computer types, operating systems, communication protocol (SNA, DECNET, IPX, TCP/IP, …), and physical data communication networks
- EDI (Electronic Data Interchange) standards had been developed since 1970 (ANSI X12 in US and EDIFACT in Europe)
- However no two EDI standards used the same approach, none were web enabled, none were totally platform independent, and many are very expensive and require special VAN (value added networks)

Format Comparison

- Fixed Length (file/record):
  - Doe John
- EDI (defined field and record separators):
  - NM1*1B*1*Doe*John
- XML:
  - <NAME>
    - <LAST>Doe</LAST>
    - <FIRST>John</FIRST>
  - </NAME>

XML is “plain text”, other formats are typically binary

EDI Cost

- Traditional EDI
  - $40,000 to $120,000 for software
  - 25 cents to $1 per transaction
- Internet Based EDI (with or without XML)
  - $300 to $50,000 for software
  - No transmission charge

EDI Market Shares

- Traditional EDI – about 30%
- Internet Based EDI – about 20%
- XML Based – 30% (fastest growing)
- Proprietary – 5%
- Other (old fashion file transfer)– 15%

XML (and JSON) will be the Heart of the B2B Internet

- XML will eventually replace today's EDI formats for defining the information that is electronically exchanged between companies (B2B E-Commerce)
- XML is faster to implement, has lower maintenance costs, and is more reliable than EDI formats and processing
- Internet B2B will mean significant cost reductions for business, for example estimates:
  - 14 % reduction in the cost to build a car in that industry
  - 5 to 40% reduction in other industries

XML Structure

- An XML document may have two parts:
  - A DTD (Document Type Definition) or XML Schema which defines the format (type and name of document items)
  - The body of the document, which contains the content and the “names” of the content items

XML Structure (con't)

- For an XML document to be "well-formed" it must meet the well-formedness constraints (WFC's) defined in the 1.0 Recommendation
- For an XML document to be "valid", it must meet the validity constraints (VC's) defined in the 1.0 Recommendation and expressed in a DTD or XML Schema
- DTD's and XML Schemas will be discussed later

XML Body Formatting
[well-formed]

- XML format is more strict than HTML:
  - Elements (except empty elements) must have start and end tags; for example in HTML you can have a <P> without a </P>; not so in XML
  - Empty elements (one without a closing tag) must end with / (ie <myTag/>)
  - All elements must be nested correctly; for example in HTML you can start <H2><P> and end with </H2></P>; in XML you must end with </P></H2>

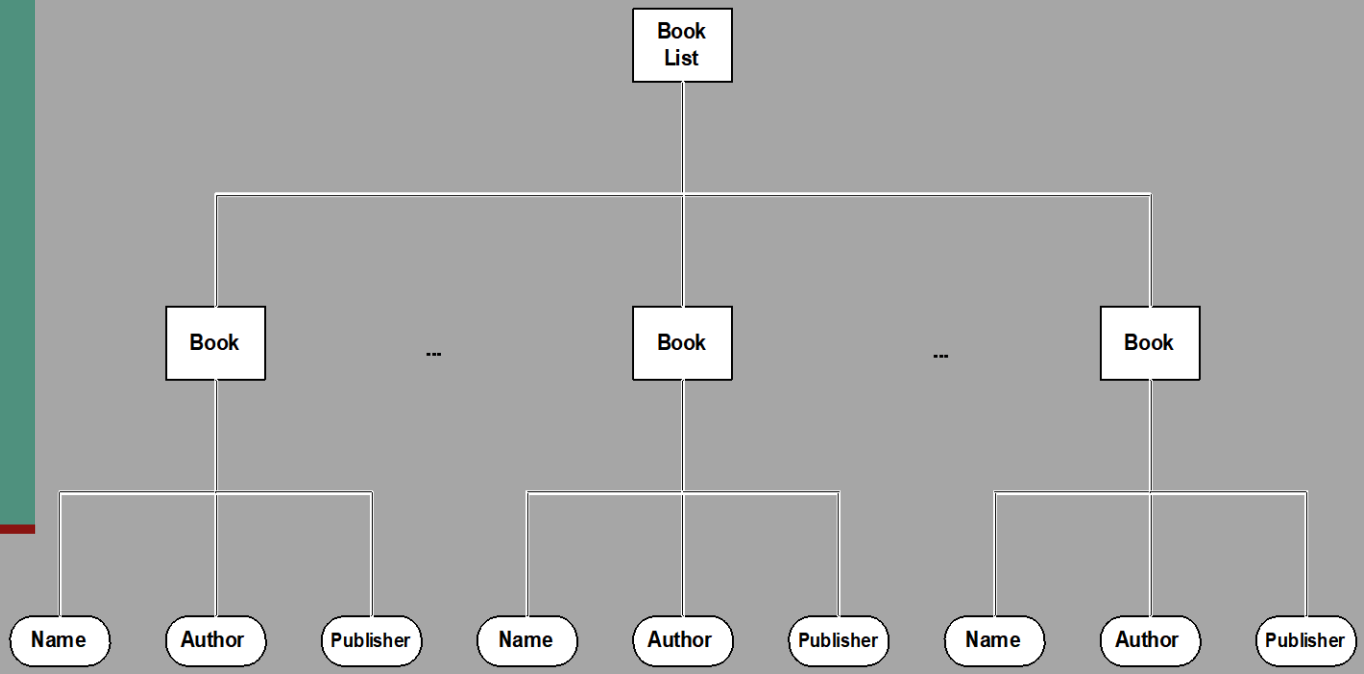
XML Body Formatting (con't)

- All attributes must be in quotation marks; in HTML only certain attributes must be in quotes such as strings and URL's.
- Attribute names must be unique within an element, and attribute values cannot contain the symbol '<'
- XML is case sensitive, and tags are lower case
- "Namespaces" may be defined using URI's (universal resource indicators) so that tag naming conflicts do not arise

XML Data Structures

- Many data structures can be set up in XML
- The most common is a “tree structure”
- Others include “graphs” and “tables”, but these are beyond the scope of this course

XML Hierarchy of Elements



"Tree Structure"

- XML Body's have a hierarchical arrangement of information
- For example, a list of books can be visualized as shown here
- BookList is the "root" element

XML Data File Format (.xml)

[Version is specified in the “prolog” , lines before the root element]

```
<?xml version="1.0"?>
<BookList>
    <Book>
        <Name>…</Name>
        <Author>…</Author>
        <Publisher>…</Publisher>
    </Book>
    <Book>
        <Name>…</Name>
        <Author>…</Author>
        <Publisher>…</Publisher>
    </Book>
    <Book>
        <Name>…</Name>
        <Author>…</Author>
        <Publisher>…</Publisher>
    </Book>
</BookList>
```

Sample XML File (books.xml)

```
<?xml version="1.0"?>

<BookList>
        <Book>
                <Name>Database Processing</Name>
                <Author>Kroenke</Author>
                <Publisher>Prentice Hall</Publisher>
        </Book>
        <Book>
                <Name>Modern Systems Analysis and Design</Name>
                <Author>Hoffer, George, Valacich</Author>
                <Publisher>Addison Wesley</Publisher>
        </Book>
        <Book>
                <Name>Data and Computer Communications</Name>
                <Author>Stallings</Author>
                <Publisher>Prentice Hall</Publisher>
        </Book>
        <Book>
                <Name>An Invitation to Computer Science</Name>
                <Author>Schneider and Gersting</Author>
                <Publisher>PWS</Publisher>
        </Book>
        <Book>
                <Name>Java, How to Program</Name>
                <Author>Deitel and Deitel</Author>
                <Publisher>Prentice Hall</Publisher>
        </Book>
</BookList>
```

Note ".xml" extension

```
<?xml version = "1.0"?>

<letter>

   <contact type = "from">
      <name>John Doe</name>
      <address1>123 Main St.</address1>
      <address2></address2>
      <city>Anytown</city>
      <state>Anystate</state>
      <zip>12345</zip>
      <phone>555-1234</phone>
      <flag id = "P"/>
   </contact>

   <contact type = "to">
      <name>Joe Schmoe</name>
      <address1>Box 12345</address1>
      <address2>15 Any Ave.</address2>
      <city>Othertown</city>
      <state>Otherstate</state>
      <zip>67890</zip>
      <phone>555-4321</phone>
      <flag id = "B"/>
   </contact>

   <paragraph>Dear Sir,</paragraph>

   <paragraph>It is our privilege to inform you about our new
      database managed with XML. This new system will allow
      you to reduce the load of your inventory list server by
      having the client machine perform the work of sorting
      and filtering the data.</paragraph>
   <paragraph>Sincerely, Mr. Doe</paragraph>

</letter>
```

XML File for A letter

Character Handling

- XML can also handle international character sets
- Unicode (ISO-10646) is the character set used
- A “character encoding” (mapping of the character set to particular bit patterns) can be specified in the XML prolog and default is UTF-8 (ASCII)

Language Handling

- For each tag holding character data, a language (locale) can be specified via a language code and optionally a country code (for “dialects”):
  - <paragraph xml:lang=“en-US”>…some text…</paragraph>
- You can also make up your own languages !

