Developing XML Solutions with JavaServer Pages™ Technology

XML (Extensible Markup Language) is a set of syntax rules and guidelines for defining text-based markup languages. XML languages have a number of uses including:

- Exchanging information
- Defining document types
- Specifying messages

Information that is expressed in a structured, text-based format can easily be transmitted between, transformed, and interpreted by entities that understand the structure. In this way XML brings the same cross-platform benefits to information exchange as the Java™ programming language has for processing.

JavaServer Pages™ (JSP™) technology provides specification and serving of documents that combine static markup language elements and elements created dynamically by Java programming language objects.

JSP technology provides a number of capabilities that are ideally suited for working with XML. JSP pages can contain any type of text-based data, so it is straightforward to generate documents that contain XML markup. Also, JSP pages can use the full power of the Java platform to access programming language objects to parse and transform XML messages and documents. In particular, as part of the Java software environment, JSP pages can use objects that leverage the new Java APIs for processing XML data. Finally JSP technology provides an abstraction mechanism to encapsulate functionality for ease of use within a JSP page.

Generating Multiple Markup Languages

There are several approaches to generating multiple markup languages from a web application:

- Single pipeline
- Multiple pipeline
- Combination pipeline

Single Pipeline

In the single pipeline approach, the web application's JSP pages generate client-specific markup by applying transformations to incoming XML data. Each type of client requires a different stylesheet and the bulk of the development costs are associated with creating and maintaining these stylesheets.

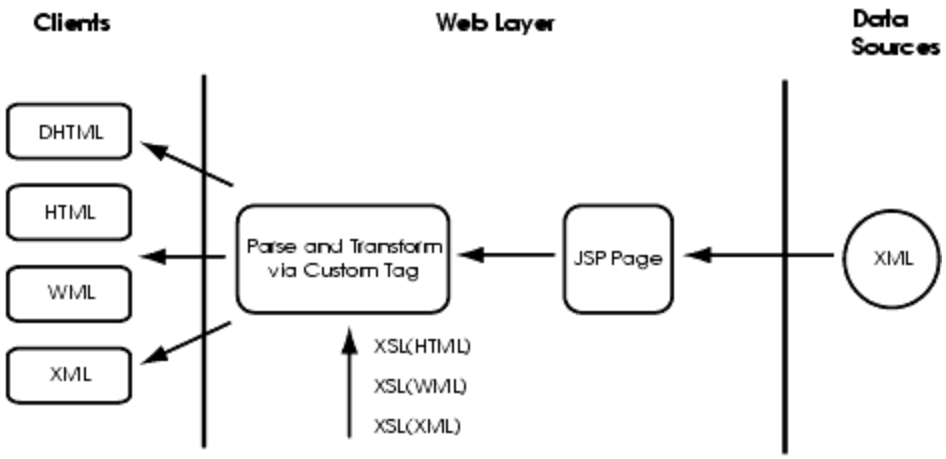

Figure 3 Single Pipeline Generating Multiple Markup Languages

This approach defers generation of both the static and dynamic portions of a response to runtime. The runtime costs are associated with:

- Parsing the XML data
- Parsing the stylesheet
- Applying the transformation

If you are using XSLT transformations, you can improve the performance of transformations by creating a binary representation (using an XSLT compiler, for example, XSLTC) of a stylesheet. However, this also makes the maintenance process more complex: each time the presentation is changed, the stylesheet must be recompiled.

Sometimes the differences between clients are minor and may not merit a full-fledged transformation. For example, there are a number of slightly different browser-based desktop clients. In some cases, one may want to take full advantage of the differences instead of generating content for the minimum common denominator. Often the differences between these clients can be encapsulated into a custom tag that generates different content depending on the properties of the client.

Generating the presentation for clients with different interaction models and flow of control (for example, PC-based browsers versus WAP phones) will require very different transformations. For example, a mobile phone cannot display a table containing book inventory data. Instead the data would have to be displayed as a set of nested lists. Supporting such transformations increases both the development and runtime costs.

Multiple Pipeline

The multiple pipeline approach uses a set of client-specific JSP pages to generate output.

