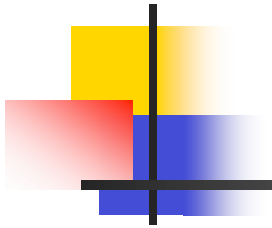
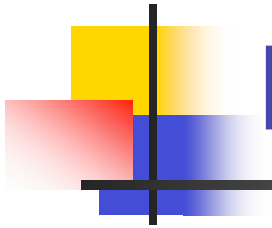


B336 Internet Systems Programming

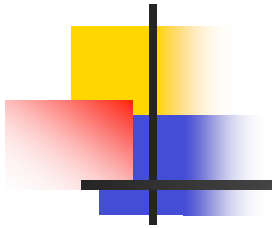
Introduction to XML

(Week 5 Lecture 1)



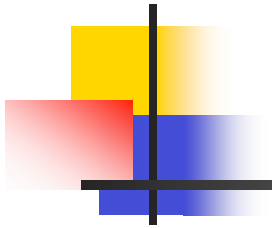
Lecture Objectives

- Understand why there is a need for XML.
- A first look at
  - The components of XML.
  - How XML relate to other technologies.



Lecture Outline

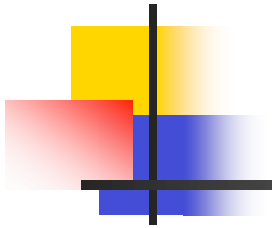
- The concept of markup languages.
- HTML and the need for XML.
- The components of an XML solution.



Foreword

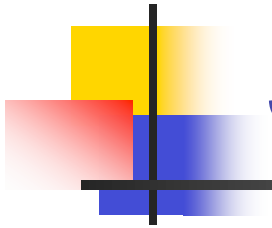
- A simpler version of this introductory lecture was given in B211 in previous years. Those who have done B108 should also have had a gentle introductory lecture to XML as well.

- As a reminder before we get into the gory details of XML, I will give this introductory lecture again.



Mark-up Languages and XML

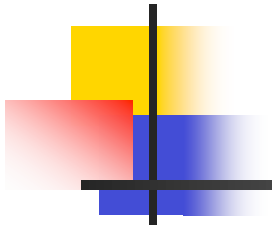
- The use of “mark-up languages” is prevalent among the computing world today
  - HTML is the most visible one of all.
- To understand the point of XML (eXtensible Markup Language), it is essential you have a good understanding of the concept of mark-up languages.
  - XML is basically about creating mark-up languages.



Style, Structure and Content

- Back when all documents were only in printed form, a major consideration for the documents was its “**style**” (or “**presentation**” or “**layout**”)

- The structure of the document was left to the reader of the printed document to work out based on the style.



An Example

- For example, looking at this page, you as a reader can decipher that:
  - The words with the big fonts at the top (“An Example”) is a *title*.
  - The words on the right of each large dot is a *main point*.
  - The words on the right of each smaller dot is a *sub-point* to the main points.
  - The characters on the bottom right is a *page number*.
  - etc.

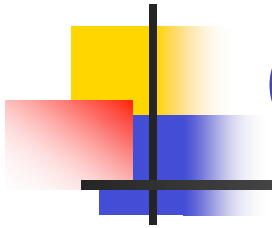
Example “Style” Instructions in Documents

- An example Rich Text Format (RTF) paragraph:

```
\keepn\par\b0
The complete, absolute, utterly useless sentence.
```

\keepn means keep this paragraph with the next.
\par means start a new paragraph here
\b0 means to turn off bolding the text.

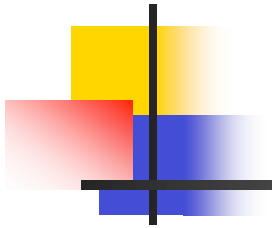
Note: RTF is a format created by Microsoft to ease document exchanges between platforms.



Can a program do the same?

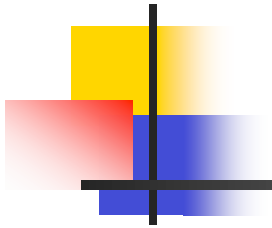
- But we have moved way beyond that basic “let the reader do it themselves” approach to information.
  - The key to the ability to have vast amounts of information being passed around in a network environment is to have the information processed and dealt with by automated programs.
- If a program receives a document containing only style instructions, can it work out the structure of the information?

\par\font=28\bold
For example
\no-bold\newline\par\font=24
\bullet
For example, looking at this page, you as a reader can decipher that:
\newline\indent
\bullet
The words with the big fonts at the top (“An Example”) is a
\italic
title.
\no-italic\newline\indent
The words on the right of each large dot is a
\italic
main point.
...



