An Overview of XML 101

- Introduction
- How is it used
- Requirements
- Elements
- Attributes
- Schema
<?xml version="1.0"?>
<nchelp>
  <pescmeeting>
    <workshop ID="1">
      <title>XML 101</title>
      <presenter>Kristi Blabaum</presenter>
      <date>April 26, 2011</date>
      <location>Boston, MA</location>
    </workshop>
  </pescmeeting>
</nchelp>
An XML Document

<?xml version="1.0"?>  XML Declaration
<pesc>  Root Element
<pescmeeting>  Element
<workshop ID="1">  Attribute
<title>XML 101</title>
<presenter>Kristi Blabaum</presenter>
<date>April 26, 2011</date>
<location>Boston, MA</location>
</workshop>
</pescmeeting>  Close Element
</pesc>
What is XML?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML.
- XML was designed to describe data.
- XML tags are not predefined in XML.
  - You must define your own tags.
- XML is self describing.
- XML uses a DTD (Document Type Definition) to formally describe the data.
- XML Schema is a form of DTD
XML is for structuring data

- Structured data includes things like:
  - spreadsheets, address books, configuration parameters, financial transactions, and technical drawings
- XML is a set of rules for designing text formats that let you structure your data.
- XML is not a programming language, and you don't have to be a programmer to learn it
- XML makes it easier for a computer to generate data, read data, and ensure that the data structure is unambiguous
- XML avoids common pitfalls in language design:
  - it is extensible
  - it is platform-independent
  - and it supports internationalization and localization
XML is text

- Programs that produce spreadsheets, address books, and other structured data often store that data in binary format.
- A text format allows people to look at data without the program that produced it.
- Text formats allow developers to more easily debug applications.
- XML files are text files that people shouldn't have to read, but can when the need arises.
- The rules for XML files are strict. A forgotten tag, or an attribute without quotes makes an XML file unusable.
XML is modular

- XML allows you to define a new document format by combining and reusing other formats.

- Since two formats developed independently may have elements or attributes with the same name, care must be taken when combining those formats.
  - does "<p>" mean "paragraph" from this format or "person" from that one?

- To eliminate name confusion when combining formats, XML provides a namespace mechanism.

- XML Schema is designed to support modularity by defining XML document structures, making it easy to combine two schemas to produce a merged document structure.
XML does not DO anything

- XML is not a programming language
- It was not designed to DO anything
- XML is designed to structure, store, and send information

```xml
<note>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
</note>
```

- This note has a header, and a message body. It also has sender and receiver information. But still, this XML document does not DO anything. It is just pure information wrapped in XML tags. Someone must write a piece of software to send it, receive it or display it.
XML is used to Exchange Data

- With XML, data can be exchanged between incompatible systems.

- In the real world, computer systems and databases contain data in incompatible formats.

- One of the most time-consuming challenges for developers is to exchange data between such systems over the Internet.

- Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.
XML can be used to Share Data

- Since XML data is stored in plain text format, XML provides a software and hardware independent way of sharing data.

- This makes it much easier to create data that different applications can work with.

- It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.
<?xml version="1.0"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>
      <amount>5.95</amount>
      <currency>US Dollars</currency>
    </price>
    <description>two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>
      <amount>7.95</amount>
      <currency>US Dollars</currency>
    </price>
    <description>light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
  </food>
</breakfast_menu>
An example of XML [attribute]

<?xml version="1.0"?>
<breakfast_menu>
<food>
  <name>Belgian Waffles</name>
  <price currency="US Dollars">5.95</price>
  <description>two of our famous Belgian Waffles with plenty of real maple syrup</description>
  <calories>650</calories>
</food>
<food>
  <name>Strawberry Belgian Waffles</name>
  <price currency="US Dollars">7.95</price>
  <description>light Belgian waffles covered with strawberries and whipped cream</description>
  <calories>900</calories>
</food>
</breakfast_menu>
XML documents use a self-describing and simple syntax. 

