

IS660G - Lecture 2

Prof. Burns

Review Assignment 1

- Review XML
- What is Tomcat?
- What is Axis?
- What is Xerces?
- Chapter 6

XML

- XML was designed to describe data and to focus on what data is.
- HTML was designed to display data and to focus on how data looks.
- XML uses a Document Type Definition (DTD) or an XML Schema (XSD) to describe the data

XML

- **A "Well Formed" XML document has correct XML syntax (see Menu.xml from Chap. 6).**
 - The first line in the document is the XML declaration
 - The next line describes the root element of the document
 - All XML elements must have a closing tag `</ ...>` or `<... />`
 - All XML elements must be properly nested
 - Attribute values must always be quoted
 - Data can be stored in child elements or in attributes
 - Note: XML tags are case sensitive

XML

- A "Valid" XML document is well-formed and conforms to a DTD or XML Schema (see Menu.xsd from Chapter 6).

XML Schema

- The XML Schema language is also referred to as XML Schema Definition (XSD)
- The purpose of an XML Schema is to define the legal building blocks of an XML document
- XML Schema has Support for Data Types
- XML Schemas use XML Syntax

XML Schema

- Example: Interpret → 1999-03-11
 - 3. November or (in some other countries) as 11. March
- Alt. `<date type="date">1999-03-11</date>` ensures a mutual understanding of the content because the XML data type date requires the format CCYY-MM-DD.

What's a namespace?

- Since element names in XML are not fixed, very often a name conflict will occur when two different documents use the same names describing two different types of elements.
- xmlns attribute
 - `xmlns:namespace-prefix="namespace"`

XSL Stylesheets

- XSL - XML Style Sheets
- Stylesheets are used to transform XML documents into other formats like HTML, or, into other XML documents (adding, modifying or deleting elements).
- CSS: HTML uses predefined tags and the meanings of tags are **well understood**.
- It started with XSL (i.e. Menu.xml) and ended up with XSLT, XPath, and XSL-FO. XSL consists of three parts:
 - XSLT is a language for transforming XML documents
 - XPath is a language for defining parts of an XML document
 - XSL-FO is a language for formatting XML documents
- The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.

XSL Stylesheets

In the transformation process, XSLT uses XPath to define parts of the source document that **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. The parts of the source document that do not match a template will end up unmodified in the result document.

```

<?xml version="1.0" encoding="ISO-
8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
...
</catalog>

```

↑
xml file

→ xsl file

```

<?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>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Add → `<?xml-stylesheet type="text/xsl" href="xxxxxxx.xsl"?>` to the xml file

= Result is an HTML file in your browser

<http://localhost/examples/cdcatalog.xml>

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

Chapter 6 – ThaiPalace/ByteGourmet

- ThaiPalace – Web Service → Produces Menu
- BG – Client ← Consumes Menu

Tomcat

- Tomcat is the code name for the flagship product of the Jakarta Project . It is a commercial-quality, open-source implementation of the Java Servlet and JavaServer Pages. Specifications that runs either standalone or integrated with a web server. It is also the official Reference Implementation for those specifications (sanctified by Sun).
- Used for creating web applications (ie. Look under your Tomcat installation /webapps subdirectory)

Servlets

They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons:

- **The Web page is based on data submitted by the user.** For example the results pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well.
- **The data changes frequently.** For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.
- **The Web page uses information from corporate databases or other such sources.** For example, you would use this for making a Web page at an on-line store that lists current prices and number of items in stock.

JavaServer Pages

- JSP is a technology that lets you mix regular, static HTML with dynamically-generated HTML.
- JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion `println` statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.

JavaBeans

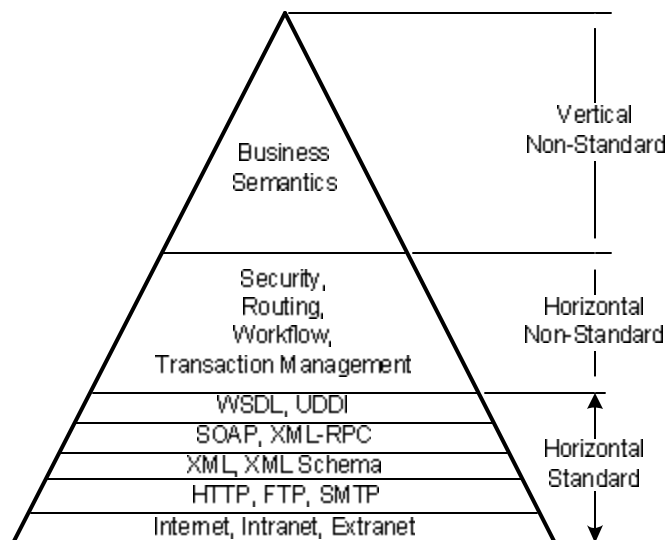
- JavaBeans brings component technology to the Java platform. With the JavaBeans API you can create reusable, platform-independent components. Using JavaBeans-compliant application builder tools, you can combine these components into applets, applications, or composite components. JavaBean components are known as *Beans*.

Pulling it all together

- MVC
 - The Model View Controller pattern was invented in a Smalltalk context for decoupling the graphical interface of an application from the code that actually does the work.

Now ... into Web Services

The Web-Services Pyramid



Web Service

- <http://localhost:8080/axis/tpmenu/ThaiPalaceMenuService.jws?wsdl>
- When you make a service available using Axis, there is typically a unique URL associated with that service. For JWS files, that URL is simply the path to the JWS file itself. For non-JWS services, this is usually the URL "http://<host>/axis/services/<service-name>".

Client

WSDL2Java tool generated 4 files based on the WSDL (ie. <http://localhost:8080/axis/tpmenu/ThaiPalaceMenuService.jws?wsdl>)

- ThaiPalaceMenuService.java (SDI)
- ThaiPalaceMenuServiceService.java
- ThaiPalaceMenuServiceServiceLocator.java
- ThaiPalaceMenuServiceSoapBindingStub.java

All are part of the same java package
(localhost\axis\tpmenu\ThaiPalaceMenuService_jws)