Mark-up Languages

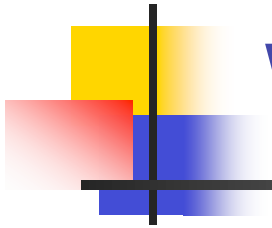
- So the point of mark-up languages is to:
  - Provide information about the structure and content in the documents, so that they can be processed by programs automatically.
  - Separate out the structure and content parts of the documents from the style parts.



Mark-up Languages for Different Purposes

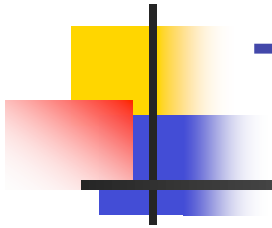
- But different problem domains will have different ways of describing their data.
  - It is impossible to come up with just one set of mark-up tags and expect it to work for all problems.

- So mark-up **meta**-languages like XML do not define a specific set of tags for all purposes - it defines a standard way for defining the tags of a completely new mark-up language.



Why Mark-up Meta-Languages?

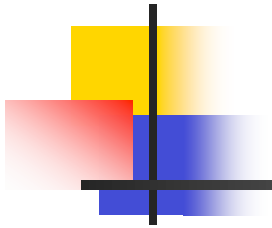
- The key is standardization. By having these meta-languages, we have
  - A **standardized method** and a **consistent set of tools to create new mark-up languages** which is appropriate for a certain problem domain, and to make public these new languages.
  - The ability to **parse** and **process the documents** of these new mark-up languages in **a consistent way** - very important for software development in the problem domain.
- If the meta-language is defined well, it will also mean **an easy way** to create the new languages.



The Beginning of XML

- XML actually began as the Standard Generalized Mark-up Language (SGML), which began as the the Generalized Mark-up Language (GML) at IBM in the 1960's.

- They wanted to created a mark-up meta-language for the purpose as we mentioned.



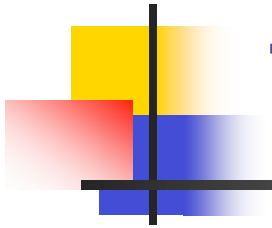
ISO and SGML

- In 1986, the International Standards Organization (ISO) adopted IBM's SGML as an official standard.
  - It provided a very sophisticated and extensive specification for creating, presenting and exchanging documents of any type.
  - Its problem was that it was extremely complex, and too hard for software developers to create tools, and document publishers to use the tools.

HTML and the WWW

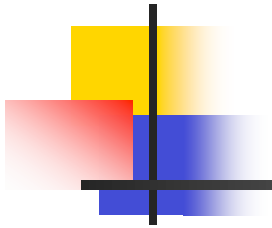
- When the web started catching on in the early 1990's, SGML was considered a perfect vehicle for creating new document types used to exhange information.

- HTML was created loosely following SGML specifications.
  - It provided a simple language to define information users and software developers could develop without having much technical experience.



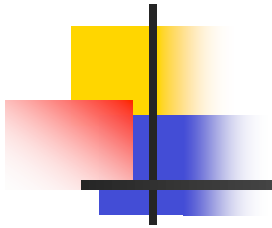
The Simplicity of HTML

- At the beginning of the web, most development in document format concentrated on HTML rather than SGML, since HTML was:
  - simple and easy to learn
  - provided enough mechanisms to do a lot of very interesting things (display formatting, links to other documents, embed graphics and other multimedia, etc)
  - is much more forgiving on errors (like forgetting a closing tag) than SGML.
- As applications became more complex, developers began using HTML beyond what it was originally defined to do.



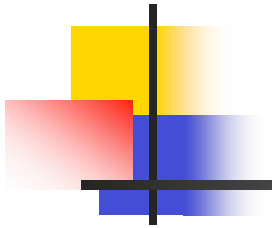
Beyond HTML

- Some example applications which developers were finding hard to implement due to HTML's simplicity:
  - personalization in web communication - to create and pass information specifically catered to a user, group or organization.
    - Technologies such as HTTP cookies, dynamic server-side content, were in response to this challenge.
  - Connecting to Databases - what field should information in a … tag go into?
  - Enterprise Application Integration - an organization's information available from different sources could not be processed by the same applications.