ISO Codes

- Common ISO 639 Language Codes:
  - ar Arabic
  - ch Chinese
  - de German
  - es Spanish
  - fr French
  - it Italian
  - ja Japanese
  - ru Russian
- Common ISO 3166 Country Codes:
  - CA Canada
  - CN China
  - DE Germany
  - EN England
  - ES Spain
  - FR France
  - IT Italy
  - JA Japan
  - RU Russia
  - US United States

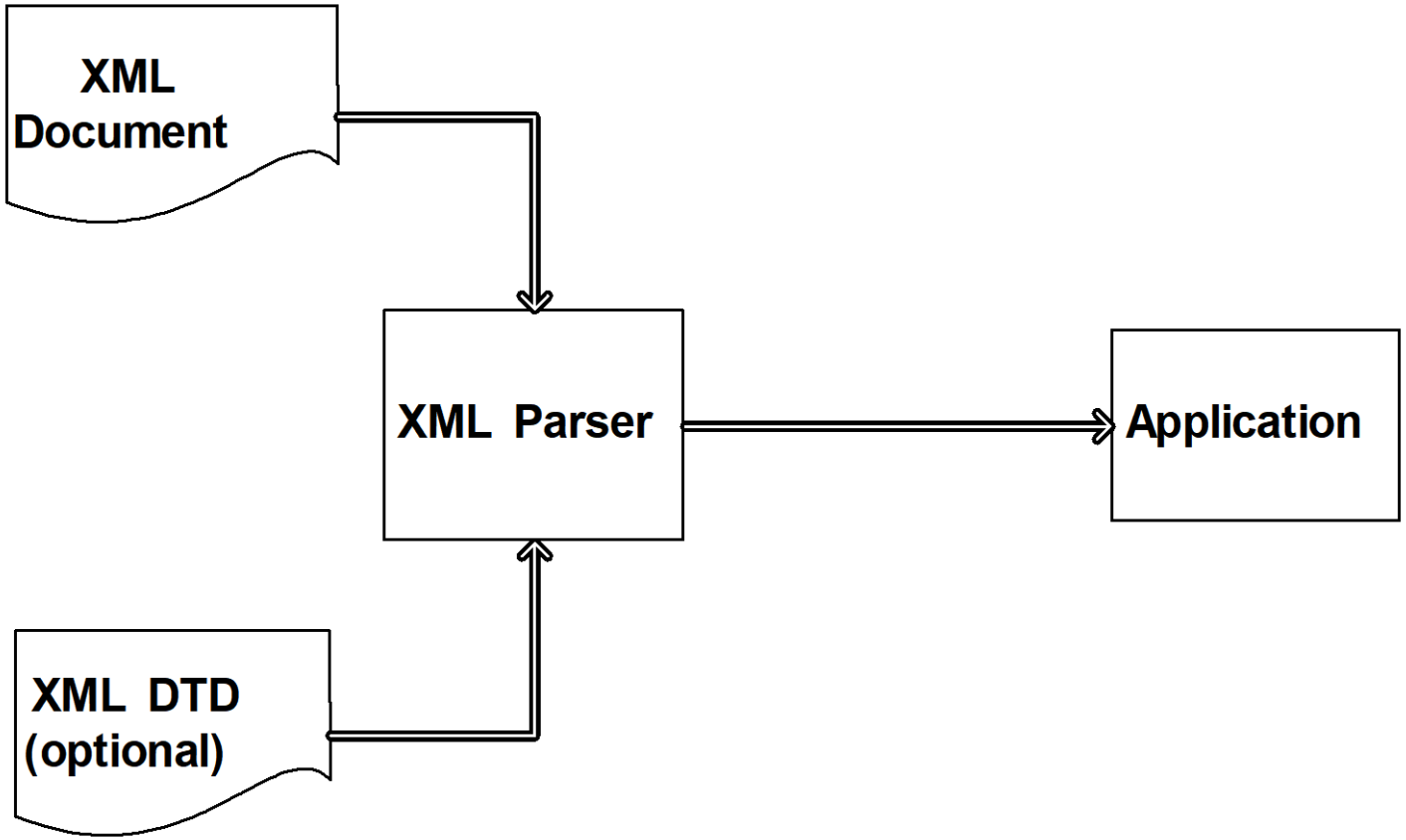
Producing XML Files

- XML files are just text files and can be manually created
- The most common way the files are produced is by a program (ie C++, PHP, or Java program) which inputs an SQL statement, retrieves the indicated data from the relational database, and outputs the data in the desired XML format
- Latest versions of major programs and databases have features to export data into XML format

Reading (“parsing’) an XML Document

- A software program called an XML Parser is used to read and process an XML document
- These programs are typically written in Java, C++, PHP, or Python
- A number of general purpose parsers are freely available: www.xml.com/xml/pub/Guide/XML_Parsers
- Parsers usually are of two types: DOM, or SAX
- Parsers that use a DTD (or XML Schema) are called “validating” parsers

XML Parsing



SAX Parsers

- "Simple API for XML"
- As the XML document is scanned, events are generated for the various starting and ending tags and text contained in tags
- A user's application writes event handler functions to take specific actions when these events occur (the event object returns the tag or text encountered)
- Modern SAX parsers are usually written in Java or PHP

DOM Parsers

- A DOM Parser reads the entire XML document into the computer's memory in the DOM format (tree structure)
- Browsers use the DOM approach, IE's built in parser is called "msxml"
- The DOM model defines properties and methods for the XML "document" and the "nodes" in the document tree
- These properties and methods can be used via JavaScript to manipulate an XML document (such as creating dynamic HTML)

Other Parsers

- Some applications use special “embedded” parsers
- These embedded parsers may be special purpose for a particular type of XML document (particular DTD)
- The embedded parses are usually faster and take up less memory that a general purpose parser

