CSE 510
Web Data Engineering

XML, XHTML, XSLT
and Web Application Programming
Evolving as We Speak...

cse@buffalo
XML-based Model 2 (MVC)
Architecture of Today

View:
XSLT-based presentation

XML/XML Schema-defined interaction

Model - Controller
XML-based Model 2 (MVC)
Architecture of Tomorrow

View:
XSLT-based presentation

XML/XML Schema-defined interaction

Controller

XQuery or WSDL requests / XML results

Model uses Web services and XML-wrapped sources
XHTML: HTML Without Syntactic Sugar

- Tim Berners-Lee, Robert Cailliau, Jean-François Groff: 
  The World-Wide Web. 
  *Computer Networks 25(4-5): 454-459 (1992)*

<dt name="www">
  <img src="greenball.gif"/>
  Tim Berners-Lee, Robert Cailliau, Jean-François Groff: 
  <a href="http://scholar.google.com/scholar?q=%22The
  +World%2DWide+Web%22">
    The World-Wide Web.
  </a>
  <i>Computer Networks 25(4-5): 454-459 (1992)</i>
</dt>
Executive Summary:

- **XML** = XHTML + user-definable ("semantic") tags
- **Separation** of data and its presentation

=> Simple, very flexible data exchange format:
  - *semistructured data model*

=> New applications:
  - Information *exchange* (B2B), *sharing* (diglib), *integration* ("mediation"), *archival*, ...
  - Web application development:
    - Cleaner separation of View from Controller
    - Unifying data model for multiple information sources
No Explicit Structure, Semantics, or Object-Orientation

Tim Berners-Lee, Robert Cailliau, Jean-François Groff:
\[ \text{The World-Wide Web} \]
\[ \text{Computer Networks 25(4-5): 454-459 (1992)} \]
=> HTML is inappropriate for
  - Data exchange
  - Separation of logical from presentation aspects of a Web application’s view
XML is Based on Markup

 Markup indicates structure and semantics

Decoupled from presentation
<bibliography>
<paper id="www">
<authors>
  <author>Tim Berners-Lee</author>
  <author>Robert Cailliau</author>
  <author>Jean-François Groff</author>
</authors>
<title>The World-Wide Web.</title>
<booktitle>
</booktitle>
</paper>
</bibliography>
Element Attributes

```xml
<bibliography>
  <paper id="www">
    <authors>
      <author>Tim Berners-Lee</author>
      <author>Robert Cailliau</author>
      <author>Jean-François Groff</author>
    </authors>
    <fullPaper source="http://scholar.google.com/scholar?q=%22The+World%2DWide+Web%22">
      <title>The World-Wide Web.</title>
    </fullPaper>
  </paper>
</bibliography>
```
XML, XHTML = Labeled Ordered Trees

∼ semistructured data
∼ labeled trees/graphs

can also represent
• relational and
• object-oriented data
<person id="tim"> Tim's info </person>

<bibliography>
  <paper id="www" role="publication">
    <authors>
      <author authorRef="tim">
        Tim Berners-Lee
      </author>
    </authors>
    <fullPaper source="wwwPaper"/>
    <title>The World-Wide Web.</title>
    <related papers="browsers html"/>
  </paper>
</bibliography>
• Why Stylesheets?
  – Separation of content (XML) from presentation (XSL)

• Why not just CSS for XML?
  – XSL is far more powerful:
    – selecting elements
    – transforming the XML tree
    – content based display (result may depend on data)
Using XSLT for View Components

**Client Side**
- Browser shows HTML
  - View Component runs XSLT on XML
    - XML, XSLT
  - Action component/Controller
    - XML, XSLT
  - HTML

**Server Side**
- Browser runs XSLT on XML to Produce HTML
  - XML, XSLT
  - Action component/Controller
XSLT Overview

• XSLT stylesheets are denoted in **XML syntax**
• XSL components:
  1. A language for **transforming** XML documents  
     (**XSLT**: integral part of the XSL specification)
  2. An XML **formatting vocabulary**
     (Formatting Objects: >90% of the formatting properties inherited from CSS)
XSLT Processing Model

