Web Databases and Applications

Web Databases and XML

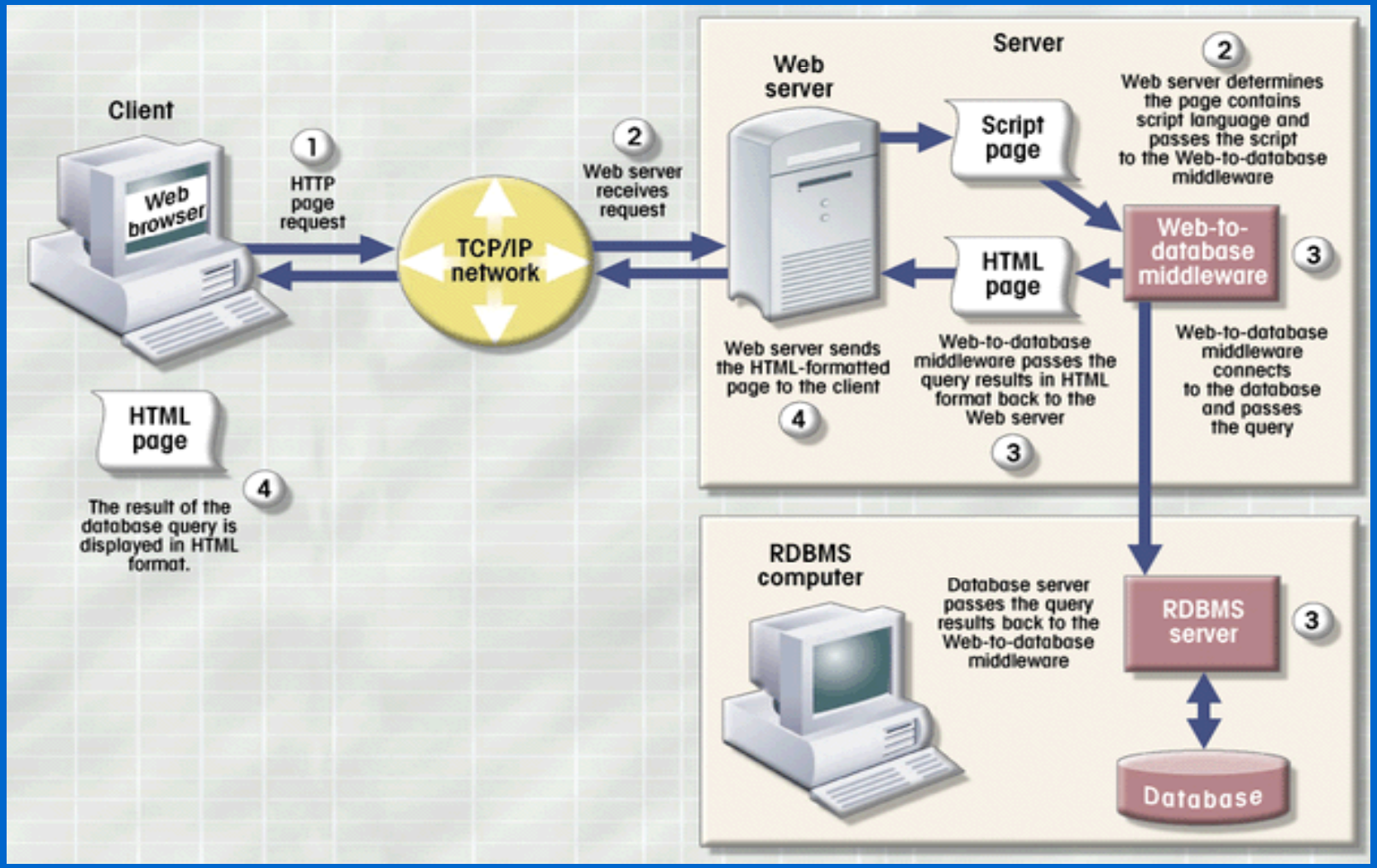
Learning Objectives

- An understanding of Web Databases in general
- XML Basic
- References
  - *Ramakrishnan, Gherke: Database management Systems, Chapter 7 and Chapter 27*
  - *W3C Recommendations at http://www.w3c.org/TR/*

What is WDB?

- Database you access through the Web by filling in a form on a Web page
- Usually resides on a database server, a computer that stores and provides access to a database

What is WDB?


Client
Web browser
1
HTTP page request
TCP/IP network
2
Web server receives request
Server
Web server
Script page
2
Web server determines the page contains script language and passes the script to the Web-to-database middleware
Web-to-database middleware
3
HTML page
Web server sends the HTML-formatted page to the client
4
Web-to-database middleware passes the query results in HTML format back to the Web server
3
Web-to-database middleware connects to the database and passes the query
HTML page
4
The result of the database query is displayed in HTML format.
RDBMS computer
Database server passes the query results back to the Web-to-database middleware
RDBMS server
3
Database

What is WDB?

- **What are web databases?**
  - Two technologies come together
  - Databases
    - Network, Hierarchical, Relational, Object-oriented
    - Systems use for storing, organizing and manipulating data
    - Most businesses have databases for their operations
  - **World-Wide Web (WWW)**
    - Before the WWW, it was hard to access databases in different networks
    - After mid-1990's, there is almost a web browser accessed by every user
    - People can reach almost sites globally to get products and services
  - Types
    - Using Web as a frontend (Database-to-Web)
    - Using Web as a medium (Database-to-Application-to-Database)

Why we need Web Database?