Processing (“parsing”) XML in a Web Page

- Built-In (in Browser)
- Java Applets
- ActiveX Controls
- Processing on Server

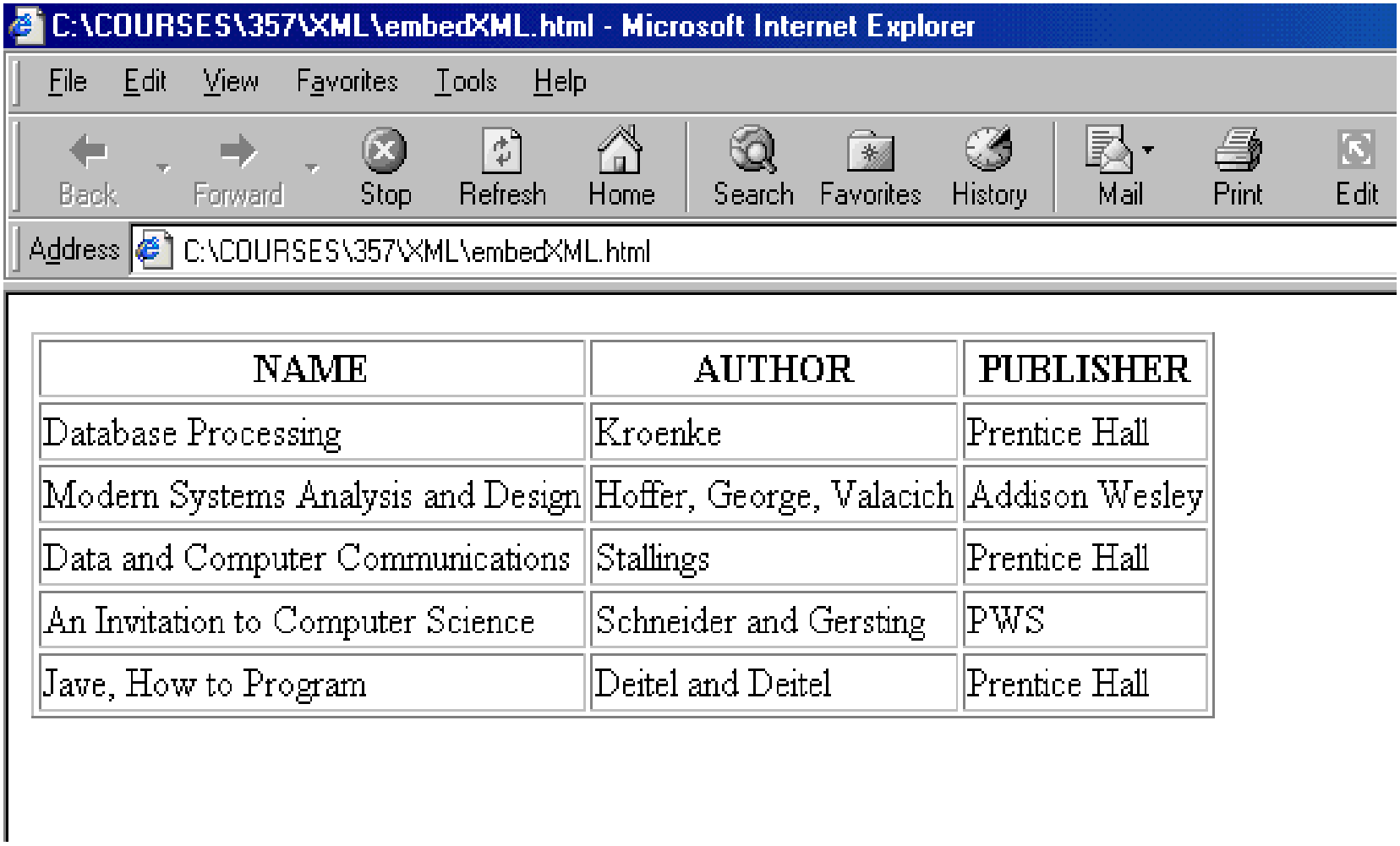
Embedded

- An XML document can be “embedded” in an HTML file so that the data can be “presented”
- An XML document is embedded in an HTML file via the XML tag, and the ID of the XML tag is the name of the DSO (data source object) created
- The embedded XML file is called a “data island”

```
<HTML><BODY>
<XML ID = "xmlDoc">
   <BookList>
        <Book>
                    <Name>Database Processing</Name>
                    <Author>Kroenke</Author>
                    <Publisher>Prentice Hall</Publisher>
        </Book>
        <Book>
                    <Name>Modern Systems Analysis and Design</Name>
                    <Author>Hoffer, George, Valacich</Author>
                    <Publisher>Addison Wesley</Publisher>
        </Book>
        <Book>
                    <Name>Data and Computer Communications</Name>
                    <Author>Stallings</Author>
                    <Publisher>Prentice Hall</Publisher>
        </Book>
        <Book>
                    <Name>An Invitation to Computer Science</Name>
                    <Author>Schneider and Gersting</Author>
                    <Publisher>PWS</Publisher>
        </Book>
        <Book>
                    <Name>Jave, How to Program</Name>
                    <Author>Deitel and Deitel</Author>
                    <Publisher>Prentice Hall</Publisher>
        </Book>
   </BookList>
</XML>
<TABLE BORDER = "1" DATASRC = "#xmlDoc">
   <THEAD>   <TR>
      <TH>NAME</TH>
      <TH>AUTHOR</TH>
      <TH>PUBLISHER</TH>
   </TR>   </THEAD>
   <TR>
      <TD><SPAN DATAFLD = "Name"></SPAN></TD>
      <TD><SPAN DATAFLD = "Author"></SPAN></TD>
      <TD><SPAN DATAFLD = "Publisher"></SPAN></TD>
   </TR>
</TABLE>
</BODY></HTML>
```

← Table Repeater

embedXML.html



NAME	AUTHOR	PUBLISHER
Database Processing	Kroenke	Prentice Hall
Modern Systems Analysis and Design	Hoffer, George, Valacich	Addison Wesley
Data and Computer Communications	Stallings	Prentice Hall
An Invitation to Computer Science	Schneider and Gersting	PWS
Jave, How to Program	Deitel and Deitel	Prentice Hall

XML data islands are no longer supported

Support for *XML data islands* has been removed in Internet Explorer 10 standards and quirks modes for improved interoperability and compliance with HTML5. This means that XML data islands are now parsed as HTML, just like in other browsers. This change can impact pages written exclusively for Windows Internet Explorer or pages that use browser sniffing to alter their behavior in Internet Explorer.

Note Most pages are unaffected by this change.

Impact

- Applies to Internet Explorer 10 and later.
- Affects IE10 Standards mode and later, including interoperable quirks mode.

A page using XML data islands worked as intended in Windows Internet Explorer 9, but no longer works in Internet Explorer 10.

If the page works correctly in other browsers, consider using feature detection to treat Internet Explorer 10 like other browsers. Otherwise, add the following meta tag near the top of the page to opt into Internet Explorer 9 behavior:

HTML

Copy

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9">
```

External XML File

- The XML tag can also refer to an external XML file:

```
<HTML><BODY>
<XML ID = "xmlDoc" SRC = "books.xml"></XML>
<TABLE BORDER = "1" DATASRC = "#xmlDoc">
   <THEAD><TR>
      <TH>NAME</TH>
      <TH>AUTHOR</TH>
      <TH>PUBLISHER</TH>
   </TR></THEAD>
   <TR>
      <TD><SPAN DATAFLD = "Name"></SPAN></TD>
      <TD><SPAN DATAFLD = "Author"></SPAN></TD>
      <TD><SPAN DATAFLD = "Publisher"></SPAN></TD>
   </TR>
</TABLE>
</BODY></HTML>
```

No longer directly supported in some browsers !

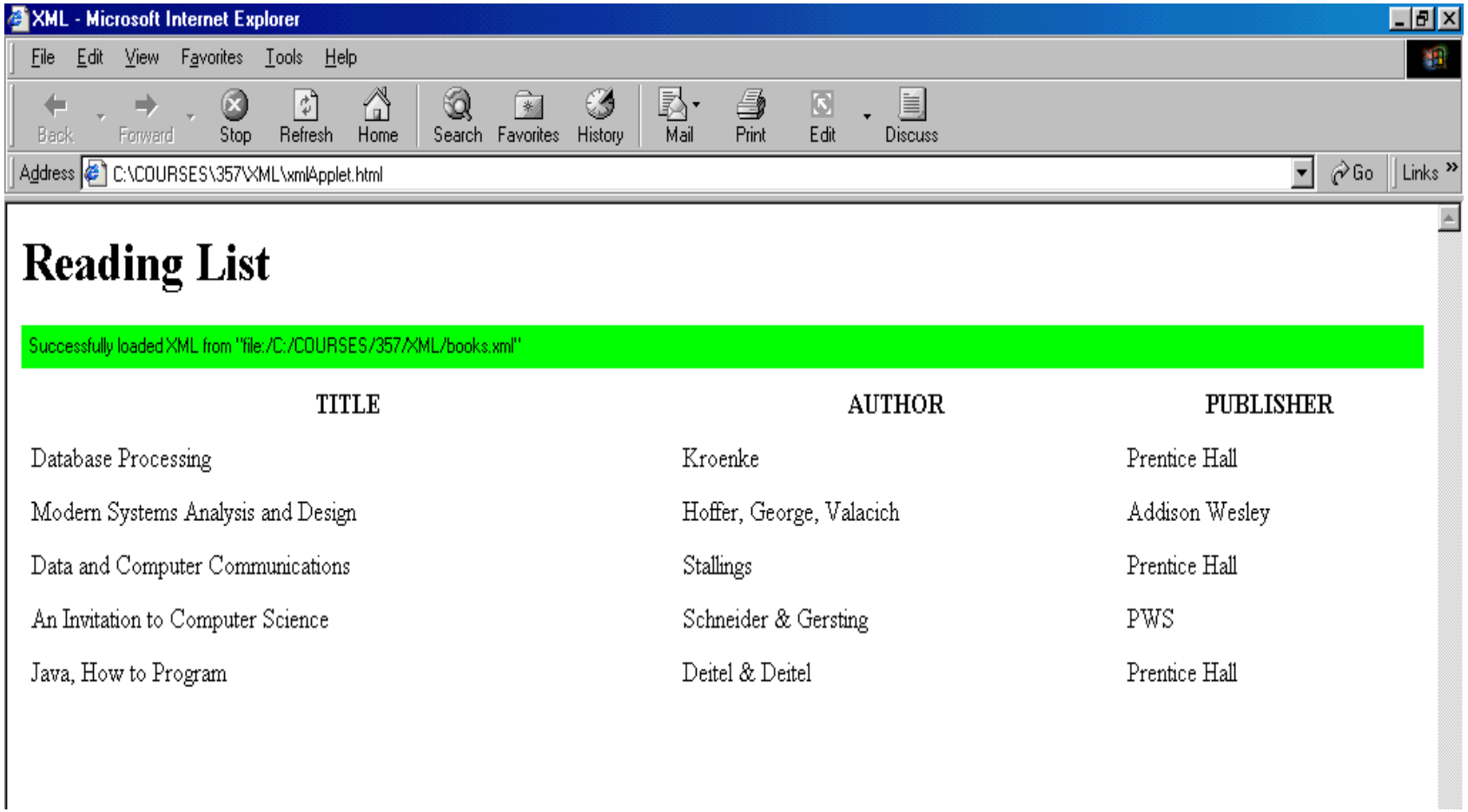
ExternalXML.html

Using an Applet

- One can write (or use an existing Java Applet) to read an XML file and to process it
- The processing and display can be done entirely within the Applet; that is, the Applet Java code can display the information on the web page (see later example of this)
- Alternatively, an Applet can turn the XML file into a DSO (data source object)
  - As a DSO the information can be processed as has been shown in previous material; the most common way to do it is in an HTML table (see next slide)

