XML

Presented by:
Guerreiro João
Thanh Truong Cong
XML: Definitions

- XML = Extensible Markup Language.

- Other Markup Language: HTML.

- XML ≠ HTML
  - XML describes a Markup Language.
  - XML is a Meta-Language.
  - Users can define “tags”.
XML : Browsers

- XML docs can:
  - Exist by themselves.
  - Be embedded in HTML.

- Data island = XML doc within HTML page.
XML : Tags

- <ProductID> 09785 </ProductID>
- <BusStop> Polackasbacken </BusStop>
- <Author> Darrel Ince </Author>
XML : Attributes

- `<Bag colour="black" brand="decathlon"> The bag I take to school </Bag>`

- `<father name="George">`
- `<son name="Mike"> That’s me </son>`
- My father. </father>`
XML : Comments

- Yes you can!
- Begin with “<!--”.
- Finish with “-->”.
- Ex:

  <!-- Shopping List -->
  <Product> Salad </Product>
  <Product> Egg </Product>
  <!-- End of List -->
XML : CDATA sections

- To distinguish a series of characters from tags.

- <example>
- <![CDATA[ <aaa>bb&cc<<]]>
- </example>
XML : Name Conflicts

- Suppose we have:
  
  ```xml
  <table>
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
  </table>
  ```

- But also:
  
  ```xml
  <table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
  </table>
  ```

- How does an XML application know how to process “<table>”?
XML: Solving Name Conflicts

We use name prefixes:

```xml
<h:table> <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
</h:tr></h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```
XML : XMLNS

- XMLNS = XML Namespaces

- Ex:

  ```xml
  <h:html xmlns:xdc="http://www.xml.com/books"
          xmlns:h="http://www.w3.org/HTML/1998/html4">
    ...
  </h:html>
  ```
XML : XMLNS

- Elements prefixed with:

XML: Well-formed documents

- Have a root element.
- Elements have a closing tag.
- Tags are case sensitive.
- Attributes must be quoted.
- Elements must be properly nested.
XML : Conversions

- We can convert XML docs to other formats.
XML : DTD

- DTD = Document Type Definition

- Purpose of DTD is to define:
  - The structure of XML document.
  - The list of legal elements.

- A DTD can be declared:
  - Inline = in the xml document.
  - External = in another file.
<?xml version="1.0"?>
<!DOCTYPE note [ 
<!ELEMENT note (to,from,heading,body)> 
<!ELEMENT to (#PCDATA)> 
<!ELEMENT from (#PCDATA)> 
<!ELEMENT heading (#PCDATA)> 
<!ELEMENT body (#PCDATA)> ]>

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
DTD : External

In note.dtd file:

```xml
<?xml version="1.0"?>
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

The rest in the .xml file with header:

```xml
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd" />
```
XML: Why use DTD?

- Describe the file’s format.
- Create a standard to exchange data.
- Verify the correctness of a received file.
XML : Parsing and DTD

- **XML parser**: check XML doc against its DTD.

- **2 types of XML parsers**:
  - Validating
  - Non-Validating

- **Validating**: checks doc conforms with DTD’s rules.

- **Non-Validating**: checks doc is well-formed.
XML : Parsing and Processing XML

- Parsing XML document (files or text) to do something with XML.

- Approaches in XML processing
  - Event based interface **SAX**
  - Object based interface **DOM**
  - eXtensible Style Language **XSL** (aka **XSLT** – XSLTransformations)
Parsing : Motivation

- Parsers shield developers from the intricacies of XML syntax
- Converting to relational tables.
- Converting to objects in programming language (class, attribute, method).
- Collecting data.
- Transforming to other formats.
Parsing : SAX

- SAX stands for Simple API XML
- SAX is an event processing.

There are events for:

- Element opening and closing tags
- Content of elements
- Entities
- Parsing errors

When event is triggered, the method associated with this event is executed.
SAX contains number of methods:
- `startDocument()` - `endDocument()`
- `startElement()` - `endElement()`
- `startPrefixMapping()` - `endPrefixMapping()`
- `characters()` - `ignorableWhitespace()`
- `processingInstruction()`
- `skippedEntity()`
- ...
Parsing : DOM

- DOM = Document Object Model represents a tree view of the XML document.
- XML DOM parser loads the XML document to memory and converts into a tree view (DOM).
- Parser traverses through the document tree
Parsing: DOM

- The DOM tree represents the XML document
- Each node represents a pair of <opening> and </closing> tags.
- Sub elements become children nodes
XSLT uses XPath- a language for finding information in an XML document.

XPath is used to navigate through elements and attributes in an XML document.

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.
And the XSLT looks like
<xsl:template match = "BOOK">
  <P>
    <xsl: value-of select = "TITLE"/>
  </P>
</xsl:template>

The transformation that will occur when a BOOK element is encountered.

→ A paragraph tag is emitted followed by the text associated with TITLE and terminated with the paragraph end tag.
THE END