- Internet and corporate intranets offer services like:
  - Purchasing books online,
  - Online auctions,
  - Online submission of bids,
  - Distant learning
- The first generation of Internet sites were collections of HTML files and these proved to be inadequate:
  - No declarative query languages,
  - No easy ways of data updates,
  - No database transaction behavior,
  - Semantically completely unstructured (HTML mark-up is for presentation only)

Why we need Web Database?

- Modern electronic commerce sites rely on database systems

- These pose new challenges on DBMS:
  1. Large number of concurrent users (scalability),
  2. Storing and handling unstructured and semistructured documents
  3. Ranked keyword search

Introduction to XML

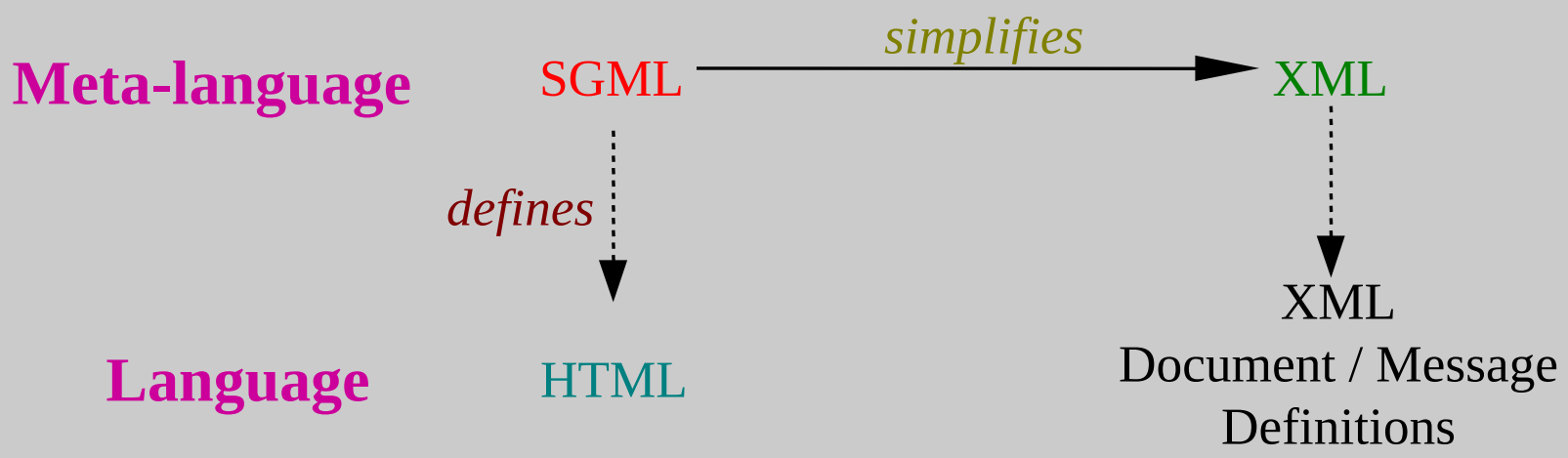
- Stands for eXtensible Markup Language
- Based on Standard Generalized Markup Language (SGML)
- Version 1.0 and 1.1, recommended by W3C
- XML is a meta-markup language
- derived or application languages
  - XBRL, SMIL, ebXML, VRML

Why XML Database?

- The main motive for introducing XML was to enable a seamless flow of data between parties communicating over Internet
- But soon, people realized that they need not only to exchange data, but also to:
  - Store,
  - Query,
  - Update,
  - Protect …
- Hence, database research for its specific requirements

XML versus HTML

- Misconception:
  - XML is a generalized HTML
- Fact:
  - XML is simplified from SGML
  - HTML is a specific *definition* of SGML
- Compare XML and HTML in the context of their popularity and future!



Introduction

- XML
  - Technology for creating markup languages
  - Enables document authors to describe data of any type
  - Allows creating new tags
    - HTML limits document authors to fixed tag set

Introduction to XML Markup

- XML document (**`intro.xml`**)
  - Marks up message as XML
  - Commonly stored in text files
    - Extension **`.xml`**

```
1  <?xml version = "1.0"?>
2
3  <!-- Fig. 5.1 : intro.xml
4  <!-- Simple introduction to XML markup -->
5
6  <myMessage>
7     <message>Welcome to XML!</message>
8  </myMessage>
```

Document begins with declaration that specifies XML version 1.0

Element `message` is *child element* of *root element* `myMessage`

Line numbers are **not** part of XML document. We include them for clarity.

Line numbers are not **part of XML document. We include them for clarity.**
Document begins with *declaration* that specifies XML version 1.0
Comments
Element `message` **is *child element* of *root element*** `myMessage`