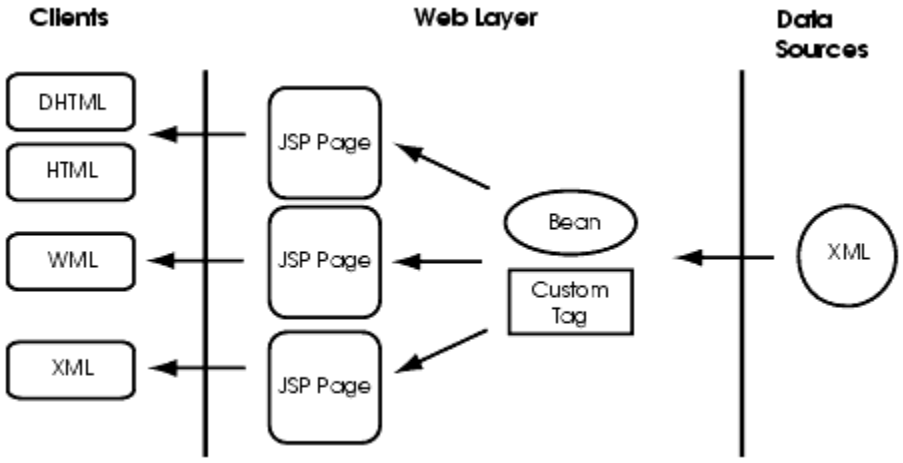

Figure 4 Multiple Pipelines Generating Multiple Markup Languages

As compared with using XSLT transformations, this approach keeps the work of creating the static content in the development phase with the dynamic content generation occurring at runtime.

Aside from creating the client-specific JSP pages, development costs are incurred in creating and maintaining server-side objects that represent the application's data abstractions. This step is not required in the single pipeline approach. Nevertheless the multiple pipeline approach can be more cost effective than the single pipeline for the following reasons:

- Data abstractions can be reused by different kinds of JSP pages.
- Data abstractions typically change at a much lower rate than presentation.
- Executing a JSP page to generate markup is much more efficient than performing an XSLT transformation to generate the same markup.

The following table summarizes the costs of each approach:

Pipeline	Development	Runtime
Single	Client-specific stylesheets	Parse XML data Parse stylesheet Apply transformation
Multiple	Data abstractions Client-specific JSP pages	Parse XML data Instantiate and initialize data abstraction components

		Execute JSP page

Combination

You can combine the single and multiple pipeline approaches. If your clients speak different languages you probably should use one pipeline for each language. To generate dialects of a language, you can apply XSLT transformations to that language's pipeline.

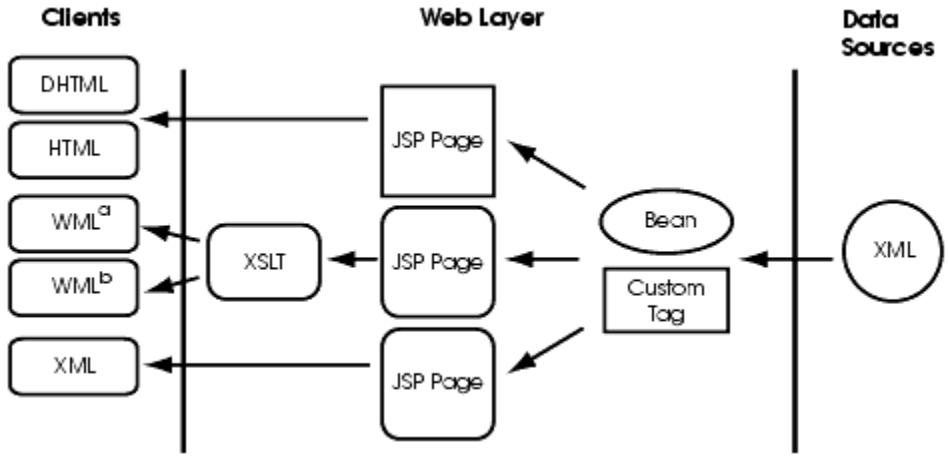

Figure 5 Combination Pipeline Generating Multiple Markup Languages

XSL Formatting Objects

XSL Formatting Objects (XSL-FO) are the second half of the Extensible Stylesheet Language (XSL). XSL-FO is an XML application that describes how pages will look when presented to a reader. A style sheet uses the XSL transformation language to transform an XML document in a semantic vocabulary into a new XML document that uses the XSL-FO presentational vocabulary. While one can hope that Web browsers will one day know how to directly display data marked up with XSL formatting objects, for now an additional step is necessary in which the output document is further transformed into some other format, such as Adobe's PDF.

Formatting Objects and Their Properties

XSL-FO provides a more sophisticated visual layout model than HTML+CSS. Formatting supported by XSL-FO, but not supported by HTML+CSS, includes right-to-left and top-to-bottom text, footnotes, margin notes, page numbers in cross-references, and more. In particular, while CSS (Cascading Style Sheets) is primarily intended for use on the Web, XSL-FO is designed for broader use. You should, for instance, be able to write an XSL style sheet that uses formatting objects to lay out an entire printed book. A

different style sheet should be able to transform the same XML document into a Web site.

Using FOP

At the time of this writing, no browser can directly display XML documents transformed into XSL formatting objects. However, there are several applications that can convert an XSL-FO document into a viewable format such as PDF or TeX. The one used here is the XML Apache project's open source FOP. FOP is a command-line Java program that converts FO (formatting object) documents to Adobe Acrobat PDF files.