Beyond HTML

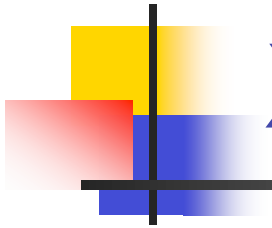
- The main problem with HTML was that it was never designed to be *extensible.*
  - As a software developer, if you want to exchange documents on the web, but find that HTML doesn't do what you want it to do, you cannot easily create and use a new languages that extend HTML, and share this new language with others.



Enter XML

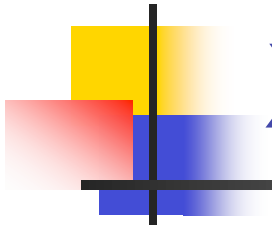
- Standards for XML began to appear under the umbrella of the World-Wide-Web Consortium (W3C) in 1996 to address some of the above problems.

- Their approach was to take SGML, keep the more powerful features, and leave out the more obscure and complex features.
  - Basically, XML is a subset of SGML.



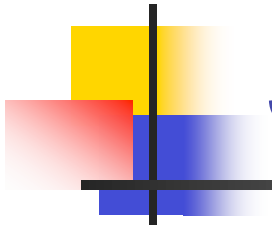
XML Base Recommendations

- The first working draft of the XML standard came out in November 1996.
  - Software for XML began to appear after that.
- The first official base recommendation for XML (version 1.0) by W3C appeared in Feb 1998.
- The current version is XML 2nd Edition, released Oct 2000.
- XML version 1.1 is now in draft form.



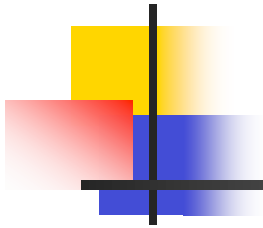
XML Base Recommendations

- Most of the activities in XML is not really in the base recommendation - it is in the technologies built on top of it.



So what is XML?

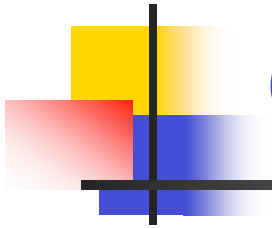
- At the foundations is the XML Base Recommendations, which defines what an XML document must and must not have.
- This forms the basis for creating a new mark-up languages.
  - New languages created following the XML specifications are called **applications** of XML.



Our example on Page 7

- Going back to our example on page 7, to mark-up the page properly, we can have a document like this:

```
<page>
   <title>An Example</title>
   <main_point>For example, looking at this page, you as a
     reader can decipher that:</main_point>
   <sub_point>The words with the big fonts at the top ("An
     Example") is a title.</sub_point>
   <sub_point>The words on the right of each large dot is a
     main point.</sub_point>
   <sub_point> The words on the right of each smaller dot is a
     sub-point to the main points.</sub_point>
   <sub_point> The characters on the bottom right is a page
     number.</sub_point>
   <sub_point>etc.</sub_point>
   <page_number>7</page_number>
</page>
```

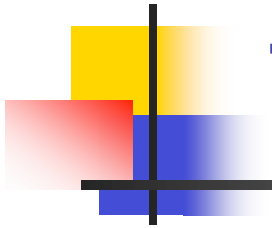


Our example on Page 7

- Compare this to what is on page 10
- Now writing a program to process the different components of the page will be so easy.

A more useful XML document

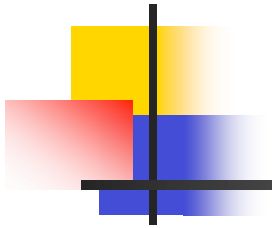
```
<course>
     <name>Bachelor of Science - Internet Computing</name>
     <duration>3 years</duration>
     <unit>
          <title>B336 Internet Systems Programming</title>
          <lecturer>
               <surname language="English">Hiew</surname>
               <othernames language="English">Hong Liang</othernames>
               <email>h.hiew@murdoch.edu.au</email>
          </lecturer>
     </unit>
     <unit>
          <title>B108 Introduction to Multimedia and the Internet</title>
          <lecturer>
               <surname>Rai</surname>
               <othernames>Shri</othernames>
               <email>s.rai@murdoch.edu.au</email>
          </lecturer>
     </unit>
     ...
</course>
```



The DTD

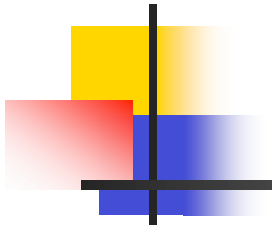
- A Document Type Definition (DTD) defines the structure of an XML document.
  - It can be put at the beginning of a document, or linked to the document.

```
<!DOCTYPE course [
      <!ELEMENT course (name, unit+)>
      <!ELEMENT name (#PCDATA)>
      <!ELEMENT unit (title, lecturer*, tutor*)>
      <!ELEMENT title (#PCDATA)>
      <!ELEMENT lecturer (surname, othernames?, email*)>
      <!ELEMENT tutor (surname, othernames?, email*)>
      <!ELEMENT surname (#PCDATA) >
      <!ATTLIST surname language CDATA "English">
      <!ELEMENT othernames (#PCDATA) >
      <!ATTLIST othernames language CDATA "English">
      <!ELEMENT email (#PCDATA) >
]>
```



Is that all there is to XML?