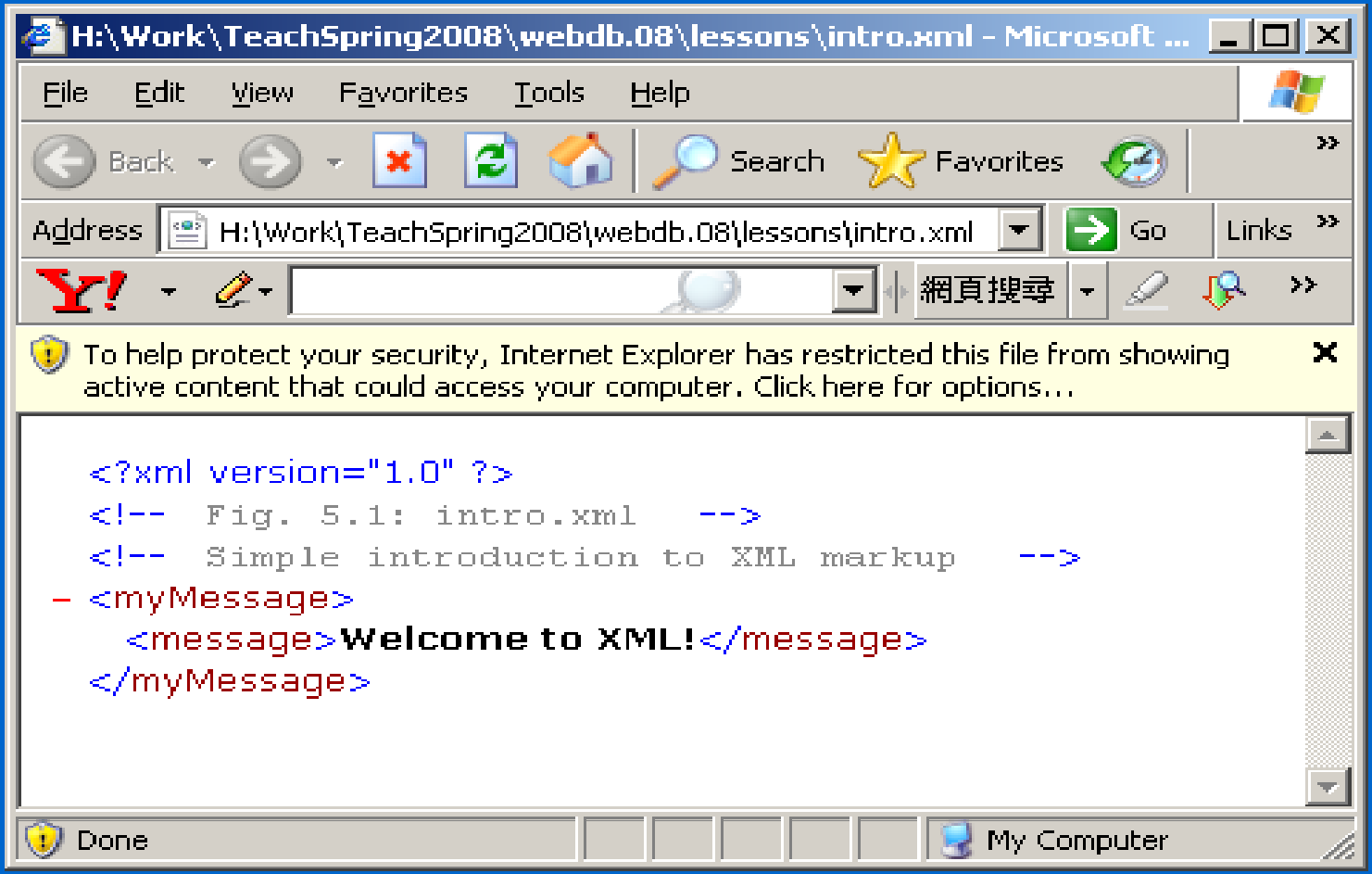
Introduction to XML Markup

- XML documents
  - Must contain exactly one *root element*
    - Attempting to create more than one root element is erroneous
  - Elements must be nested properly
    - Incorrect: `<x><y>hello</x></y>`
    - Correct: `<x><y>hello</y></x>`