```
<HTML><HEAD><TITLE>XML</TITLE></HEAD>
<BODY>
<H1>Reading List</H1>
<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
        HEIGHT="25" ID="xmldso" MAYSCRIPT="true">
       <PARAM NAME="url" VALUE="books.xml">
</APPLET>
<TABLE ID="table" BORCER="2" WIDTH="100%"
     DATASRC="#xmldso" CELLPADDING="5">
     <THEAD><FONT FACE="Arial" SIZE="2">
             <TR>
                     <TH>TITLE</TH>
                     <TH>AUTHOR</TH>
                     <TH>PUBLISHER</TH>
             </TR>
     </FONT></THEAD>
     <FONT FACE="Arial" SIZE="2">
             <TR>
                     <TD VALIGN="top"><DIV DATAFLD="Name"
                         DATAFORMATS="HTML"></DIV></TD>
                     <TD VALIGN="top"><DIV DATAFLD="Author"
                         DATAFORMATS="HTML"></DIV></TD>
                     <TD VALIGN="top"><DIV DATAFLD="Publisher"
                         DATAFORMATS="HTML"></DIV></TD>
             </TR>
     </FONT>
</TABLE>
</BODY></HTML>
```

xmlApplet.html



XML Parsing via Applets

- Use a “canned” applet
  - Like: `com.ms.xml.dso.XMLDSO.class`
- Write your own applet and use an available XML parser class (typically covered in a Java Programming class)
  - There are two basic types:
    - SAX Parsers – event oriented – “call back”
    - DOM Parsers – turns XML document into DOM representation in memory
  - www.apache.org has examples of both of these
  - Also Sun [JAXP – java.sun.com/xml/download.html] and IBM [www.alphaworks.ibm.com/formula/xml] have free versions):
- Write your own Java parser (for example to minimize memory and download time)

Sample XML Quiz File

```
<?xml version="1.0"?>
<Test>
      <Version>
            1.0
      </Version>
      <Encrypt>
            No
      </Encrypt>
      <Password>
            apples
      </Password>
      <Description>
            Intro to Memphis - Test 1
      </Description>
      <Mode>
            Trial
      </Mode>
      <EMail>
            imabyte@cbu.edu
      </EMail>
      <Time>
            60
      </Time>
      <TotalPoints>
            100
      </TotalPoints>
```

```
<Question>
    <Sort>
        1
    </Sort>
    <Keyword>
        Beale Police
    </Keyword>
    <Problem>
        What is located at the star on the map ?
    </Problem>
    <Image>
        downtown.gif
    </Image>
    <Chapter>
        1
    </Chapter>
    <PageRef>

    </PageRef>
    <Difficulty>
        Hard
    </Difficulty>
    <Type>
        Multiple
    </Type>
    <Points>
        50
    </Points>
    <ShortAnswer>

    </ShortAnswer>
    <Explain>
        The map shows the downtown area around Beale.
    </Explain>
    <CorrectChoice>
        2
    </CorrectChoice>
    <Choice>
        Graceland
    </Choice>
    <Choice>
        Beale Street Police Station
    </Choice>
    <Choice>
        CBU
    </Choice>
</Question>
```

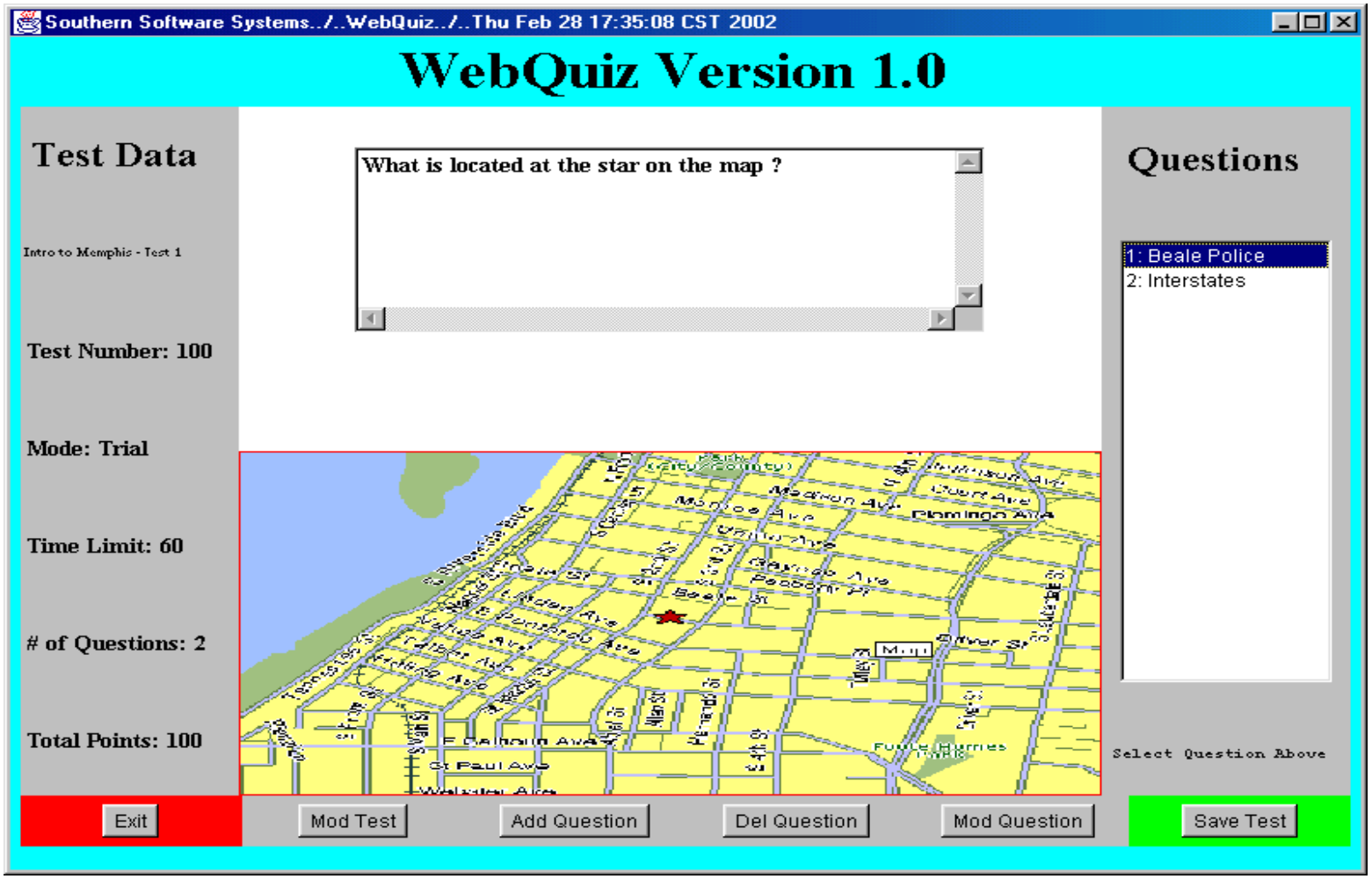
```
        <Question>
                        <Sort>
                                2
                        </Sort>
                        <Keyword>
                                Interstates
                        </Keyword>
                        <Problem>
                                What is the junction of the two interstates on the map called ?
                        </Problem>
                        <Image>
                                junction55_240.gif
                        </Image>
                        <Chapter>
                                3
                        </Chapter>
                        <PageRef>
                                34
                        </PageRef>
                        <Difficulty>
                                Medium
                        </Difficulty>
                        <Type>
                                Multiple
                        </Type>
                        <Points>
                                50
                        </Points>
                        <ShortAnswer>