XML source tree

Transformation

XSL stylesheet

XML, HTML, … result tree
XSLT Processing Model

- **XSL stylesheet**: collection of **template rules**
- **template rule**: (pattern \(\Rightarrow\) template)
- main steps:
  - **match pattern** against source tree
  - **instantiate template** (replace current node “.” by the template in the result tree)
  - **select** further nodes for processing
- control can be
  - **program-driven** ("pull": \(<xsl:foreach> ... \)
  - **data/event-driven** ("push": \(<xsl:apply-templates> ... \)
Template Rule: Example

1. **match pattern**: process `<product>` elements
2. **instantiate template**: replace each product with two HTML tables
3. **select** the `<product>` grandchildren ("sales/domestic", "sales/foreign") for **further processing**
Example of Turning XML into HTML

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="FitnessCenter.xsl"?>
<FitnessCenter>
    <Member level="platinum">
        <Name>Jeff</Name>
        <Phone type="home">555-1234</Phone>
        <Phone type="work">555-4321</Phone>
        <FavoriteColor>lightgrey</FavoriteColor>
    </Member>
</FitnessCenter>
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/ Transform" version="1.0">
    <xsl:output method="html"/>
    <xsl:template match="/">
        <html>
            <head><title>Welcome</title></head>
            <body>
                Welcome!
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
Extracting the Member Name

```xml
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <head><title>Welcome</title></head>
      <body>
        Welcome
        <xsl:value-of select="/FitnessCenter/Member/Name"/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
Navigating the XML Document

- Extracting values from an XML Document:
  - use the `<xsl:value-of select="...">` XSL element

- Navigating:
  - The slash ("/"") indicates parent/child relationship
  - A slash at the beginning of the path indicates that it is an absolute path, starting from the top of the XML document

/FitnessCenter/Member/Name

Start from the top of the XML document, go to the FitnessCenter element, from there go to the Member element, and from there go to the Name element
<xml version="1.0"/>

<Document>
  <Element FitnessCenter />
  <Element Member />
  <Element Name />
  <Element Phone />
  <Element Phone />
  <Element FavoriteColor />
  <Text Jeff />
  <Text 555-1234 />
  <Text 555-4321 />
  <Text lightgrey />
</Document>
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <head><title>Welcome</title></head>
      <body bgcolor="{/FitnessCenter/Member/FavoriteColor}">
        Welcome
        <xsl:value-of select="/FitnessCenter/Member/Name"/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
Note

- Attribute values cannot contain "<" nor ">
  - Consequently, the following is NOT valid:
    
    ```xml
    <body bgcolor="
      <xsl:value-of
        select='/FitnessCenter/Member/FavoriteColor'
      />
    
    ">
    
    To extract the value of an XML element and use it as an attribute
    - value you must use curly braces:
      
      ```xml
      <body bgcolor="{/FitnessCenter/Member/FavoriteColor}">
      ```

      Evaluate the expression within the curly braces, and assign the value to the attribute
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <head><title>Welcome</title></head>
      <body bgcolor="/FitnessCenter/Member/FavoriteColor">
        Welcome
        <xsl:value-of select="/FitnessCenter/Member/Name"/>
        <br />
        Your home phone number is:
        <xsl:value-of select="/FitnessCenter/Member/Phone[@type='home']"/>
        ...
    </body>
  </xsl:template>
</xsl:stylesheet>
Creating the Result Tree

• Further XSL elements for
  – Numbering
    – `<xsl:number value="position()" format="1">`
  – Conditions
    – `<xsl:if test="position() mod 2 = 0">`
  – Repetition...
More on XSL

- **XSL(T):**
  - Conflict resolution for multiple applicable rules
  - Modularization `<xsl:include> <xsl:import>`
  - ...

- **XSL Formatting Objects**
  - a la CSS

- **XPath** (navigation syntax + functions)
  = XSLT ∩ XPointer

- ...