```

<wsdl:portType name="ThaiPalaceMenuService">
- <wsdl:operation name="getMenu">
  <wsdl:input message="impl:getMenuRequest" name="getMenuRequest" />
  <wsdl:output message="impl:getMenuResponse" name="getMenuResponse" />
</wsdl:operation>
- <wsdl:operation name="getMenuDocument">
  <wsdl:input message="impl:getMenuDocumentRequest"
name="getMenuDocumentRequest" />
  <wsdl:output message="impl:getMenuDocumentResponse"
name="getMenuDocumentResponse" />
</wsdl:operation>
- <wsdl:operation name="getMenuSchema">
  <wsdl:input message="impl:getMenuSchemaRequest"
name="getMenuSchemaRequest" />
  <wsdl:output message="impl:getMenuSchemaResponse"
name="getMenuSchemaResponse" />
</wsdl:operation>
- <wsdl:operation name="getMenuStylesheet">
  <wsdl:input message="impl:getMenuStylesheetRequest"
name="getMenuStylesheetRequest" />
  <wsdl:output message="impl:getMenuStylesheetResponse"
name="getMenuStylesheetResponse" />
</wsdl:operation>
</wsdl:portType>

```

Service Definition Interface (SDI)

ThaiPalaceMenuService.java

Methods

- getMenu
- getMenuDocument
- getMenuSchema
- getMenuStylesheet

SDI: ThaiPalaceMenuService.java

```
package localhost.axis.tpmenu.ThaiPalaceMenuService_jws;
```

```

public interface ThaiPalaceMenuService extends java.rmi.Remote {
  public java.lang.String getMenu() throws java.rmi.RemoteException;
  public java.lang.String getMenuDocument() throws
  java.rmi.RemoteException;
  public java.lang.String getMenuSchema() throws
  java.rmi.RemoteException;
  public java.lang.String getMenuStylesheet() throws
  java.rmi.RemoteException;
}

```

Or PortType Operations in the WSDL

Bindings

- A Stub class implements the SDI. Its name is the binding name with the suffix "Stub" (i.e. ThaiPalaceMenuServiceSoapBindingStub.java). It contains the code which turns the method invocations into SOAP calls using the Axis Service and Call objects. It stands in as a **proxy** (another term for the same idea) for the remote service, letting you call it exactly as if it were a local object. In other words, you don't need to deal with the endpoint URL, namespace, or parameter arrays which are involved in dynamic invocation via the Service and Call objects. The stub hides all that work for you.

Services

- Normally, a client program would not instantiate a stub directly. It would instead instantiate a service locator and call a get method which returns a stub. This locator is derived from the service clause in the WSDL. WSDL2Java generates two objects from a service clause.

```
<wsdl:service name="ThaiPalaceMenuServiceService">  
- <wsdl:port binding="impl:ThaiPalaceMenuServiceSoapBinding"  
  name="ThaiPalaceMenuService">  
  <wsdlsoap:address  
    location="http://localhost:8080/axis/tpmenu/ThaiPalaceMenuSer  
      vice.jws" />  
  </wsdl:port>  
</wsdl:service>
```

ThaiPalaceMenuServiceService.java

ThaiPalaceMenuServiceServiceLocator.java

Services

The service interface defines a get method for each port listed in the service element of the WSDL. The locator (ThaiPalaceMenuServiceServiceLocator.java) is the implementation of this service interface. It implements these get methods. It serves as a locator for obtaining Stub instances.