                        </ShortAnswer>
                        <Explain>

                        </Explain>
                        <CorrectChoice>
                                1
                        </CorrectChoice>
                        <Choice>
                                Malfunction Junction
                        </Choice>
                        <Choice>
                                Mixmaster
                        </Choice>
                        <Choice>
                                Disaster Zone
                        </Choice>
                        <Choice>
                                Ground Zero
                        </Choice>
            </Question>
    </Test>
```

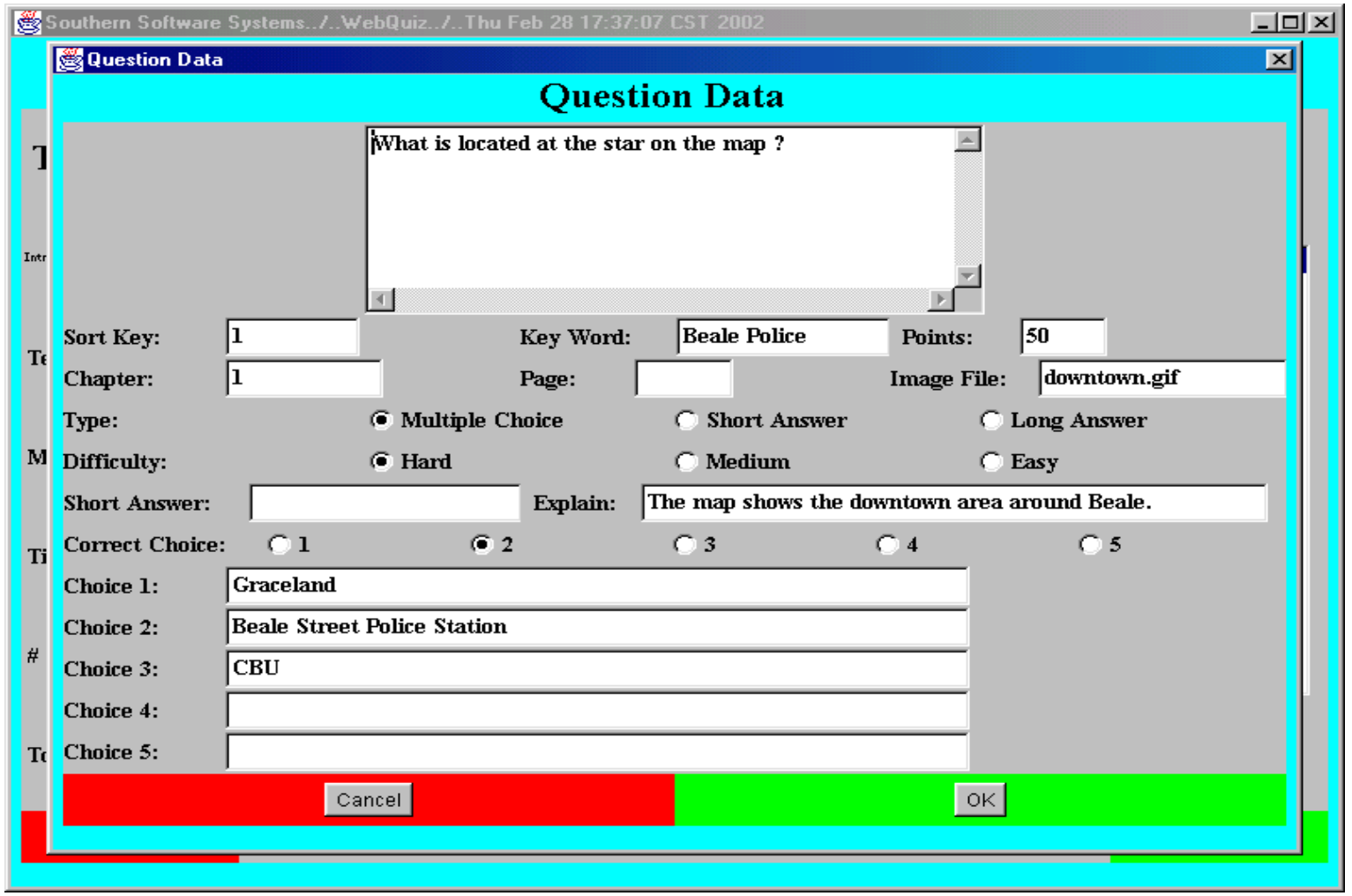
Test Data After Parsing into Web Page

[Uses embedded special purpose parser in a Java Applet]



Question Data After Parsing into Web Page

[Java Applet]



Server Processing

- The most common processing technique on the server side is for a Java Servlet to read the XML file and then dynamically create an HTML page which it sends to the client browser
- This can also be done with Java Server Pages or PHP (discussed in a later session)

DTD & Schemas

- Both DTD's and Schema's can be used to describe the valid contents of an XML document
- DTD is the original method based on SGML
- Version 1.0 of XML Schema was adopted in May of 2001 and is the newer method
- A problem with DTD was that they were not themselves written in XML which made it difficult for servers to standardize the formulation and exchange of document specifications

Document Type Definition (DTD Files)

- A DTD is optional, but is needed for “standardized” types of information exchange
- DTD’s can be used to “validate” an XML document or to facilitate production of XML files
- The DTD is optional, but a “type-valid” XML document must have a DTD or XML Schema
- The DTD can be in the XML document or be referenced via a URL in the prolog section of the XML file:
  - `<?xml version = "1.0”?>`
  - `<!DOCTYPE letter SYSTEM "letter.dtd">`
- For database applications of XML, the DTD corresponds to a view (“Create View xxx AS…”

DTD (con't)

- Specifying the rules that structure a document is done with the EBNF (Extended Backus-Naur Form) grammar [covered in CS/ITM 171/172 courses & textbook]:
  - For parent elements:
    - <!ELEMENT elementName (subElement1@, subElement2@, …)>
    - Where @ is blank for one subElement of that type, + for one or more, * for any number, or ? for zero or one
  - For child elements:
    - <!ELEMENT elementName dataSpecs>
  - For attributes of an element:
    - <!ATTLIST elementName attributeName dataSpecs>
  - dataspecs involve a data type (CDATA for character data [including 'variables'], #PCDATA for parseable character data [text only], EMPTY, enumerated lists, etc.) and a default value type (#IMPLIED – not required, #REQUIRED, "specificDefaultValue", etc.)
  - See XML references for full specifications

XML Document and its DTD

- <? xml version="1.0"?>
- <!DOCTYPE dept SYSTEM "dept.dtd">
- <dept>
  - <emp id ="jd">
    - <name>John Doe</name>
    - <email> jd@abc.com</email>
  - </emp>
  - <emp id ="ib"
    - <name>Ima Byte>
    - <url href="www.abc.com/~ib/"/>
  - </emp>
- </dept>

- <!ELEMENT dept (emp*)>
- <!ELEMENT emp (name, (email | url))>
- <!ATTLIST emp id CDATA #REQUIRED)
- <!ELEMENT name (#PCDATA)>
- <!ELEMENT email (#PCDATA)>
- <!ELEMENT url EMPTY>
- <!ATTLIST url href CDATA #REQUIRED>

DTD for Letter

```
<!ELEMENT letter (contact+, paragraph+)>

<!ELEMENT contact (name, address1, address2, city, state,
   zip, phone, flag)>
<!ATTLIST contact type CDATA #IMPLIED>

<!ELEMENT name (#PCDATA)>
<!ELEMENT address1 (#PCDATA)>
<!ELEMENT address2 (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT flag (EMPTY)>
<!ATTLIST flag id CDATA #IMPLIED>

<!ELEMENT paragraph (#PCDATA)>
```

```
<?xml version = "1.0"?>
<!DOCTYPE letter SYSTEM "letter.dtd">

<letter>

   <contact type = "from">
      <name>John Doe</name>
      <address1>123 Main St.</address1>
      <address2></address2>
      <city>Anytown</city>
      <state>Anystate</state>
      <zip>12345</zip>
      <phone>555-1234</phone>
      <flag id = "P"/>
   </contact>

   <contact type = "to">
      <name>Joe Schmoe</name>
      <address1>Box 12345</address1>
      <address2>15 Any Ave.</address2>
      <city>Othertown</city>
      <state>Otherstate</state>
      <zip>67890</zip>
      <phone>555-4321</phone>
      <flag id = "B"/>
   </contact>

   <paragraph>Dear Sir,</paragraph>

   <paragraph>It is our privilege to inform you about our new
      database managed with XML. This new system will allow
      you to reduce the load of your inventory list server by
      having the client machine perform the work of sorting
      and filtering the data.</paragraph>
   <paragraph>Sincerely, Mr. Doe</paragraph>