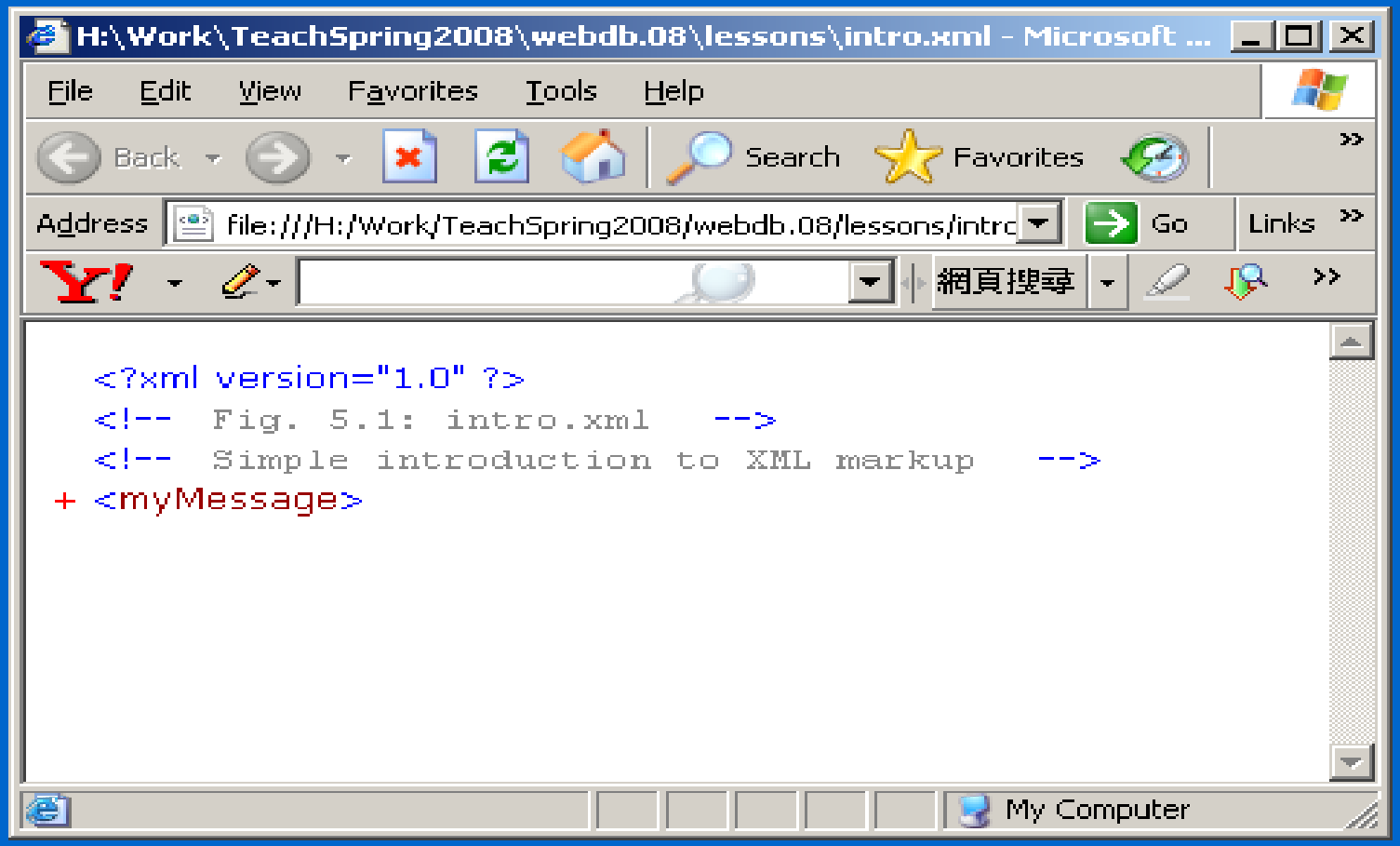
Parsers and Well-formed XML Documents

- XML document syntax
  - Considered *well formed* if syntactically correct
    - Single root element
    - Each element has start tag and end tag
    - Tags properly nested
    - *Attribute* (discussed later) values in quotes
    - Proper capitalization
      - Case sensitive