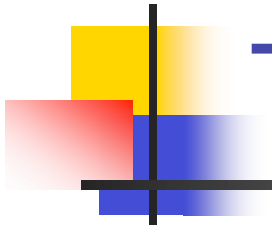
- Also associated with the core XML is a whole set of other technologies. Some examples:
  - Alternate ways of define document types (XML-Schema)
  - Programming language APIs which allow you to easily write software to parse and process XML documents.
  - Specifications on how to represent an XML document at a program level (eg. DOM, SAX).
  - Specifications on how to transform an XML document - to another document type, or to another XML document of the same type (XSL).
  - Specifications on how to define special elements in a document (eg. links in documents using XLink)



Is that all there is to XML?

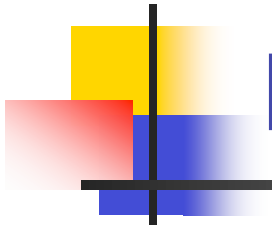
(cont'd)

- Special browsers and general web browsers capable of displaying XML documents
  - But how do general web browsers know what to display when it doesn't know what the tags "mean"? This is a question we will explore as we study the various XML technologies.
- The XML applications themselves (eg. MathML, SMIL, ChemML, etc).



The Design Goals of XML

- As given by W3C:
  - XML shall be straightforwardly usable over the Internet.
  - XML shall support a wide variety of applications.
  - XML shall be compatible with SGML.
  - It shall be easy to write programs which process XML documents.
  - The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
  - XML documents should be human-legible and reasonably clear.
  - The XML design should be prepared quickly.
  - The design of XML shall be formal and concise.
  - XML documents shall be easy to create.
  - Terseness in XML markup is of minimal importance.



Example Popular XML Applications

- Synchronized Multimedia Integration Language (SMIL)
  - Defines audiovisual presentations integrating streaming audio and video with images, text and other media types.
- MathML
  - Mathemeatical expressions, and their presentation for high-quality visual display.
- electronic business XML (ebXML)
  - standardize XML exchange between businesses and organizations.
- XHTML