</letter>
```

XML file for a letter with external DTD specified

Using DTD's to generate XML Files

- A common way XML files are produced is by a program (ie C++, PHP, or Java program) which:
  - inputs an SQL statement and a DTD file (specifying the format of the output XML data)
  - retrieves the indicated data from the relational database
  - and outputs the data in the XML format

Standardized Markup Languages

- MathML for math expressions
  - Replaces earlier special packages like TeX
- FpML – Financial Products Markup Language – exchange of financial information
- ebXML - Electronic Business XML, see next slide

ebXML


ebXML - Enabling A Global Electronic Market - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss Real.com
Address http://www.ebxml.org/
Go Links
ebXML
ENABLING A GLOBAL ELECTRONIC MARKET
ABOUT | INDUSTRY SUPPORT | NEWS | SPECS | REPORTS | REFERENCE | WHITE PAPERS | TECHNICAL WORK
ebXML
About
Industry Support
News and Articles
Implementations
FAQ
Contacts
Logo
Presentations
Events
Resources
Initiative Archive
SPECS/DOCUMENTS
Specifications
Technical Reports
Reference Materials
White Papers
TECHNICAL WORK
Overview
Business Process
Core Components
ebXML enables enterprises of any size, in any location to meet and conduct business through the exchange of XML-based messages.
Development of the ebXML specifications is an on-going effort sponsored by OASIS and UN/CEFACT. Technical committees for the ebXML Registry, Messaging, Collaborative Partner, and Implementation are hosted by OASIS, and Business Process and Core Component work continues at UN/CEFACT.
JOINTLY SPONSORED BY
UN/CEFACT
OASIS
ebXML NEWS
[12 June 2002] boleroXML takes further steps towards full ebXML convergence
[07 June 2002] Hong Kong University's E-
SPEC STATUS
ebXML Registry specs, RIM v2 and RS v2, have been approved as OASIS Standards
ebXML Messaging v2 has
ebXML IMPLEMENTATIONS
Sybase provides an open source implementation of the ebXML Messaging Service Spec 2.0 for its developer community.
More implementations
Internet

Other Industry Custom DTD's

- Industry consortiums have developed to define DTD's that will be used for commerce and other communication needs on a particular industry or group. Some are:
  - RosettaNet – standard DTD's and process flow for buyers and sellers
  - XEDI – standard mapping from legacy systems to XML (Aerospace Industries Association)
  - BizTalk – standard XML grammar for processing non-XML data
  - XBRL (formerly XFRL) – standard DTD for reporting financial data
  - PDML – Product Data Markup Language

XML Schemas

- The newer way to define an XML document content's is with XML Schemas
- These schemas are themselves XML documents
- XML Schemas use namespaces and can more precisely define document contents in a variety of ways such as specific min and max cardinalities on elements, more data types (int, float, boolean, etc.), regular expressions, inheritance of schemas, etc.

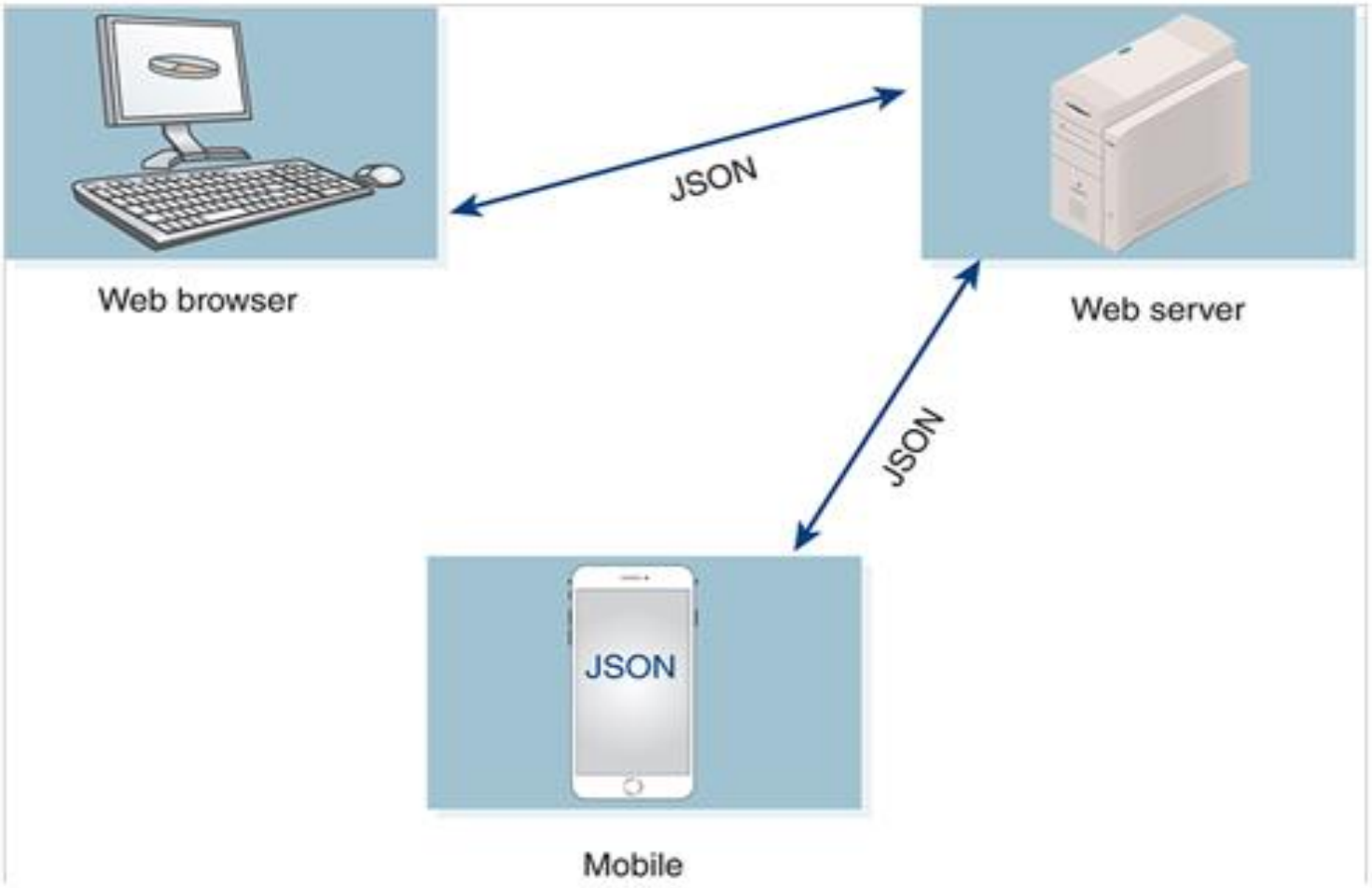
Extensible Style Language (XSL)

- Another way to express XML document format is via XSL (Extensible Style Language); an HTML document can have multiple XSL's (such as different display or data sort orders)
- CSS can also be used
- CSS and XSL data can be in the XML document or referenced separately via URL
- A subset of XSL called XSLT can be used to transform an XML document into a target format (ie HTML document or other formats (ie pdf))

JSON [http://www.json.org/]

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
- It is easy for humans to read and write, and it is easy for machines to parse and generate
- It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999
- JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
- JSON is built on two structures:
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence
- These are universal data structures, and virtually all modern programming languages support them in one form or another

JSON (con't)



Example JSON Format Data

- { "firstName": "John",
- "lastName": "Smith",
- "isAlive": true,
- "age": 25,
- "address":
  - { "streetAddress": "21 2nd Street",
  - "city": "New York",
  - "state": "NY",
  - "postalCode": "10021-3100" },
- “phoneNumbers": [
  - { "type": "home",
  - "number": "212 555-1234" },
  - { "type": "office",
  - "number": "646 555-4567" }]
- }

JSON (con't)

A JSON object looks like a JavaScript object, in that it is grouped by left and right braces { } and consists of comma-separated name–value pairs. The following JSON object, for example, contains employee data:

```
{"name": "Bill Smith", "age": 50, "ID": 123456, "Salary":
105000, "Email": "BSmith@Company.com"}
```

Within a JSON object, you group each field name within quotes, separating the name from its value with a colon (:). The previous object consists of four fields. Within a JSON object, you separate name–value pairs with commas. JSON supports a variety of value types. In this case, the object uses numbers and strings.

The following JSON object represents a student:

```
{"name": "Laurie Jones", "GPA": 3.5, "Grade": 97}
```

As you can see, the student object contains three name–value pairs. The first, name, uses a string value. The second and third, GPA and Grade, use numeric values.

JSON (con't)

JSON Field Data Types

JSON objects consist of name–value pairs. Each pair has a unique quoted name and a value. Each value must be one of the following data types:

- String: Contains a value within quotes
- Number: An integer or floating-point value
- Object: A nested object enclosed in left and right braces { }
- Array: A collection of values grouped in left and right brackets [ ]
- Boolean: A true or false value
- Null: Represented by the keyword null

The following JSON book object represents the use of each JSON data type:

```
{
    "Title": "Data Mining and Analytics",
    "Chapters": 16,
    "Author": {"Last": "Jamsa", "First": "Kris", "ID": 1983},
  "OtherBooks": true,
    "Universities": [ "ASU", "UNLV", "San Diego State
University"],
     "NextBook": null
  }
```

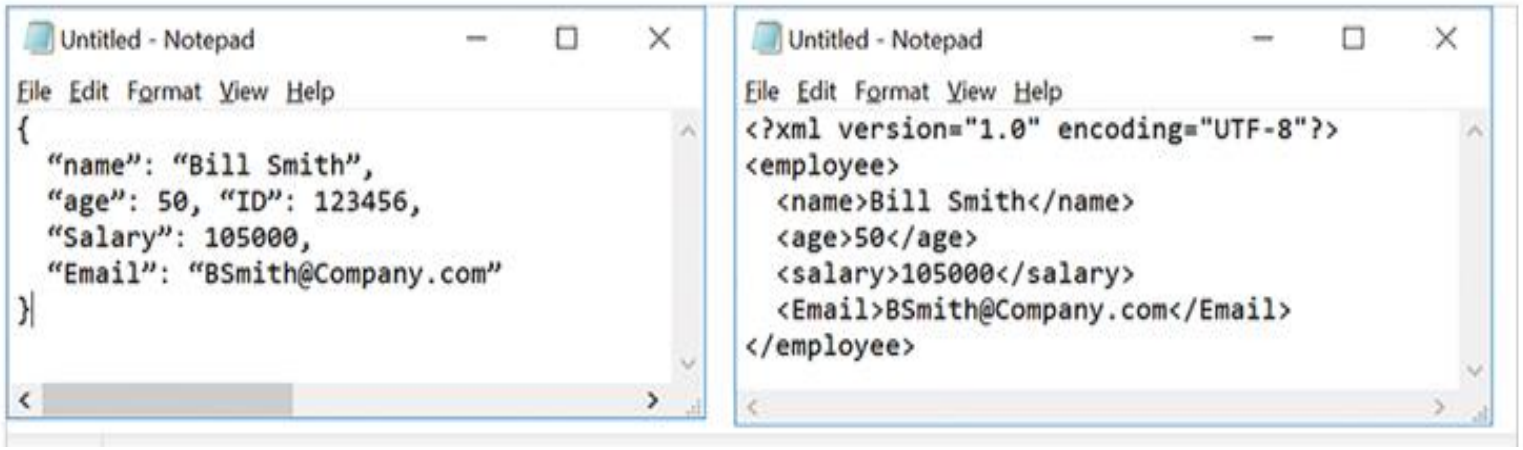
JSON (con't)

JSON Is Self-Describing, Human Readable, Language Independent, and Lightweight

A key feature of JSON is that JSON objects are self-describing, meaning there is not an overall schema that describes what each object looks like, but rather, the object structure is described by the groupings of name–value pairs found within the object itself.

JSON objects are also human readable, which means (after a little practice) that you can look at an object and identify the object's name–value pairs. In addition, JSON objects are language independent, which means you can create and use JSON objects in Java, C#, Python, and more.

Extensible Markup Language (XML) is also self-describing, human readable, and language independent. As such, developers must often choose whether they will use JSON or XML to



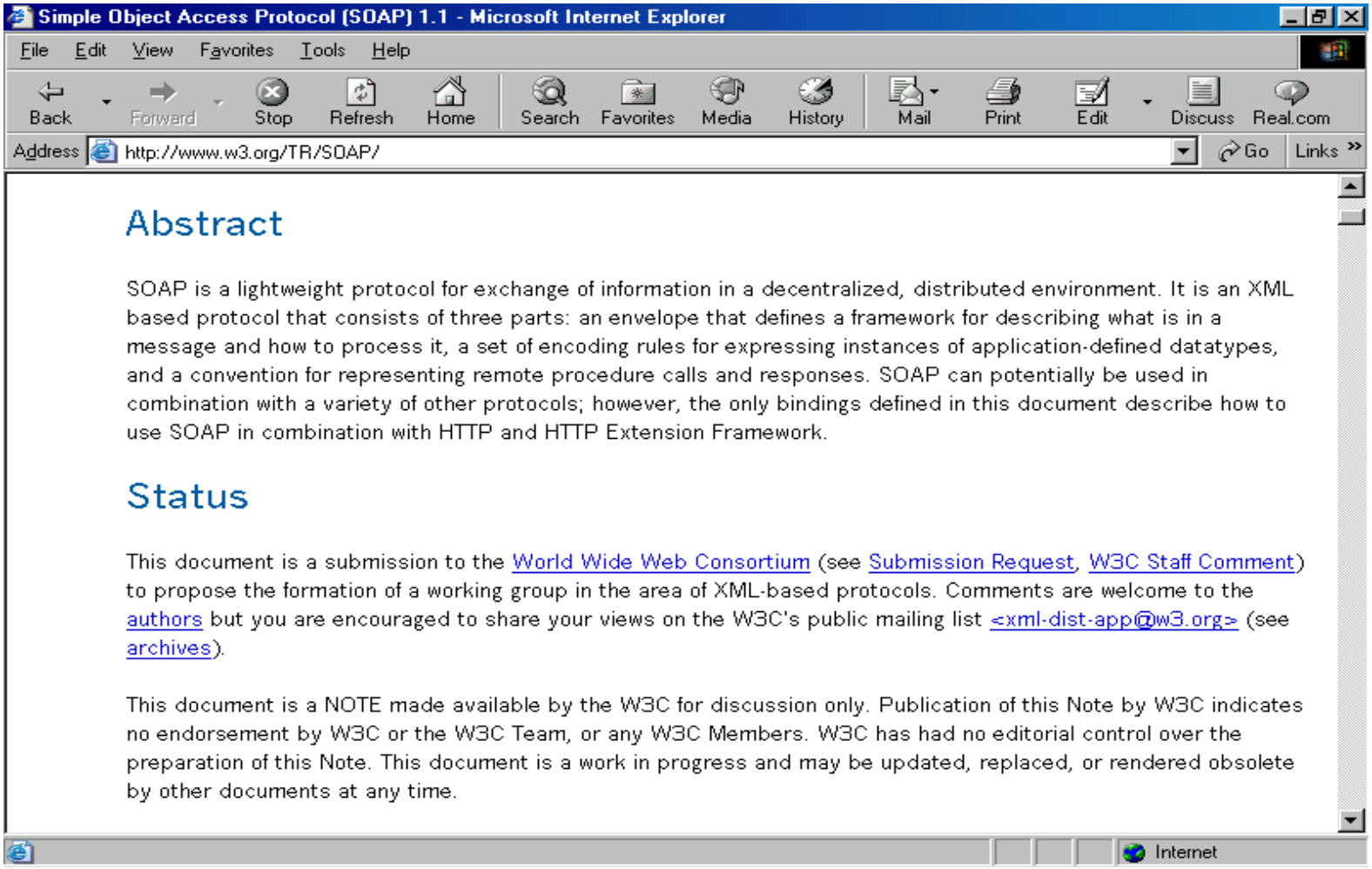
JSON & JavaScript

- Since JSON was derived from JavaScript and its syntax is (mostly) a subset of the language, it is often possible to use the JavaScript eval() function to parse JSON data
  - var p = eval('(' + json_string + ')');
- This is unsafe if the string is untrusted
- Instead, a JSON parser library or JavaScript's native JSON support should be used for reading and writing JSON
  - var p = JSON.parse(json_string);
- A correctly implemented JSON parser only accepts valid JSON, preventing potentially malicious code from being inadvertently executed
- Since 2010, web browsers such as Firefox and Internet Explorer have included support for parsing JSON
- As native browser support is more efficient and secure than eval(), native JSON support is included in Edition 5 of the ECMAScript standard

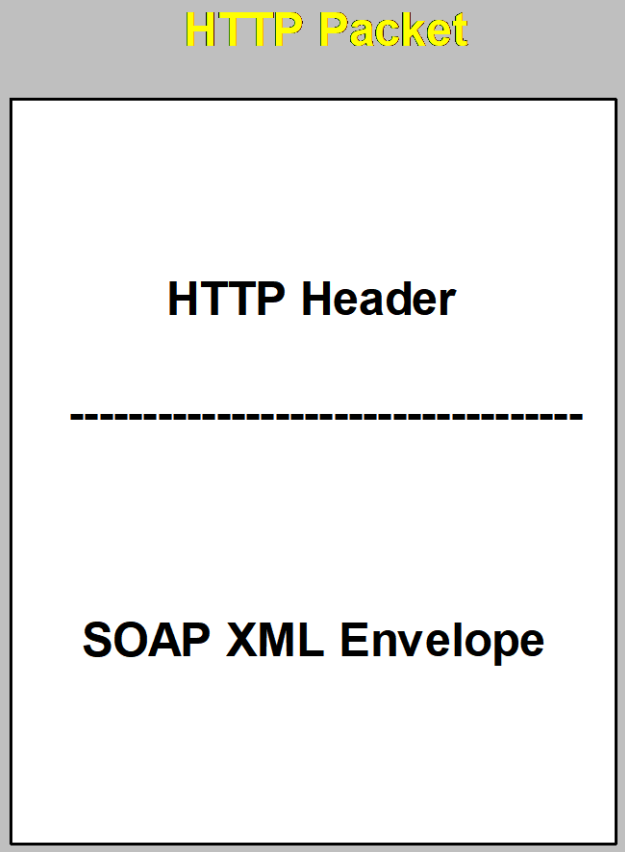
Simple Object Access Protocol

- “SOAP” is a ‘lightweight’ protocol for information exchange in a distributed environment; instead of a more full featured ‘heavyweight’ protocol such as DCOM, CORBA-IIOP, Java RMI
- Encodes information in an XML or JSON wrapper, thus is text based and vendor neutral
- Typically used for RMI (remote method invocation)
  - Interpret a remote method’s parameter needs
  - Place those parameters in an XML document using a specific layout to invoke the remote method (“web services”)
- SOAP uses HTTP protocol

www.w3.org/tr/soap



SOAP Uses HTTP Protocol



HTTP header indicates destination, content format (XML here), …

SOAP Envelope

- <?xml>
- <soap:Envelope …>
  - <soap:Header>
    - <soap:Body>
      - …
    - </soap:Body>
  - </soap:Header>
- </soap:Envelope>

- Header is optional
- For method call: first child in body has to be name of the remote method
- For method return info: first node in body is response name and next node is return data

Example Remote Procedure Call

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Remote method call

Remote method arguments

Example Return Information

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/enc
oding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-
URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Method response

Method return data

Web Services

- Web Services are applications that can interact with each other across the Intra/Internet using open Web Standards
  - Self-describing
  - Platform and implementation neutral
  - Developed using open standards for description, discovery, and invocation
  - Connect business partners' business processes together
  - Typically transactional, requiring integration with existing systems

Web Services (con't)

- One of the most important ideas in creating scalable Web sites is to decompose programs into modular Web services
- The Web pages will no longer read a database directly, process the business logic and spew out HTML
- Instead we would subdivide our pages into multiple sections, and render each section by calling different Web services
- In this the load is spread across multiple servers

Web Services - Basics

- Expose services to other processes
  - Internet and intranet
- Black boxes (Encapsulation)
  - Component-like, reusable
- Not dependent on one platform
