CM0133 Internet Computing

XML
The eXtensible Markup Language 2

Overview

- XSLT
  - Stylesheets
  - Cascading stylesheets (CSS)
  - XSL stylesheets
- XML namespaces
- XHTML
- XML schema
- Parsing XML

The Extensible Stylesheet Language

- A cascading stylesheet creates a style for specific XML elements
- An XSL stylesheet creates a template – this is a design for (part of) the page
- The template is used to format XML elements which match a specified pattern
- XSL can be used to produce any type of markup
  - HTML, LaTeX, PDF, Rich Text Format
- XSL stylesheets are included using the following line:

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

XSL

- The following line declares that the file is a stylesheet, and creates a namespace

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

- The following line declares an XSL template

```
<xsl:template match="/">
```

- A stylesheet can contain multiple templates for use in different situations. This example defines a single template (which is applied to the whole document) using the pattern matching command `match`
- Any element matching the pattern will be subject to the transformations it includes

Example

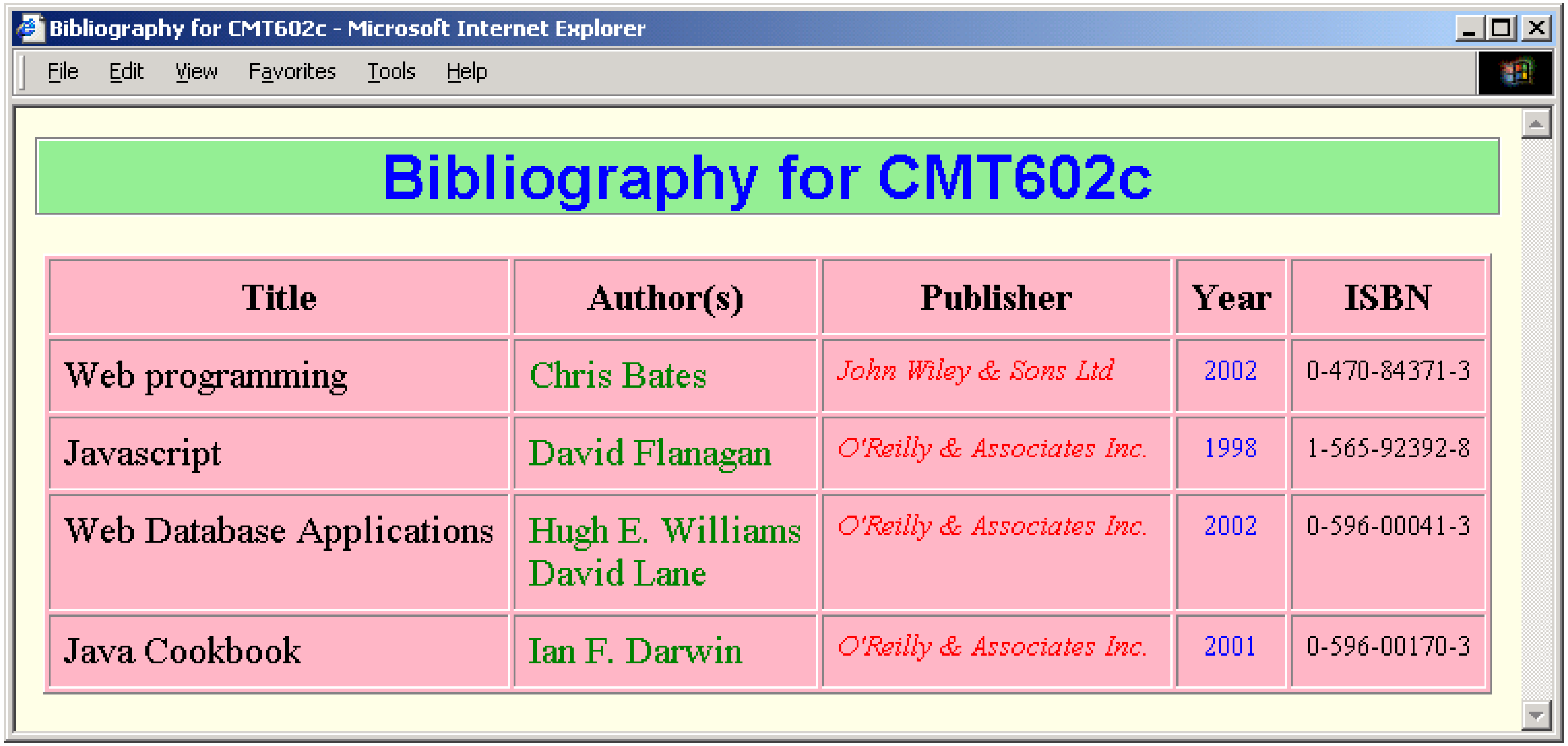
```
<html>
<body bgcolor="lightyellow">
  <h1><!-- put bibliography title here --></h1>
  <table border="1">

   <!-- for every book -->
   <tr>
     <td><!-- put title here --></td>
     <td><!-- put authors here --></td>
     <td><!-- put publisher here --></td>
     <td><!-- put year here --></td>
     <td><!-- put ISBN here --></td>
   </tr>
  </table>
 </body>
</html>
```

