

Comp 336/436 - Markup Languages

Fall Semester 2019 - Week 6

Dr Nick Hayward

XML - validation - DTD - add to XML

- a document type declaration
 - *added after the XML declaration*
 - *a mechanism for naming the document type for compliance*
 - *and for including its definition...*
- valid XML documents must declare the document type
 - *editors, browsers &c. can read **DTD***
 - *helps define the template structure...*
- a document type declaration names the document type
 - *references the root element of the document*
- it can reference an external DTD
 - *the external DTD subset*
- include the DTD internally
 - *in the internal DTD subset*
- or use both...
- document type declarations use the following form, e.g.

```
<!DOCTYPE NAME SYSTEM "file">
```

- e.g. reference an external DTD

```
<!DOCTYPE authors SYSTEM "authors.dtd">
```

XML - validation - DTD - issues

- DTD issues for XML include:
 - *DTDs do not make use of XML syntax*
 - *DTDs have no constraints on character data*
 - *e.g. if character data is allowed...any character data allowed*
 - *poor support for schema evolution, extension, or inheritance of declarations*
 - *DTDs provide simplistic attribute value models*
 - *DTDs provide a simple ID attribute mechanism*
 - *DTDs allow only default values for attributes, not for elements*

XML - validation - schema - intro

- DTDs inherited by XML from its predecessor SGML
- DTDs helped ease the transition from SGML to XML
- XML Schema became a W3C recommendation in 2004
 - *provides a rich and flexible mechanism for defining XML vocabularies*
- **Schemas** are written using XML syntax
- **Schemas** are referenced as external documents
- an XSL Schema (**XSD**) is itself an XML document

XML - validation - schema - a few benefits

- XML Schemas are now the successors to DTDs.
- a number of benefits to schemas, e.g.
 - *written in XML*
 - *support namespaces and data types*
 - *richer and more powerful than DTDs*
 - *extensible to future additions*

XML - validation - schema

- an XML schema defines elements and attributes allowed in a document, e.g.
 - *child elements*
 - *the order of child elements*
 - *the number of child elements*
 - *whether an element is empty or can include text*
 - *data types for elements and attributes*
 - *default and fixed values for elements and attributes*

XML - validation - schema - basics

- XML Schema starts with the document declaration
 - *continues by opening the root element <schema>*
 - *and by defining the specific namespace*
- within root element all specifications are defined
- schema ends with a closed root element </schema>
 - *as a well-formed XML document*
- XSD file - simple text file with .xsd
 - *basic outline, e.g.*

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  ...
</xsd:schema>
```

XML - validation - schema - body

- body of the schema contains element declarations
- four main schema elements
 - *xsd:element* declares an element and assigns it a type
 - *xsd:attribute* declares an attribute and assigns it a type
 - *xsd:complexType* defines a new complex type
 - *xsd:simpleType* defines a new simple type
- XML Schema provides a set of 19 primitive data types
 - e.g. *boolean*, *string*, *decimal*, *date*, *time*...
 - use directly in an element or attribute definition, e.g.

```
<xsd:element name="name" type="xsd:string" />
<xsd:attribute name="age" type="xsd:integer" />
```


XML - validation - schema - declarations

- `xsd:complexType` and `xsd:simpleType` are used to define new types
- simple declarations define elements
 - *do not have any children or attributes*
 - *may only contain text*
- complex declarations describe elements
 - *may have children and attributes*
 - *may contain text*
- declarations are not actually types
 - *they create an association between a name and defined constraints*
 - *constraints may govern appearance of a name*
 - *applied in documents governed by associated schema*

XML - validation - schema - example book element

- XSD example defining an element
 - *book* of a user-defined complex type *bookType*
- sub-elements in *bookType* definition
 - a simple element of type string or a number (*gYear*)
 - *gYear* = one calendar year, e.g. 1997
- element `<xsd:sequence>` identifies a sequence of elements

