Introduction to XML
Day 0, Sunday 9 July,
David Fergusson
Objectives

• To understand basic XML syntax

• To explore the concept of namespaces

• To understand the role of Schema
What is XML

- XML stands for extensible markup language
- It is a hierarchical data description language
- It is a sub set of SGML a general document markup language designed for the American military.
- It is defined by w3c.
How does XML differ from HTML?

- HTML is a presentation markup language – provides no information about content.
- There is only one standard definition of all of the tags used in HTML.
- XML can define both presentation style and give information about content.
- XML relies on custom documents defining the meaning of tags.
What is a Schema?

• A schema is the definition of the meaning of each of the tags within a XML document.

• Analogy: A HTML style sheet can be seen as a limited schema which only specifies the presentational style of HTML which refers to it.

• Example: in HTML the tag `<strong>` pre-defined. In XML you would need to define this in the context of your document.
Pre-existing schema

• A schema can ‘inherit’ from another and extend it.

(analogous to extending a class in JAVA)

• For example the basic tags which allow you to write schema are defined in:
  http://www.w3.org/2001/XMLSchema
A minimal XML document

<?xml version="1.0" ?>
<document name="first">Jim</document>
Valid and well formed

- A correct XML document must be both valid and well formed.

- Well formed means that the syntax must be correct and all tags must close correctly (e.g. `<…> </…>`).

- Valid means that the document must conform to some XML definition (a DTD or Schema).

(Otherwise there can be no definition of what the tags mean)
Namespaces in XML

- Schema require namespaces.

- A namespace is the domain of possible names for an entity within a document.

- Normally a single namespace is defined for a document. In this case fully qualified names are not required.
Common namespace prefixes

xsi  http://www.w3c.org/2000/10/XMLSchema-instance
  namespace governing XMLSchema instances

xsd  http://www.w3c.org/2000/10/XMLSchema
  namespace of schema governing XMLSchema (.xsd) files

tns  by convention this refers to “this” document

wsdl http://schemas.xmlsoap.org/wsdl/
  WSDL namespace

soap http://schema.xmlsoap.org/wsdl/soap/
  WSDL SOAP binding namespace
Using namespaces in XML

- To fully qualify a namespace in XML write the namespace:tag name. eg.
  \begin{verbatim}
  <my_namespace:tag> </my_namespace:tag>
  \end{verbatim}

- In a globally declared single namespace the qualifier may be omitted.

- More than one namespace:
  \begin{verbatim}
  <my_namespace:tag> </my_namespace:tag>
  <your_namespace:tag> </your_namespace:tag>
  can co-exist if correctly qualified.
Namespaces in programming languages

- In C/C++ defined by `#includes` and classes (e.g. `myclass::variable`).

- In PERL defined by package namespace, `$local` and `$my` (e.g. `myPackage::variable`).

- In JAVA defined by includes and package namespace (e.g. `java.lang.Object`)

- **Defines the scope of variables**
Why namespaces in XML?

• A namespace is used to ensure that a tag (variable) has a unique name and can be referred to unambiguously.

• Namespaces protect variables from being inappropriately accessed – encapsulation.

• This makes sure that when you access a variable correctly it has the expected value.
Schema

<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns="document">
<xs:element name = "DOCUMENT">
  <xs:element name="CUSTOMER"> </xs:element>
</xs:element></xs:schema>

<?xml version="1.0"?>
<DOCUMENT xmlns="document"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Xsi:schemaLocation="order.xsd">
  <CUSTOMER>sam smith</CUSTOMER>
  <CUSTOMER>sam smith</CUSTOMER>
</DOCUMENT>

Simple schema saved as order.xsd

XML document derived from schema.
Document Type Definition (DTD)

Simple DTD saved as order.dtd

<?xml version="1.0"?>
<!DOCTYPE DOCUMENT SYSTEM "order.dtd">
<DOCUMENT>
  <CUSTOMER>sam smith</CUSTOMER>
  <CUSTOMER>sam smith</CUSTOMER>
</DOCUMENT>

XML document derived from DTD.
URI vs URL

• This is similar to the distinction between an class and an instance in Object Oriented Programming.

• A URI is a universal resource identifier which could have many forms (i.e., could be an ISBN number if these were in a URN scheme).

• A URL is a http instance of a URI.

• URN (universal resource name) is the declared name of a resource.

• URC {citation} would point to metadata.
Areas of XML Application

• Document Definition

• Data Exchange

• Metadata (Data about Data)

• Remote Procedure Calls
Document Definition

- XML used in particular applications – SGML users
  - Specialised XML Editors

- Word2000 uses XML/HTML hybrid, all OS X applications use XML configuration files.

- Microsoft .NET initiative
  - Documents encoded in XML
  - Information providers expose data in XML
  - More widespread tools (MS Word?)
Using XML for Data Exchange - Current

• Many applications express their data in an intermediate format, to aid interoperability with other applications
  – Other applications parse these documents to reconstitute the data
Using XML for Data Exchange - Future

- XML can help, because its (standard) notation can be analysed by off-the-shelf XML parsers
Using XML as Metadata

- XML metadata provides information about the structure and meaning of any data
- XML metadata can be used to perform more intelligent web searches for goods or information
- Cross-site searches are difficult (depends on metadata info in pages)
- XML metadata is more self-describing and meaningful, for example ...
  - Search for all plays written by William Shakespeare
  - Rather than every web site that mentions him!
Using XML for Remote Procedure Calls

• XML used to exchange data between Software Components

• Simple Object Access Protocol – SOAP
  - A lightweight protocol for exchange of information in a decentralised, distributed environment
  – Web-Sites expose interfaces for interrogation

• Universal Description, Discovery and Integration
  – UDDI
    - Integrating business services
    - ‘Yellow/White Pages’
Support for XML

• Driven by World Wide Consortium (W3C)

• Industry bodies (OASIS, BizTalk)

• Microsoft, Sun, Oracle, IBM, Novell…

• Dell – large implementation of XML

• Inland Revenue - eGIF
Industry perspectives

“I believe both Microsoft and the industry should really bet their future around XML, the standards around XML are key to where we need to go.”