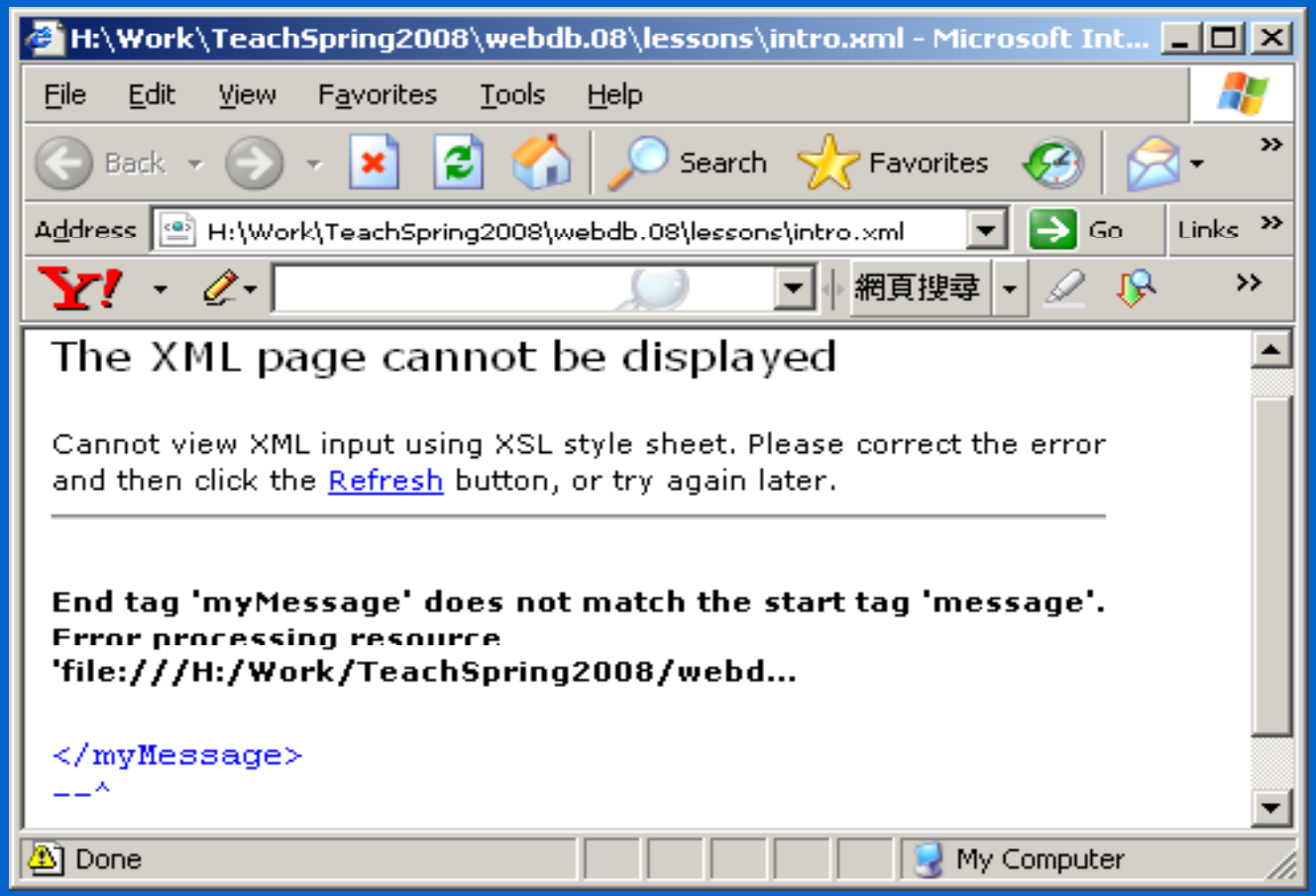
XML document shown in IE



XML document shown in IE



Error message for a missing end tag



Characters

- Character set
  - Characters that may be represented in XML document
    - e.g., ASCII character set
      - Letters of English alphabet
      - Digits (**`0-9`**)
      - Punctuation characters, such as **`!`**, **`-`** and **`?`**
- XML documents may contain
  - Carriage returns
  - Line feeds
  - *Unicode* characters
    - Enables computers to process characters for several languages

Characters vs. Markup

- XML must differentiate between
  - Markup text
    - Enclosed in angle brackets (**<** and **>**)
      - e.g,. Child elements
  - Character data
    - Text between start tag and end tag
      - e.g. line 7: **`Welcome to XML!`**

White Space, Entity References and Built-in Entities

- Whitespace characters
  - Spaces, tabs, line feeds and carriage returns
    - *Significant* (preserved by application)
    - *Insignificant* (not preserved by application)
      - Normalization
        - Whitespace collapsed into single whitespace character
        - Sometimes whitespace removed entirely

```
<markup>This is character     data</markup>
```

after normalization, becomes

```
<markup>This is character data</markup>
```

White Space, Entity References and Built-in Entities

- XML-reserved characters
  - Ampersand (**&**)
  - Left-angle bracket (**<**)
  - Right-angle bracket (**>**)
  - Apostrophe (**'**)
  - Double quote (**"**)

White Space, Entity References and Built-in Entities

- Entity references
  - They are markup that is replaced with character data when the document is parsed
  - Wherever an entity reference appears in an XML document, it is textually replaced by its content
- Entity references are used in XML documents in place of specific characters (like: <, “,…) that would otherwise be interpreted as part of markup
- Example:
  - Allow to use XML-reserved characters
    - Begin with ampersand (**&**) and end with semicolon (**;**)
  - Prevents from misinterpreting character data as markup

White Space, Entity References and Built-in Entities

- Build-in entities
  - Ampersand (**`&`**)
  - Left-angle bracket (**`<`**)
  - Right-angle bracket (**`>`**)
  - Apostrophe (**`'`**)
  - Quotation mark (**`"`**)
  - Mark up characters "**`<>&`**" in element **`message`**

```
<message>&lt;&gt;&amp;</message>
```

Using Unicode in an XML Document

- XML Unicode support
  - Figure shown in next slide encodes Arabic words
    - Arabic characters
      - represented by entity references for Unicode characters

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

<!-- Fig. 5.4 : lang.xml    -->
<!-- Demonstrating Unicode  -->

<!DOCTYPE welcome SYSTEM "lang.dtd">

<welcome>
   <from>

      <!-- Deitel and Associates -->
      &#1583;&#1575;&#1610;&#1578;&#1614;&#1604;
      &#1571;&#1606;&#1583;

      <!-- entity -->
      &assoc;
   </from>

   <subject>

      <!-- Welcome to the world of Unicode -->
      &#1571;&#1607;&#1604;&#1575;&#1611;
      &#1576;&#1603;&#1605;
      &#1601;&#1610;&#1616;
      &#1593;&#1575;&#1604;&#1605;

      <!-- entity -->
      &text;
   </subject>
</welcome>
```

Document type definition (DTD) defines document structure and entities

XML document that contains Arabic words.

C:\Documents and Settings\administrator\Desktop\xmlhtp1\xmlhtp1_examples\ch05\...

File Edit View Favorites Tools Help

Back Search Favorites History Links

```
  <?xml version="1.0" ?>
  <!--  Fig. 5.4 : lang.xml        -->
  <!--  Demonstrating Unicode      -->
  <!DOCTYPE welcome (View Source for full doctype...)>
- <welcome>
  - <from>
      <!--  Deitel and Associates   -->
      دايتل أند
      <!--  entity   -->
      أسوشيتس
    </from>
  - <subject>
      <!--  Welcome to the world of Unicode   -->
      أهلاً بكم في عالم
      <!--  entity   -->
      اليونيكود
    </subject>
  </welcome>
```

My Computer

Markup

- XML element markup
  - Consists of
    - Start tag
    - Content
    - End tag
  - All elements must have corresponding end tag

    ```
    <img src = “img.gif”>
    ```

    is correct in HTML, but not XML
  - XML requires end tag or *forward slash* (**/**) for termination

    ```
    <img src = “img.gif”></img>
    ```

    or

    ```
    <img src = “img.gif”/>
    ```

    is correct XML syntax

Markup

- Elements
  - Define structure
  - May (or may not) contain content
    - Child elements, character data, etc.
  - Each XML document has exactly one root element
    - A root element contains all other elements of a document as its content
  - Elements that appear inside another element can have more than one instance inside the same document

Markup

- Attributes
  - Describe elements
    - Elements may have associated attributes
  - Placed within element's start tag
  - Values are enclosed in quotes
    - Element **`car`** contains attribute **`doors`**, which has value **`"4"`**