- First write a framework for the desired output (using comments)

Example

- Using the attributes of the HTML elements (including style attributes) we can produce more complex presentations


Bibliography for CMT602c - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Bibliography for CMT602c

Title	Author(s)	Publisher	Year	ISBN
Web programming	Chris Bates	John Wiley & Sons Ltd	2002	0-470-84371-3
Javascript	David Flanagan	O'Reilly & Associates Inc.	1998	1-565-92392-8
Web Database Applications	Hugh E. Williams David Lane	O'Reilly & Associates Inc.	2002	0-596-00041-3
Java Cookbook	Ian F. Darwin	O'Reilly & Associates Inc.	2001	0-596-00170-3

XSL

- The XML document is represented as a hierarchy of patterns (each separated by a forward slash)
- The following line iterates over all books

```
<xsl:for-each select="bibliography/book">
```

- The following line extracts the value of the book title

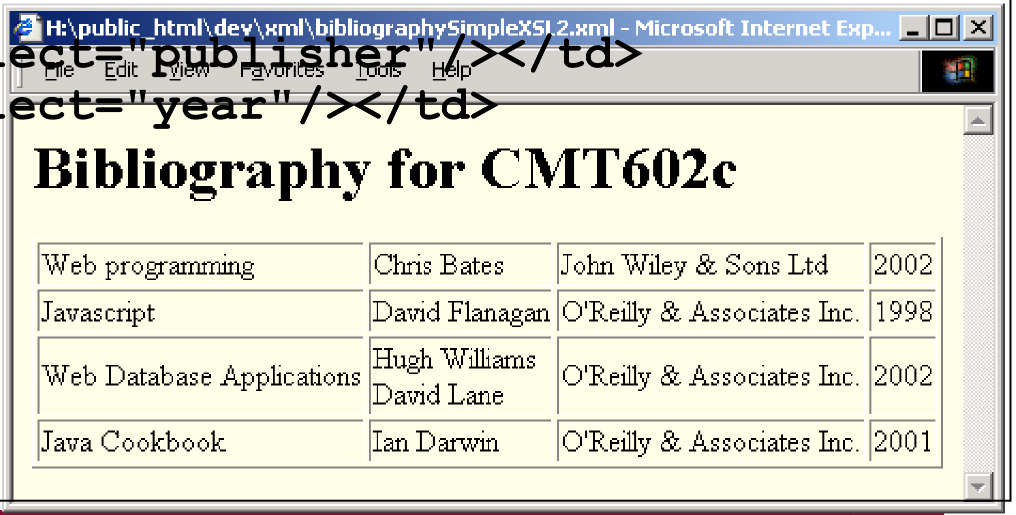
```
<xsl:value-of select="title"/>
```

- The tag is substituted in the output by the value
- The following line extracts the name attribute of the bibliography

```
<xsl:value-of select="bibliography/@name"/>
```

Example

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">
  <html>
   <body bgcolor="lightyellow">
    <h1><xsl:value-of select="bibliography/@name"/></h1>
     <table border="1">
      <xsl:for-each select="bibliography/book">
       <tr>
        <td><xsl:value-of select="title"/></td>
        <td>
          <xsl:for-each select="author">
            <xsl:value-of select="firstname"/>
            <xsl:value-of select="lastname"/><br/>
           </xsl:for-each>
        </td>
        <td><xsl:value-of select="publisher"/></td>
        <td><xsl:value-of select="year"/></td>
       </tr>
      </xsl:for-each>
     </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```