```
<xsd:element name="book" type="bookType"/>
<xsd:complexType name="bookType">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="author" type="xsd:string" />
    <xsd:element name="year" type="xsd:gYear" />
  </xsd:sequence>
</xsd:complexType>
```

- XSD uses attributes `minOccurs` and `maxOccurs` to define cardinalities
- this example defines the element *title*
 - *may occur only one time within the element book*

XML - validation - schema - types

- XSD documents allow us to derive new simple types from existing types
 - *by using the `xsd:simpleType` element*
 - *basically defining a subtype*
- different type of elements can be used to define the subtype
- `xsd:restriction` child element
 - *derives a type - by restricting legal values of base type*
- `xsd:list` child element
 - *derives a type - a white space separated list of base type instances*
- `xsd:union` child element
 - *derives a type - by combining legal values from multiple base types*

XML - validation - schema - types example

```
<xsd:simpleType name="vehicle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="car"/>
    <xsd:enumeration value="motorbike"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="racer" type="vehicle" />
```

- a new simple type, vehicle
 - *defined as enumeration of possible string values*
 - *e.g. car and motorbike*
- and element racer defined of type vehicle

XML - transforming & rendering - intro

- XSL is a family of recommendations
 - *used for defining XML document transformation and presentation*
- XSL engine uses these stylesheets to transform XML documents into other documents
 - *formats output according to specific formatting templates*
- XSL family consists of three main sub-languages
 - *XSLT (XSL Transformations)*
 - XML-based language for transforming XML documents into other XML documents
 - e.g. XML, XHTML...
 - *XPath (XML Path Language)*
 - expression language used by XSLT
 - access or refer to parts of an XML document
 - *XSL-FO (XSL Formatting Objects)*
 - XML vocabulary for specifying formatting semantics
 - used for print publications, e.g. XML to PDF...
- in XSL
 - *input document is often called the source tree*
 - *output document the result tree*

XML - transforming & rendering - XSLT & XPath

- XSLT is a powerful language for transforming XML documents, e.g.
 - *a HTML document*
 - *another XML document,*
 - *a Portable Document Format (PDF) file*
 - *a Scalable Vector Graphics (SVG) file*
 - *a flat text file*
 - ...
- XSLT stylesheet defines the rules for transforming an XML document
 - *chosen XSLT processor does the work and produces the output*
- XSLT relies on a technology called XPath
- XPath helps XSLT to identify nodes in XML documents
 - *e.g. elements, attributes, and other objects...*
- XPath also provides various functions for performing calculations...

XML - transforming & rendering - XSLT Browser Support

- Mozilla Firefox
 - *Firefox supports XML, XSLT, and XPath from version 3*
- Internet Explorer
 - *Internet Explorer supports XML, XSLT, and XPath from version 6*
 - *Internet Explorer 5 is NOT compatible with the official W3C XSL Recommendation*
- Google Chrome
 - *Chrome supports XML, XSLT, and XPath from version 1*
- Opera
- Opera supports XML, XSLT, and XPath from version 9. Opera 8 supports only XML + CSS
- Apple Safari
 - *Safari supports XML and XSLT from version 3*

XML - transforming & rendering - XSLT Process Outline

- select the XML document you want to transform into XHTML
- create an XSL style sheet with a transformation template
- link the XSL style sheet to the XML document
- XSLT compliant browser will transform XML into XHTML

e.g. **Agatha Christie - XML - part I**

- XSLT outputs
 - *document heading*
 - *book title*
 - *book author*

XML - XSLT working example - Agatha Christie

XML file - part I

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="christie-full.xsl"?>
<catalogue>
  <book>
    <title>Evil Under the Sun</title>
    <author>Agatha Christie</author>
    <country>UK</country>
    <publisher>Collins Crime Club</publisher>
    <price>7</price>
    <year>1941</year>
  </book>
</catalogue>
```

XML - XSLT tests - initial XML

Exercise - part I

- create an initial XML document for your own preferred *catalogue* of items
 - e.g. *music albums, toys, books, art, household items, sports, &c.*
- add at least two examples
 - e.g. *two or more books, albums, artworks &c.*
- add *title, author/creator/editor &c.* for each item
- add other details such as
 - *date, value, location, owner &c.*

10 minutes...