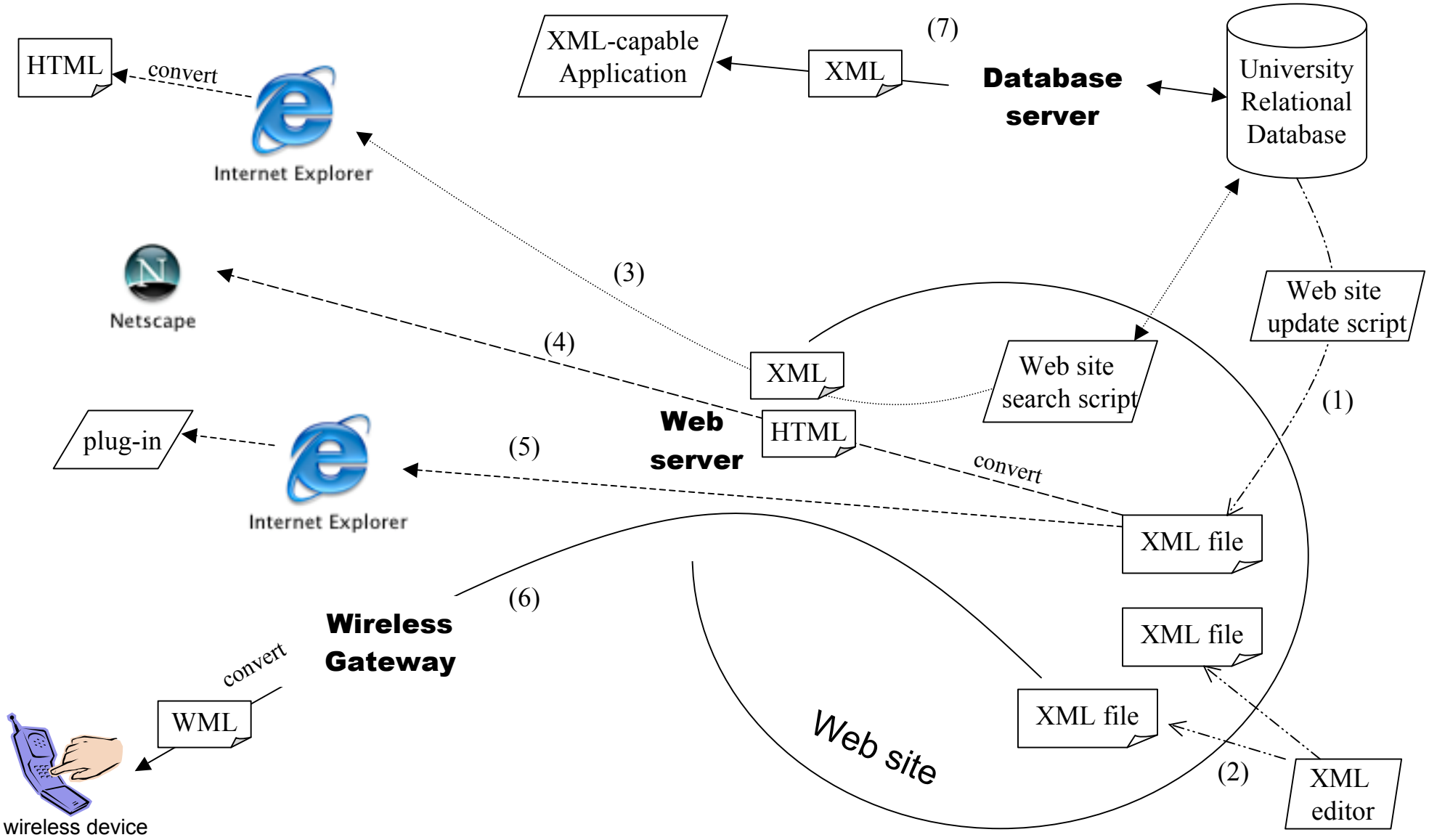
An Example SMIL Document

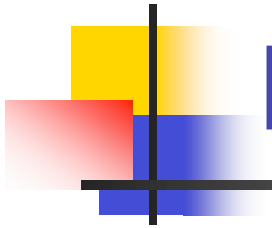
```
<?xml version="1.0">
<smil>
  <head>
    <meta name="title" content="A SMIL Document" />
    <region id="image_region"
       left="0" top="0" height="200" width="200" />
  </head>
  <body>
    <seq>
      <img src="header.gif"
         region="image_region" fill="remove" />
      <par>
        <audio src="track.wav" />
        <img src="picture.gif"
           region="image_region" fill="remove" />
      </par>
    </seq>
  </body>
</smil>
```

Present in sequence

Present in parallel

Examples of how our course XML (from pages 26-27) may be used:





Examples

- From the previous diagram:

(1) A daily update script extracts course information from a relational database and generates XML files to put on a web site.

(2) A data-entry person uses an XML editor to create new XML files with course information.

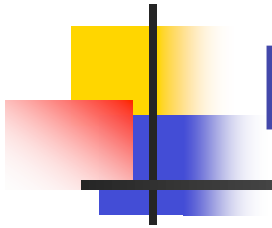
(3) A user surfing the web site uses the search facility to find some course information. The search script queries the database, gets back the results, and sends the results back to the browser in XML. Browser converts the XML to appropriate HTML and displays.

(4) A user using a web browser accesses some course information. The web server converts the XML document to HTML and sends HTML back to the browser. Browser displays HTML as normal.

(5) A user using a web browser with a special course-info plug-in accesses some course information. The web server sends the XML document directly to the browser. The browser uses the plug-in to display the course information.

(6) A user on a wireless device accesses some course information. The XML document is sent to the user's wireless gateway, converted to WML and then sent to the wireless device. Device displays WML as normal.

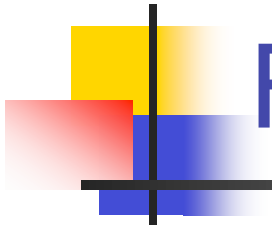
(7) An XML-enabled application connects to the database and retrieves some course information. The database server sends results back in XML.



Endless Possibilities

- The previous diagram only shows a very small fraction of what is possible with XML software.

- The possibilities for which software does what, when, where and how, are endless.



For further information...

- Follow W3C's XML activities:
  - http://www.w3.org/
  - http://www.w3.org/XML/