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
</note>
```

- The first line in the document – the XML declaration – defines the XML version and the character encoding used in the document.

- In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.
An example XML document

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
</note>
```

- The second line describes the root element of the document – describing the document as a note
An example XML document

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
</note>
```

- The next 4 lines describe 4 child elements of the root element (to, from, heading, and body)
An example XML document

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
</note>

• And finally the last line defines the end of the root element
**XML requirements**

- With XML, it is illegal to omit the closing tag

- XML tags are case sensitive
  - With XML, the tag `<Letter>` is different from the tag `<letter>`
  - Opening and closing tags must therefore be written with the same case:
    - `<Message>This is incorrect</message>`
    - `<message>This is correct</message>`

- In XML all elements must be properly nested within each other

- Attribute values must be quoted
XML requirements

- The first tag in an XML document is the root tag.

- All XML documents must contain a single tag pair to define the root element.

- All other elements must be nested within the root element.

- All elements can have sub elements (children). Sub elements must be correctly nested within their parent element:

  `<root>
   <child>
       <subchild>.....</subchild>
   </child>
  </root>`
XML Elements are Extensible

- XML documents can be extended

- Using the following example:
  
  ```xml
  <note>
    <to>Michael</to>
    <from>Kristi</from>
    <heading>Reminder</heading>
    <body>You have a meeting today</body>
  </note>
  ```

- Imagine that we have an application that extracts the `<to>`, `<from>`, and `<body>` elements from the XML document to produce this output:
  
  MESSAGE
  To: Michael
  From: Kristi
  You have a meeting today
XML Elements are Extensible

- Now, Imagine that the author of the XML document added some extra information to it:
  <note>
  <date>04/28/2011</date>
  <to>Michael</to>
  <from>Kristi</from>
  <heading>Reminder</heading>
  <body>You have a meeting today</body>
  </note>

- Should the application break or crash?
  - No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.
XML Elements have Relationships

- Elements are related as parents and children.
- To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described.

```xml
<root>
  <parent>
    <child>…</child>
    <child>…</child>
    <child>…</child>
  </parent>
  <parent>…</parent>
</root>
```
Element Naming

- XML elements must follow these naming rules:
  - Names can contain letters, numbers, and other characters
    `<name001>`
  - Names must not start with a number or punctuation character
    `<001name>  <!name>`
  - Names must not start with the letters "xml"
    `<xml_name>`
  - Names cannot contain spaces
    `<my name>`
  - Any name can be used, but make names descriptive
  - PESC standards have adopted Uppercase Camel (LastName)
Use of Elements vs. Attributes

- Data can be stored in child elements or in attributes.
  
  ```xml
  <person sex="female">
    <firstname>Kristi</firstname>
    <lastname>Blabaum</lastname>
  </person>
  
  <person>
    <sex>female</sex>
    <firstname>Kristi</firstname>
    <lastname>Blabaum</lastname>
  </person>
  
  In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information.
Here are some of the problems of using attributes:
• attributes cannot contain multiple values (child elements can)
• attributes are not easily expandable (for future changes)
• attributes cannot describe structures (child elements can)
• attributes are slightly more difficult to manipulate by program code
An XML Schema defines:

- Elements and attributes that can appear in a document
- Which elements are child elements
- The sequence in which the child elements can appear
- The number of child elements
- Whether an element is empty or can include text
- Data types for elements and attributes
- Default values for elements and attributes
XML Schema & Data Types

- With data type support it is easier to:
  - Describe permissible document content
  - Validate the correctness of data
  - Work with data from a database
  - Define data facets (restrictions on data)
  - Define data patterns (data formats)
  - Convert data between different data types
Because XML Schemas are written in XML:

- An XML editor can be used to edit Schema files
- An XML parser can be used to parse Schema files
- The XML Document Object Model (DOM) can be used to manipulate your Schema
- A Schema can be transformed with XSLT
XML Schemas are Extensible

- With an extensible Schema definition you can:
  - Reuse your Schema in other Schemas
  - Create your own data types derived from standard types
  - Reference multiple schemas from the same document
The End

Kristi Blabaum
kblabaum@glhec.org