  - .NET, J2EE, Open source (Apache, PHP)
- Based on open standards
  - HTTP, XML, WSDL, UDDI, and SOAP

Based on XML & SOAP

- Used for data formatting
- A consistent means of describing the data layout
- A means to build self-describing data sets
- A text file format
- Easily parsed by industry standard methods
- Operating system and language independent

Other Related Standards

- UDDI (Universal Description, Discovery, and Integration) - allows businesses to register with a directory for advertising the services
- WSDL (Web Service Description Language) - defines the service interface and its implementation characteristics
- SOAP (Simple Object Access Protocol) - allows applications to call object methods, or functions, residing on remote servers

Web Services Infrastructure

REST

- **Representational state transfer** (**REST**) is a software architectural style that defines a set of constraints to be used for creating Web services
- Web services that conform to the REST architectural style, called *RESTful* Web services, provide interoperability between computer systems on the Internet
- RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations
- Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations

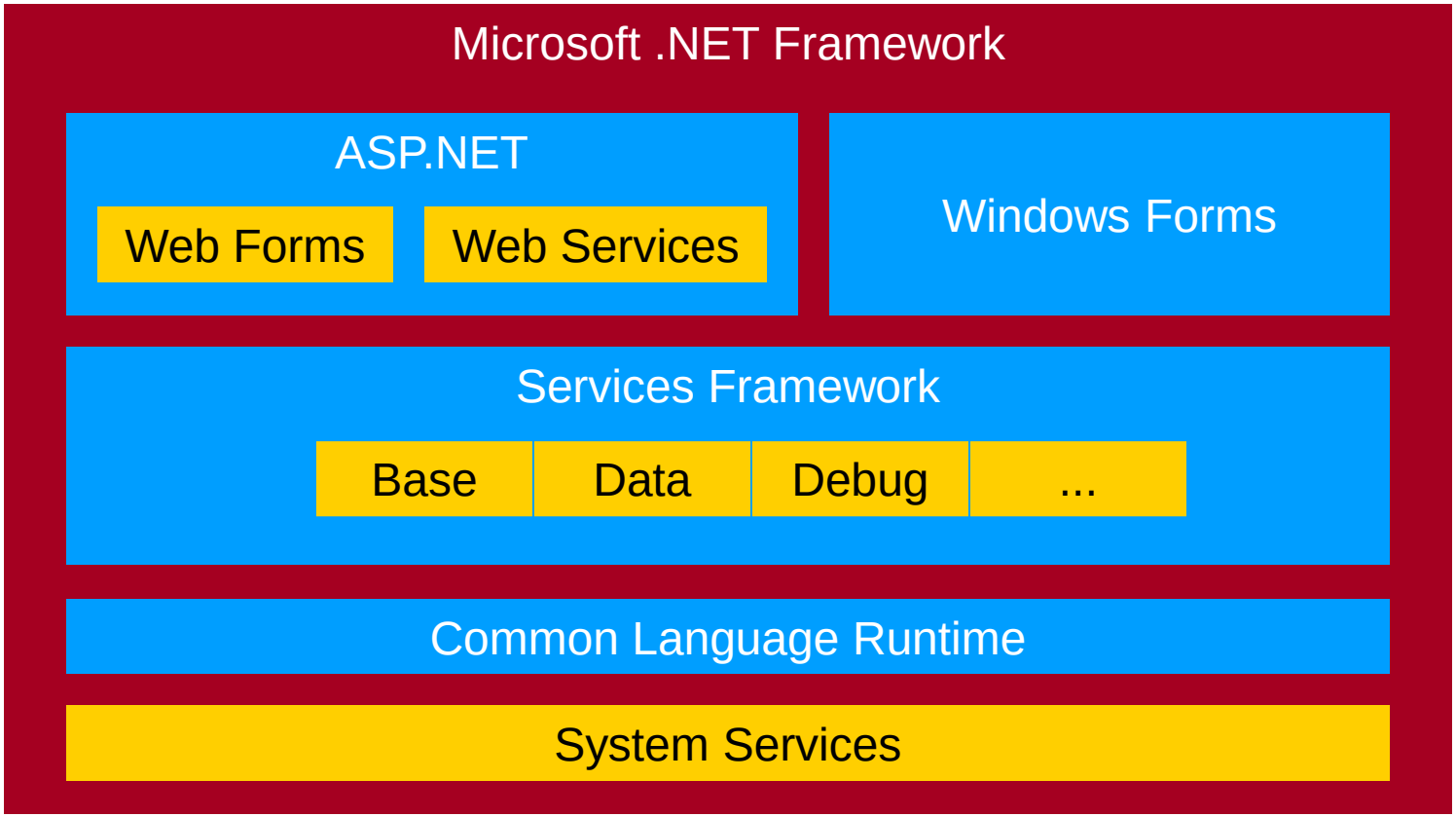
Web Services Benefits

- Finding New Partners and Interoperability
- Universal Accessibility – anywhere, anytime, any platform
- Universally Acceptable Standards
- Efficient Application Sharing – existing application data can be shared as identical platform is not required on both ends

Barriers to Diffusion of Web Services

- Forrester Research:
  - Security Issues (27%)
  - Lack of knowledge (27%)
  - Limited tools (19%)
  - Employee resistance (15%)
  - No business case (12%)
  - Lack of standards (7%)
  - No services (7%)

The .NET Framework Architecture (Microsoft)



Open Source ->XML-RPC

- XML-RPC is a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.
- It's remote procedure calling using HTTP as the transport and XML as the encoding.
- It's designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

XML-RPC (con't)

- Here's a simple PHP example that calls a function:
  - examples.getStateName(48)
- residing at betty.userland.com/RPC2 to get an American state based on an index:
  - include("xmlrpc.inc");
  - $thestate=48;
  - $f=new **xmlrpcmsg**('examples.getStateName', array(new xmlrpcval($thestate, "int")));
  - $c=new **xmlrpc_client**("/RPC2", "betty.userland.com", 80);
  - $r=$c->send($f); $v=$r->value();
  - if (!$r->faultCode()) {
    - print "State number $thestate is " . $v->scalarval() . "
"; print "<HR>I got this value back
<PRE>" . htmlentities($r->serialize()). "</PRE><HR>\n";
  - } else {
    - print "Fault: "; print "Code: " . $r->faultCode() . " Reason '" .$r->faultString()."'
";
  - }

XML-RPC (con't)

- This is the XML generated by xmlrpcmsg():
  - <?XML VERSION="1.0"?>
  - <methodCall>
    - <methodName>
      - examples.getStateName
    - </methodName>
    - <params>
      - <param> <value><i4>**48**</i4></value> </param>
    - </params>
  - </methodCall>

XML-RPC (con't)

- Creating a Web service is as simple as creating a Web Services API and receiving calls.
- Suppose we have a web server at 192.168.0.1 and we create an XML-RPC server file named rpc2.php to handle example.getStateName() requests:
  - include("xmlrpc.inc"); include("xmlrpcs.inc");
function GetStateName($params) {
    - $statenum = $params->getParam(0); switch($statenum)
      - { case 1: $name="Alabama"; break;
      - : //…
      - case 50: $name="Wyoming"; break;
      - default: $name="Unknown"; break; }
      - return new **xmlrpcresp**(new xmlrpcval($name, "string")); }
$s=new **xmlrpc_server**(array("examples.getStateName" => array("function" => "GetStateName")));
- We could call this function using:
  - include("xmlrpc.inc");
  - $thestate=48;
  - $f=new **xmlrpcmsg**('examples.getStateName', array(new xmlrpcval($thestate, "int")));
  - $c=new **xmlrpc_client**("/rpc2.php", "192.168.0.1", 80);

References

- Amazon Web Services in Action by Andreas Wittig and Michael Wittig
- RESTful Web Services by Leonard Richardson , Sam Ruby , et al
- Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services (Addison-Wesley Signature Series (Fowler)) by Robert Daigneau

Homework

- No assignments for this lesson