```
<car doors = "4"/>
```

- Attributes are appropriate for very simple data that:
  - Have no substructure, and
  - Have a repetition rate of at most one, or
  - Bear information about the document itself

Which is better – Elements vs Attributes?

- Using attributes instead of elements results in a more compact document structure
  - But attribute values are flat text
- Elements allow structure and repetition
  - So, whenever something:
    - Has subordinated objects,
    - Contains more than one component, or
    - Appears more than once

  elements and not attributes should be used
  - Also, elements are more appropriate for later extensions of a document

Markup

- Processing instruction (PI)
  - Passed to application using XML document
  - Provides application-specific document information
  - Delimited by **<?** and **?>**

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

<!-- Fig. 5.5 : usage.xml              -->
<!-- Usage of elements and attributes -->

<?xml:stylesheet type = "text/xsl" href = "usage.xsl"?>

<book isbn = "999-99999-9-X">
   <title>Deitel&amp;s XML Primer</title>

   <author>
      <firstName>Paul</firstName>
      <lastName>Deitel</lastName>
   </author>

   <chapters>
      <preface num = "1" pages = "2">Welcome</preface>
      <chapter num = "1" pages = "4">Easy XML</chapter>
      <chapter num = "2" pages = "2">XML Elements?</chapter>
      <appendix num = "1" pages = "9">Entities</appendix>
   </chapters>

   <media type = "CD"/>
</book>
```

CDATA Sections

- **`CDATA`** sections
  - May contain text, reserved characters and whitespace
    - Reserved characters need not be replaced by entity references
  - Not processed by XML parser
  - Commonly used for scripting code (e.g., JavaScript)
  - Begin with **`<![CDATA[`**
  - Terminate with **`]]>`**

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

<!-- Fig. 5.7 : cdata.xml                    -->
<!-- CDATA section containing C++ code       -->

<book title = "C++ How to Program" edition = "3">

   <sample>
         // C++ comment
         if ( this-&gt;getX() &lt; 5 &amp;&amp; value[ 0 ] != 3 )
            cerr &lt;&lt; this-&gt;displayError();
   </sample>

   <sample>
      <![CDATA[

         // C++ comment
         if ( this->getX() < 5 && value[ 0 ] != 3 )
            cerr << this->displayError();
      ]]>
   </sample>

   C++ How to Program by Deitel &amp; Deitel
</book>
```

Using a CDATA section

D:\Books\2000\xmlhtp1\frame files\ch05examples\cdata.xml - Microsoft Internet Explorer

File Edit View Favorites Tools He

Address D:\Books\2000\xmlhtp1\frame files\ch05examples\cdata.xml

```
<?xml version="1.0" ?>
<!--  Fig. 5.7 : cdata.xml                    -->
<!--  CDATA section containing C++ code       -->
- <book title="C++ How to Program" edition="3">
  <sample>// C++ comment if ( this->getX() < 5 && value[ 0 ] != 3 ) cerr <<
    this->displayError();</sample>
  - <sample>
    - <![CDATA[

                  // C++ comment
                  if ( this->getX() < 5 && value[ 0 ] != 3 )
                     cerr << this->displayError();

    ]]>
  </sample>
  C++ How to Program by Deitel & Deitel
</book>
```

Done My Computer

Is it well-formed?

- A standalone XML document is ***well formed*** if:
  1. It starts with an XML declaration,
  2. It contains a root element that embeds all other elements,
  3. All elements are properly nested,
  4. Elements that contain data have both start and end tag,
  5. Elements that do not contain data and use only a single tag end with "`/>`",
  6. Attribute values are quoted,
  7. The characters "`<`" and "`&`" are only used to start tags and entity references, respectively,
  8. The only entity references which appear are `&`, `<`, `>`, `'`, and `"`
  9. Element and attribute names must be valid XML names

XML in Action - RSS

- RSS (Really Simple Syndication) is an XML application that allows users to “subscribe” to websites.
- Sample uses: Podcasts, Apple iTune Store, news arrival.
- After XHTML, RSS is probably the XML application that web users see most often.

XML in Action

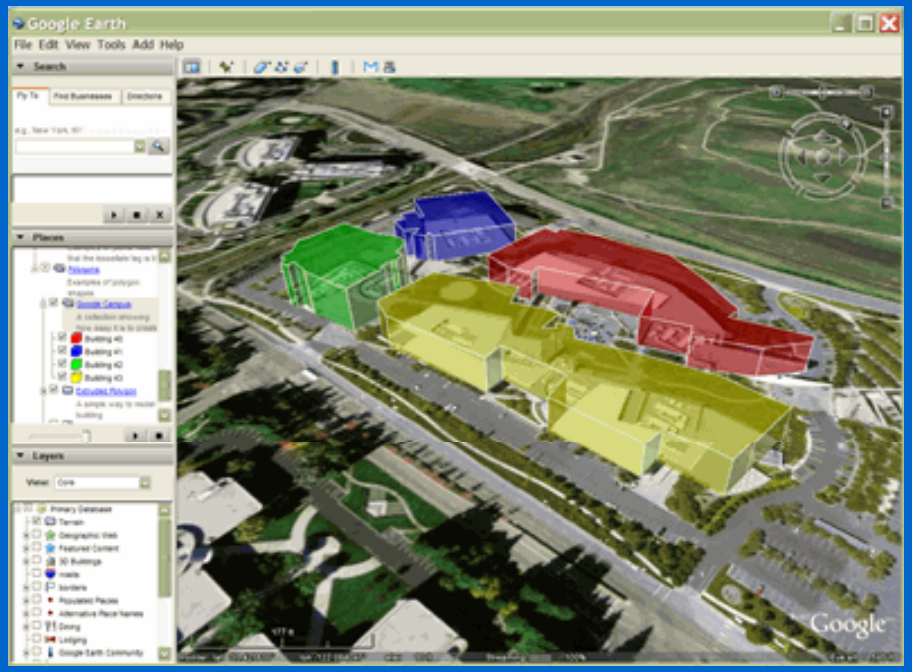
```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">