H:\public_html\dev\xml\bibliographySimpleXSL2.xml - Microsoft Internet Exp...
File Edit View Favorites Tools Help

Bibliography for CMT602c

Web programming	Chris Bates	John Wiley & Sons Ltd	2002
Javascript	David Flanagan	O'Reilly & Associates Inc.	1998
Web Database Applications	Hugh Williams David Lane	O'Reilly & Associates Inc.	2002
Java Cookbook	Ian Darwin	O'Reilly & Associates Inc.	2001

Transformation nodes

- extracts the value of a selected node
- Loops through elements
- Sorting
- Condition (comparison operators) =, !=, >, <
- …
- ...

- <xsl:value-of>
- <xsl:for-each>
- <xsl:sort>
- <xsl:if>
- <xsl:choose>
- <xsl:apply-templates>

XSL

- More Than a Style Sheet Language
- Consists of three parts:
  - XSLT - a language for transforming XML documents
  - XPath - a language for navigating in XML documents
  - XSL-FO - a language for formatting XML documents

XSLT

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation

In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

XPath

- XPath is used to navigate through elements and attributes in an XML document.
- XPath is a major element in W3C's XSLT standard - and XQuery and XPointer are both built on XPath expressions.
- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions

Browser Support

- Mozilla Firefox supports XML, XSLT, and XPath from version 3.
- Internet Explorer supports XML, XSLT, and XPath from version 6.
- Google Chrome supports XML, XSLT, and XPath from version 1.
- Opera supports XML, XSLT, and XPath from version 9. Opera 8 supports only XML + CSS.
- Apple Safari supports XML and XSLT from version 3.

XML namespaces

- Since XML allows authors to create their own tags, naming collisions can occur
- Namespaces distinguish between elements and attributes of the same name
- Namespaces are declared by an **`xmlns`** attribute whose value is the URI of the namespace
  - The document referred to by the URI need not exist
- Prefixes are attached to all element and attribute names that belong to the associated namespace

XML namespaces

- A document has a **`file`** element representing text files

```
<file filename="ccList02.txt">
  <desc>Christmas Card List 2002</desc>
</file>
```

- Another document has a **`file`** element representing image files

```
<file filename="catPic.jpg">
  <desc>A picture of the cat</desc>
  <size width="200" height="100"/>
</file>
```

- We would like to define an **`index`** element containing **`file`** elements of both types.

XML namespaces

- We can also define a default namespace

```
<index
 xmlns="http://www.cs.cf.ac.uk/d.evans/xmlns-text
 xmlns:image="http://www.cs.cf.ac.uk/d.evans/xmlns-image>

  <file filename="ccList02.txt">
    <desc>Christmas Card List 2002</desc>
  </file>

  <image:file filename="catPic.jpg">
    <image:desc>A picture of the cat</image:desc>
    <image:size width="200" height="100"/>
  </image:file>

</index>
```

XML namespaces

- This can be done using namespaces

```
<index
 xmlns:text="http://www.cs.cf.ac.uk/d.evans/xmlns-text
 xmlns:image="http://www.cs.cf.ac.uk/d.evans/xmlns-image>

  <text:file filename="ccList02.txt">
    <text:desc>Christmas Card List 2002</text:desc>
  </text:file>

  <image:file filename="catPic.jpg">
    <image:desc>A picture of the cat</image:desc>
    <image:size width="200" height="100"/>
  </image:file>

</index>
```

XHTML

- XHTML 1.0 reformulated HTML 4 as an XML 1.0 application, and provided three DTDs corresponding to the ones defined by HTML 4.
- XHTML 1.0 was formally released in January 2000
  - A new standard which all web authors should use
- XHTML expresses HTML as an XML application
- All XHTML documents must be capable of
  - being generated by XML editors
  - being parsed by XML parsers
- XHTML has two main aspects
  - control data
  - markup tags

Control data

- All XHTML documents should start with an XML declaration

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

- All XHTML documents must have a Document Type Declaration (DTD)