Bill Gates, Microsoft

“XML has the potential to address some of the traditional failings of message standards. Its impact could be considerable.”

Bank of England
Use of XML in biological databases

• EBI Molecular Structure Database (MSD) is an extraction from PDB (Protein Data Bank) which is encoded in XML.

• Uses DTDs

• Initiatives at EBI, NCBI and else where to use XML to make heterogeneous databases interoperable
XML is tree based representation

XML is an acyclic graphical structure - ie. Does not contain loops
Tree-ifying A value Graph

- **Value Node**
  - **Simple** – character data – as can be defined in a Schema
  - **Struct** – outgoing edges distinguished by role name (its accessor)
  - **Array** - outgoing edges distinguished by position (its accessor)
  - Otherwise – by role name and position (its accessor)
  - Every node has a type – explicit or determined by associated schema

- **Serialisation** – to a forest with reference links
  - A node with N incoming edges becomes
    - A top level node
    - N leaf nodes referencing it and having no components
Tree-ifying A value Graph

- No attributes for values; all values as
  - Child elements, for complex types
  - Character data for simple types
- Unqualified names for local;
- Otherwise qualified

```xml
<env:Envelope
  xmlns:env="…/soap/envelope"
  xmlns:m="http://company"
  env:encodingStyle="…encoding/" >
<env:Body>
  <m:Library se:root="1">
    <book>
      <Title>On XML</Title>
      <By href="A1"/>
    </book>
    <book>
      <Title>On WSDL</Title>
      <By href="A1"/>
    </book>
    <m:Author id="A1" se:root="0">
      <Name>Jim</Name>
      <Name>Smith</Name>
    </m:Author>
  </m:Library>
</env:Body>
</env:Envelope>
```

- Use href and id for cross-tree links
- Linked-to value must be top-level body entry
- Link can cross resource boundaries –
  href is full URL
Simple Types

• Every simple value has a type which is a (derivation of a) primitive type, as defined in Schemas standard, which defines their lexical form – (Review)
• Primitive Types
  • string
  • Boolean
  • Float
  • Double
  • Decimal
  • hexBinary
  • base64Binary
  • anyURI
  • QName
  • NOTATION
  • duration
  • dateTime
  • time
  • date
  • gYearMonth
  • gYear
  • gMonthDay
  • gDay
  • gMonth

• Derivations
  – Lengths - length, maxLength, minLength
  – Limits – minInclusive, maxInclusive, minExclusive, maxExclusive
  – Digits – totalDigits, fractionalDigits – (value range and accuracy)
  – pattern – regular expression [A-Z]
  – enumeration – list of allowed values
Compound Types

- If the order is significant, encoding must follow that required order
  - For Schema `sequence` – order is significant
  - For Schema `any` – order is not significant
- Soap encoding schema provides two compound types
  - Se:Struct – components are uniquely named
  - Se:Array – components are identified by position
- Both have href and id attributes
- Arrays have further attributes
Compound Types - Arrays

- Array is of type SEnc:Array or some derivative thereof
  - Attributes SEnc:href SEnc:id for referencing
- Can specify shape and component type

```xml
<element name="A" type="se:Array"/>

<A se:arrayType="xsd:integer [2,3] [2]">
  <A1>
    <n>111</n> <n>112</n> <n>113</n>
    <n>121</n> <n>122</n> <n>123</n>
  </A1>
  <A2>
    <n>211</n> <n>112</n> <n>213</n>
    <n>221</n> <n>122</n> <n>223</n>
  </A2>
</A>
```

- [2] - An array of 2 elements -
- [2,3] Each is a 2 x 3 array of
- Xsd:integer
Partial Arrays

- Partially transmitted array, offset at which it starts

\[
\begin{align*}
\text{<se:Array se:arrayType="xsd:integer [5]" se:offset="[2]" >} \\
\text{<! - - omitted elements 0, 1 and 2-- >} \\
\text{<i>3</i> <i>4</i> }& \\
\text{</>}
\end{align*}
\]

- Sparse Array – each element says its position

\[
\begin{align*}
\text{<se:Array se:arrayType="xsd:integer [ , ] [4]">} \\
\text{<se:Array se:position="[2]"} \\
\text{se:arrayType="xsd:integer[10,10]">} \\
\text{<i se:position="[0,0]">11</i>} \\
\text{<i se:position="[3,8]">49</i>} & \\
\text{</>}
\end{align*}
\]
Typing

- **Type of a value** must be determined, either –
  - Explicitly - as xsi:type attribute for the element itself
  - Collectively - via type of containing compound value
  - Implicitly - by name and schema definition

```
<element name="A" type="se:Array"/>
<xs:complexType name="co-ordinate">
  <xs:all>
    <xs:element name="x" type="xsd:integer">
      <xs:element name="y" type="xsd:decimal">
    </xs:all>
  </xs:complexType>
</element>

<A se:arrayType="xsd:decimal [3]">
  <A1>17.40</A1>
  <A2 xsi:type="integer">17</A2>
  <A3 xsi:type="m:co-ordinate">
    <y>12</y>
    <x>17</x>
  </A3>
</A>
```
Summary

- XML is a language that provides
  - A mark-up specification for creating self descriptive data
  - A platform and application independent data format
  - A way to perform validation on the structure of data
  - A syntax that can be understood by computers and humans
  - The way to advance web applications used for electronic commerce.