<channel>
    <title>RSS Example</title>
    <description>This is an example of an RSS feed</description>
    <link>http://www.domain.com/link.htm</link>
    <lastBuildDate>Mon, 28 Aug 2006 11:12:55 -0400 </lastBuildDate>
    <pubDate>Tue, 29 Aug 2006 09:00:00 -0400</pubDate>

    <item>
    <title>Item Example</title>
    <description>This is an example of an Item</description>
    <link>http://www.domain.com/link.htm</link>
    <guid isPermaLink="false"> 1102345</guid>
    <pubDate>Tue, 29 Aug 2006 09:00:00 -0400</pubDate>
    </item>
</channel>
</rss1
```

XML in Action - Google

- KML (Keyhole Markup Language) is a file format used to display geographic data in an Earth browser such as Google Earth, Google Maps, and Google Maps for mobile.



XML in Action - Google

```
<?xml version="1.0" encoding="UTF-8"?>
    <kml xmlns="http://www.opengis.net/kml/2.2">
    <Placemark>
            <name>Simple placemark</name>
            <description>Attached to the ground. Intelligently places itself
                    at the height of the underlying terrain.</description>
            <Point>
            <coordinates>
            122.0822035425683,37.42228990140251,0</coordinates>
            </Point>
    </Placemark>
</kml>
```

Open XML Formats

Default XML file formats for Word, Excel and PowerPoint

Fully 100% compatible with previous formats

Open, transparent format improves interoperability

XML – Transparent, XML format enables new integration scenarios for documents and LOB systems

ZIP container – allows for standard compression on all files without user effort

The Role of XML with Documents

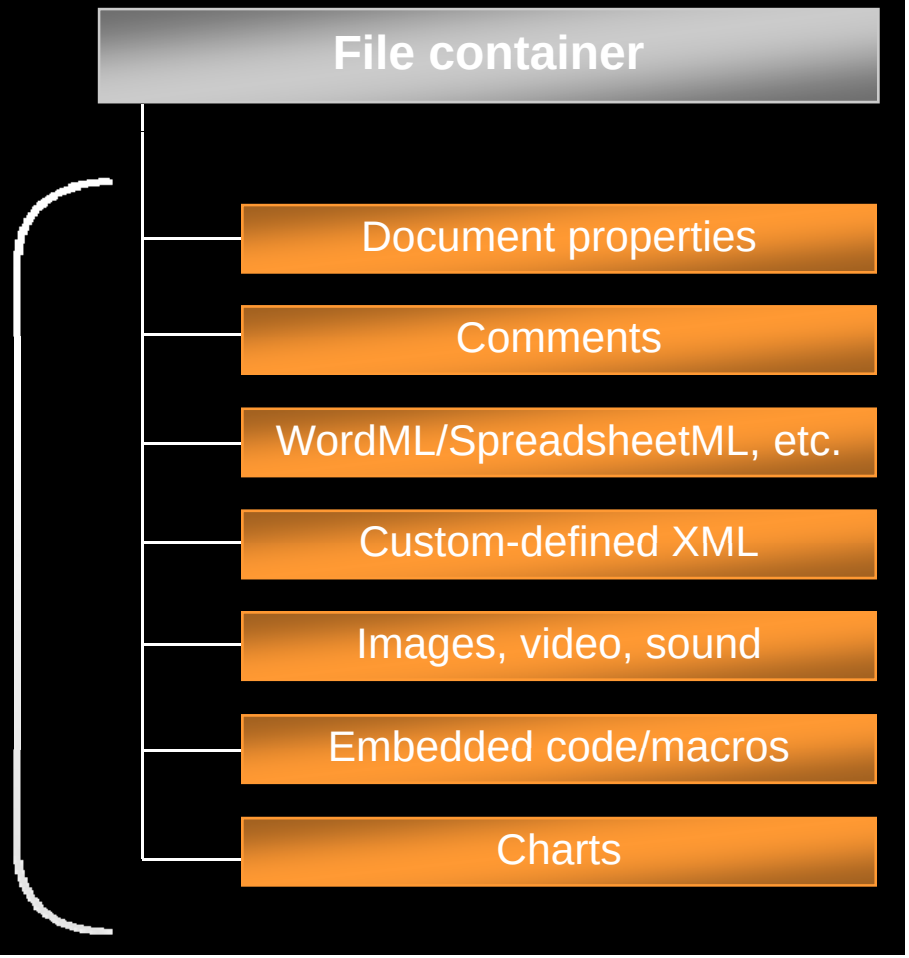
Scenario	Example
Document Assembly Server-based or user-assisted construction of documents from archived content or database content	Create sales reports from financial and forecast data stored in a CRM system