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "DTD/xhtml1-transitional.dtd">
```

Document Type Declarations

There are three DTDs to choose from

- `transitional`
  - used in pages that include presentational markup
  - e.g `<font>` tags
- `strict`
  - used when full compliance with the standard is required
  - all presentation is done using cascading stylesheets
- `frameset`
  - allows the screen to be partitioned into separate frames

The expanded HTML tag

- The top-level node of an XHTML document must be an **`html`** node
- In HTML 4.0 the **`<html>`** tag holds information regarding formatting and events
- In XHTML 1.0 the **`<html>`** tag holds information regarding the document itself

```
<html xmlns="http:www.w3c.org/1999/xhtml" xml:lang="en">
```

- The **`html`** node declares the namespace for the document, and also the language of the document

Rules for XHTML tags

- Nested tags must be terminated in the reverse order in which they were declared
- All XHTML tags must be in lower case
- All tags which may have content must have end tags (`<p>Some text</p>`)
- Empty elements must either have end tags (`<hr></hr>`) or be terminated properly (`<hr />`)
- All attribute values must be placed inside quotes

Rules for XHTML tags

```
<script>
  <![CDATA[
    // script goes here
  ]]>
</script>
```

XPath – Built In Functions

- Built In Functions cover
  - string values
  - numeric values
  - date and time comparison
  - node and QName manipulation
  - sequence manipulation
  - Boolean values
  - and more ...

XML schema

- Document Type Definitions are inherited from SGML
  - They cannot be manipulated (e.g. searched, transformed)
  - They are too “document centric” (designed for text documents)
  - XML is designed for any type of structured information
- DTDs describe the structure of XML documents, not the contents of its elements

```
<quantity>5</quantity>
<quantity>hello</quantity>
```

- DTDs cannot confirm that the `quantity` element contains numeric data
  - The application (which uses the XML document) has to do this

XML schema

- XML Schema are an alternative to DTDs
  - They are XML documents (so they can be manipulated)
  - They conform to particular DTDs (they must be valid)
- W3C XML Schema use
  - the file extension `.xsd`
  - the namespace prefix `xsd`
  - the URI `http://www.w3.org/2000/10/XMLSchema`
- The root element `schema` contains the document definition
- The element `element` defines elements
  - The attribute `name` specifies the element name
  - The attribute `type` specifies the element type
  - The attribute `ref` references an existing element

Example

