

B336 Advanced Internet Computing

Introduction to the eXtensible Mark-up Language (XML)

Learning Objectives

- Understand why there is a need for XML.
- Understand how XML relates to other technologies.
- Understand what the components of XML are.
- Know the outline of what we will be studying for the next five weeks in this unit.

Lecture Outline

- The concept of markup languages
- The history that has led to XML as we see it today.

Mark-up Languages

- It is essential you have a good understanding of the concept of mark-up languages to understand the point of XML.
- XML is basically about creating mark-up languages.

Style, Structure and Content

- Back when all documents were only in printed form, the only consideration for the documents were its **style** (or its **presentation**)
- The structure and content 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 RTF instructions

- An example Rich Text Format (RTF) paragraph:

```
\keepn\par\b0  
I would type something meaningful in this paragraph,  
except I can't think of anything.
```

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

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.
- Can a program do as well as you did in the last page, if it only has the style of the document?

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 exchange 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.
 - So began a whole new generation of HTML writers calling themselves “programmers”

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, they moved beyond what HTML's 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 different cookie formats, dynamic server-side content, were in response to this challenge.
 - Connecting to Databases - what field should information in a `<p>...</p>` tag go into?
 - Enterprise Application Integration - an organization's information available on the web at different locations cannot 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.
- The 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 Specifications

- The first working draft of the XML standard came out in November 1996, and software for XML began to appear after that.
- The official base specification for XML (version 1.0) appeared in Feb 1998, and is still the only version we have so far.
 - But as we shall see in the next few weeks, the core focus for XML technologies is not the base recommendations, but in the associated technologies.
 - XML 1.0 is only the framework that everything is built around.

So what is XML?

- At the basis is the XML 1.0 specification, 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.

An example (partial) XML document

```
<anthology>
  <poem>
    <title>The Sick Rose</title>
    <author>William Blake</author>
    <stanza>
      <line>O Rose thou art sick.</line>
      <line>The invisible worm,</line>
      <line>That flies in the night</line>
      <line>In the howling storm,</line>
    </stanza>
    <stanza>
      <line>Has found out thy bed</line>
      <line>Of crimson joy:</line>
      <line>And his dark secret love</line>
      <line>Does thy life destroy.</line>
    </stanza>
  </poem>
  <poem>
    <title>A Poison Tree</title>
    ...
  </poem>
  ...
</anthology>
```

An example XML DTD

- A Document Type Definition (DTD) defines the structure of an XML document.

```
<!DOCTYPE poetry [  
    <!ELEMENT anthology (poem*)>  
    <!ELEMENT poem (title?, author?, stanza+)>  
    <!ELEMENT title (#PCDATA) >  
    <!ELEMENT author (#PCDATA) >  
    <!ELEMENT stanza (line+) >  
    <!ELEMENT line (#PCDATA) >  
]>
```

Is that all there is to XML?

- Also associated with the core XML is a whole set of other technologies. Some examples:
 - 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.
 - Specifications on how to transform an XML document - to another document type, or another XML document of the same type.
 - Specifications on how to define special elements in a document (eg. links in documents using XLink)
 - Web Browsers which are able to display any XML documents
 - » But at best you can only expect it to display the structure of the tags - what else can it do if it doesn't know the meaning of the tags.
 - The XML applications themselves.

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.

To Create an XML Solution

- To create an XML solution, we need to:
 - Define what the format of the data, and what tags we are going to use to represent the data in our documents.
 - Create the software required to process the documents appropriately.
 - Create the display tools we will use to display the documents
 - » In the end, humans have to see some parts of the information at some stage.
 - Decide and implement solutions on integrating with existing legacy software.
 - etc.
- More on this once we have suffered through the gory details of XML in the next couple of weeks.

In upcoming lectures...

- The precise syntax of an XML documents.
- How to define the syntax of a new language using DTD.
- Converting to and from XML documents using XSLT
- Perl APIs available to parse and process XML documents
- Example real world XML applications and technologies.

In upcoming lectures...

- The most important thing you should be keeping in mind (as with all other technologies):
 - “How can I use all these technical details to help me in finding a solution to problems?”