Content Reuse Much easier to move content between documents, including different document types	Leverage content in Feature Specs to create Feature Lists
Content Tagging Add domain-specific metadata to document content to enable custom solutions	Tag presentations using a specific taxonomy to improve knowledge management efficiency
Document Sanitization Remove unwanted content like comments or embedded code from your document when appropriate	Remove all tracked changes and comments from a Word document before it is published

Open XML Formats Architecture

User view:
Single file

Questionnaire.
docx

Developer view:
Modular file



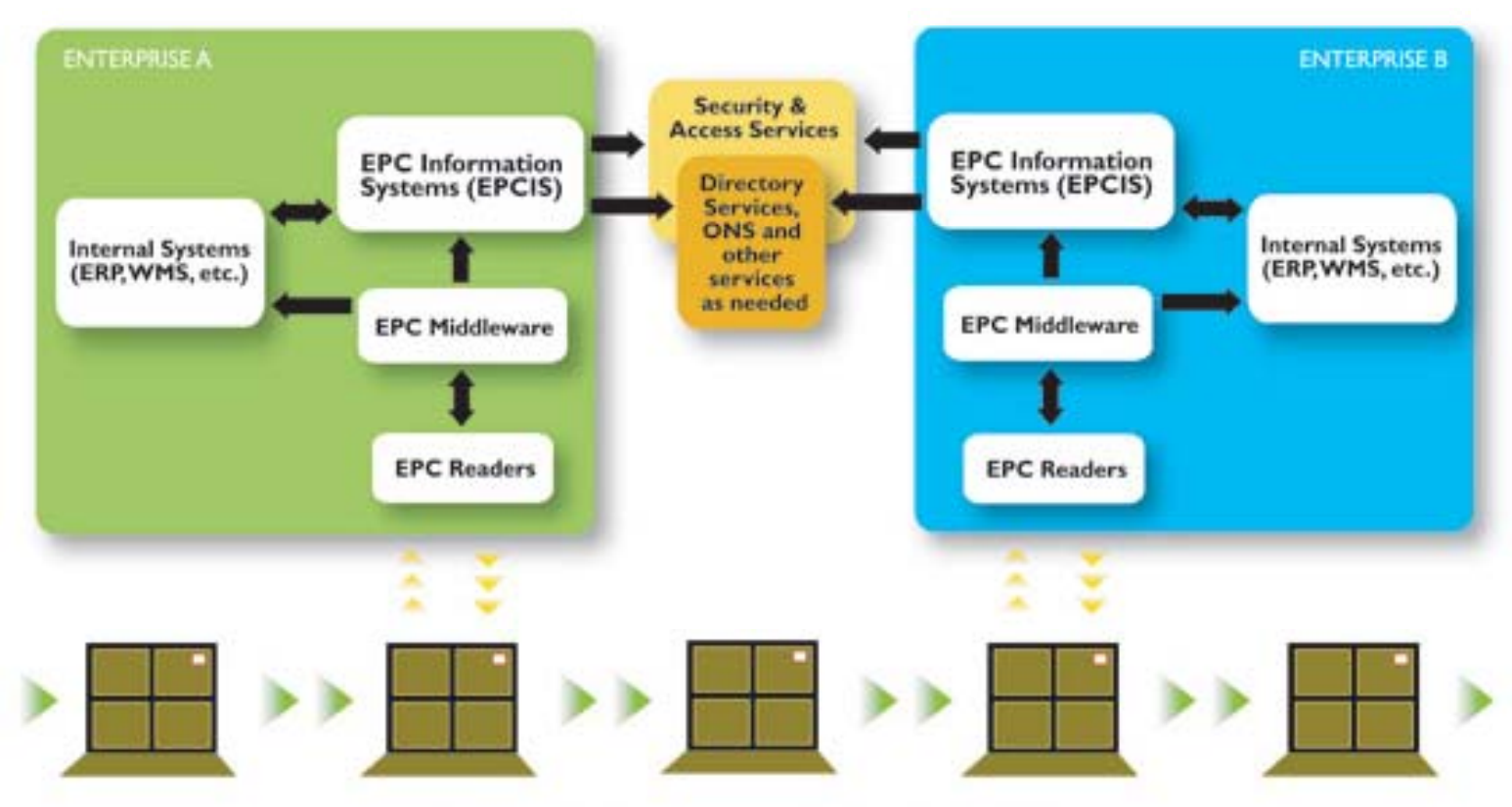
Document Parts

- Most parts are XML
- Each XML part is a discreet, compressed component
- Can add, extract and modify individual parts without using Office programs
- Corruption or absence of any part would not prohibit the file from being opened

RFID/PML

- Electronic product code (EPC): an unique code for each object
  - RFID Tag
  - RFID reader
- Object Name Service(ONS) : each number corresponds with an address in database
- Product Markup Language (PML)
  - In PML server, PML is used to describe and store information about the item.
- Savant: can work as a router, it get EPC information from the RFID reader, send the information to ONS Server and combine with application program for management of the item.

RFID/PML



Tagged Units Moving Through the Supply Chain

RFID/PML

```
<pmlcore: Sensor>
        <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
        <pmlcore:Observation>
                <pmlcore:DateTime>2002-11-06T13:04:34-
                        06:00</pmlcore:DateTime>
                <pmlcore:Tag>
                        <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
                </pmlcore:Tag>
        </pmlcore:Observation>
</pmlcore:Sensor>
```