```
<?xml version="1.0">
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema>
 <xsd:element name="bibliography">
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="pages" type="xsd:integer"/>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="book">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="title" type="xsd:string"/>
       <xsd:element name="author" type="xsd:string"/>
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="firstname" type="xsd:string"/>
          <xsd:element name="lastname" type="xsd:string"/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      <xsd:element name="publisher" type="xsd:string"/>
...etc...
```

XML schema

- XML elements are either of simple or complex type
- Simple types define elements that are containers for data
- Complex types define elements that contain other elements and/or have attributes
- Attributes of a complex element must be defined after those of any child elements it may have
- The components of a complex type must form a sequence, which must be declared
- Elements and attributes may have a data type, defined as an attribute of the element/attribute definition

XSD – Validation and Typing

XSD does two things at the same time:

1. Validation checks for structural integrity (is the document schema-valid?)
   - checking elements and attributes for proper usage (as with DTDs)
   - checking element contents and attribute values for proper values
2. Type annotations make the types available to applications
   - instead of having to look at the schema, applications get the Post-Schema Validation Infoset (PSVI)
   - type-based applications (such as XSLT 2.0) can work on the typed instance

http://dret.net/lectures/xml-fall09/xsd-1#(7)

XPath

- XML structures data into a rather small number of different constructs, most notably elements and attributes. The XML Path Language (XPath) defines a way how to select parts of XML documents, so that they can be used for further processing. XPath's primary use in in XSL Transformations (XSLT), but other XML technologies use it as well, e.g. XSD. XPath is a very compact language with a syntax that resembles path expressions well-known from file systems. These path expressions, however, are generalized and therefore much more powerful than the rather simple path expressions in file systems. Because of its use in different XML technologies, XPath is one of the most important XML core technologies.

http://dret.net/lectures/xml-fall09/xpath#(2)

XQuery

- The XML Query (XQuery) language has been designed to query collections of XML documents. It is thus different from XSLT, which primarily transforms one document at a time. However, the core of both languages is XPath 2.0, which means that learning XQuery (and XSLT 2.0) is not very hard when starting with a solid knowledge of XPath 2.0. XQuery's main concept is an expression language which supports iteration and binding of variables to intermediate results. The final result of an XQuery is a tree, which can be serialized in various serialization formats.

http://dret.net/lectures/xml-fall09/xquery-1#(2)

Parsing - Definition

- In linguistics, to divide language into small components that can be analyzed. For example, parsing this sentence would involve dividing it into words and phrases and identifying the type of each component (e.g., verb, adjective, or noun)
- For XML, parsing means reading an XML document, identifying the various components, and making it available to an application

http://cdn.cs259.net/2007/fall/lectures/2/lecture2.pdf

Parsing – Grammars in Backus-Naur Form

- In order to parse a document, you need to be able to specify exactly what it contains
- XML specification does this for XML using a grammar in Backus-Naur Form (BNF)
- A grammar describes a language through a series of rules
  - A rule describes how to produce a something
  - (e.g., a start tag) by assembling characters and other
  - non-terminal symbols
  - Made up of
    - non-terminal symbols
    - terminal symbols (data that is taken literally)

http://cdn.cs259.net/2007/fall/lectures/2/lecture2.pdf

Event Based XML Parsing

- data-centric view of XML
- When an element is encountered, the idea is to process it and then forget about it.
- Event-based parsers return the element, its list of attributes, and the content.
- More efficient for a number of applications especially searches.
  - requires less code
  - less memory since
  - not all elements are available simultaneously
- XMLReader, SAX

DOM Based XML Parsing

- DOM parsers are tree-based parsers
- provide a document-centric view of XML
- In-memory tree is created for the entire document
- Once the entire XML document is parsed all elements and attributes are available at once,
- Useful if you need to navigate around the document and perhaps change various document chunks
  - Requires more coding
  - memory-intensive for large documents
- DOM Parsers, SimpleXML

Parsing XML with PHP

- Examples are on the web
  - http://www.ibm.com/developerworks/xml/library/x-xmlphp2.html
  - http://www.kirupa.com/web/xml_php_parse_beginner.htm
  - http://www.w3schools.com/php/php_xml_parser_expat.asp
- Depending on your application you need to decide whether you want
  - An Event Based Parser
  - A DOM Based Parser

PHP – Parsing XML

- For an event based parser the following steps are necessary:
  - Create the parser
  - Set the start and end tag handlers
  - Set the data handler
  - Open the XML file
  - Read the XML file
  - Parse the XML data
  - Destroy the parser
  - Close the XML file

PHP - Parsing XML 1

Creating the parser:

```
$xml_parser = xml_parser_create();
```

Setting the start tag, end tag, and data handlers:

```
xml_set_element_handler($xml_parser, "startTag", "endTag");
xml_set_character_data_handler($xml_parser, "contents");
```

- Don't forget to free the parser memory when you finished parsing
- Close the XML file
- Always escape illegal XML characters < , > , & , ' , "

PHP - Parsing XML 2

```
$dom = new DomDocument();
$dom->prevservWhiteSpace = false;

if (!@$dom->load($file)) {
    echo "example.xml doesn't exist!\n";
    return;
}

$titleList = $dom->getElementsByTagName('title');
$titleCnt  = $titleList->length;

print "<ul>";

for ($idx = 0; $idx < $titleCnt; $idx++) {
    print "<li>" . $titleList->item($idx)->nodeValue . "\n</li>" ;
}
```

XML Editors

- Today XML is an important technology, and development projects use XML-based technologies like:
  - XML Schema to define XML structures and data types
  - XSLT to transform XML data
  - SOAP to exchange XML data between applications
  - WSDL to describe web services
  - RDF to describe web resources
  - XPath and XQuery to access XML data
  - SMIL to define graphics
- To be able to write error-free XML documents, you will need an intelligent XML editor!

Literature

- Comprehensive XML course: http://dret.net/lectures/xml-fall09/
- W3C documents
- W3 schools

- Examples presented in the lecture are available as download: http://users.cs.cf.ac.uk/F.A.Twaroch/materials/php_XML_parser.zip