XML - transforming & rendering - XSLT basic usage

- XSLT transformation file is an XML document
- it follows the same syntax of any other XML
- using XSLT language, we have to define the appropriate namespace
- example file,

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[... other directives ...]
</xsl:stylesheet>
```

- in the above example,
 - *first line is the XML declaration*
 - *second defines the root element `<xsl:stylesheet>` and attributes*
- stylesheet ends with the root element closing tag
 - *as a well-formed XML document...*

XML - transforming & rendering - XSLT basic usage

- a few other necessary directives for XSLT
- need to intercept the root element and apply a stylesheet template
 - *taking the associated content*
 - *rewriting it as content of the HTML tag <h1>*
 - e.g.

```
<xsl:template match = "/" >
<html><body>
<h1><xsl:value-of select="book" /></h1>
</body></html>
</xsl:template>
```

- define a template and apply it to matched XML

XML - transforming & rendering - XPath basic usage

- XPath language is used to locate the desired element
 - e.g. the expression `"/"` identifies the root element
- once root element has been selected
 - extract content using another XSLT directive, `<xsl:value-of...>`
 - attribute `select` contains another XPath expression
 - extract the content inside the element `book`
- wrap extracted content with HTML elements
 - create basic HTML page

XML - transforming & rendering - XSLT template

- `<template>` is the main element of a XSLT document
 - *may contain many additional directives (elements)*
- a template may be iteratively applied to elements in the XML document
 - *using different tags, e.g. `<xsl:for-each ...>`*
 - e.g.

```
<xsl:template match="chapter">
<xsl:for-each select="paragraph">
<xsl:value-of select=".">
</xsl:for-each>
</xsl:template>
```

- this example applies a template
 - *selects content of each paragraph inside a chapter*

XML - transforming & rendering - XSLT template usage

- one or more sets of rules called templates
- a template contains rules for a specified matched node
- `<xsl:template>` element is used to build templates
- `match` attribute associates a template with an XML element

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

```
<?xml-stylesheet type="text/xsl" href="christie-full.xsl"?>
```

XML - XSLT working example - Agatha Christie

XSLT - part I

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>Collection</h2>
    <table>
      <tr>
        <th>Title</th>
        <th>Author</th>
      </tr>
      <tr>
        <td>.</td>
        <td>.</td>
      </tr>
    </table>
  </body>
</html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part I

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:value-of>`

- `<xsl:value-of>`
 - used to extract the value of an XML element
 - and add to transformation output
 - e.g.

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

- select attribute's value contains an XPath expression
- XPath expression navigates to the given position in the XML
- value of an attribute can be found using an expression such as

```
<xsl:value-of select="catalogue/book/title/@id"/>
```

or

```
<xsl:value-of select="catalogue/book/title/attribute::id"/>
```

XML - XSLT working example - Agatha Christie

XSLT - part 2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>Collection</h2>
    <table>
      <tr>
        <th>Title</th>
        <th>Author</th>
      </tr>
      <tr>
        <td><xsl:value-of select="catalogue/book/title" /></td>
        <td><xsl:value-of select="catalogue/book/author" /></td>
      </tr>
    </table>
  </body>
</html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 2

XML - XSLT tests - initial XML

Exercise - part 2

- create an initial XSL document
 - *output heading for rendered*
 - *get value of the first two elements per item*
 - *e.g. title & author from book*
- test XSL with XML file
 - *open XML with XSL in browser and check rendering...*

10 minutes...

Demos

- XML
 - *Agatha Christie - XML - part 1*
 - *Agatha Christie - XSLT - part 2*

References

- [Oxygen XSLT Processors](#)
- [W3C - XML well formed](#)
- [W3C - XSLT 1.0](#)