

Developing with XSLT and Java™

Antony **Scott** <antony.scott@rivcom.com>

Abstract

This presentation shows how a self-contained and self-describing application can be developed entirely in using XML and XSLT, allowing implementation in any suitable platform. A Java-driven implementation of the application is demonstrated and discussed.

1. Introduction

In the past couple of years, XSLT has been widely adopted, both as a tool for producing HTML web sites, and more generally as a component in XML applications. As a key piece in the jigsaw of the XML family of standards, it has allowed developers to specify the processing of XML input independently of platform, operating system or processor. However, its scope remains limited to taking XML as input, and outputting XML, text or HTML. A developer wishing to implement XSLT-based applications in environments wider than the controlled sphere of the prototype or proof of concept may have to tackle some practical issues which XSLT on its own does not address. For example,

- how can user interaction in the web interface be controlled, and integrated with an underlying XML / XSLT application?
- given that any XSLT process requires an input file, an XSLT stylesheet and an output file, how can a developer express and implement these processing requirements, where there might be multiple, sequential XSLT transformations?
- how can XSLT components be integrated with existing applications and components, without requiring extensive rewriting of legacy code?

This presentation will look at approaches to these issues in the context of the Java environment, and demonstrated some of the practicalities of using the two technologies side by side.

2. Why XSLT and Java?

As a well supported, platform-independent programming language, Java is already widely used both to develop applications, and to deliver applications applications on the web. It has embraced XML to a considerable extent, and both the human and programming infrastructure which Java needs in order to be able to work with XML is well developed. JAXP™ (the Java API for XML Parsing) includes interfaces for both SAX (Simple API for XML) and DOM (Document Object Model), allowing processing and manipulation of XML documents.

XML and XSLT, on the other hand, provide a standard for data storage and processing which, as with Java, has been widely adopted both in the web sphere and elsewhere. Its underlying simplicity, platform and vendor independence and standards-based provenance have seen its use mushrooming in recent years.

What might be the attractions of XSLT to the Java developer, and vice versa?

To the Java developer, XSLT provides a mechanism for extracting from the body of Java code anything which is concerned with rule-based processing of XML input, and placing it in discrete, non-Java components. This has the following advantages:

- The code which specifies how XML should be processed is no longer locked up in Java classes, accessible only to the Java developer, and maintainable only by the Java developer.
- Providing XSLT processors are available, transformations can be performed either client- or server-side, depending on the system requirements, and on availability of processing capacity.
- Developers can make use of existing XSLT code, and reuse their own XSLT code, both within and outside the boundaries of the Java environment.

To the XSLT developer, Java provides a platform in which to deploy applications, and a tool for doing the things which XSLT cannot do. For example:

- A Java-based web application can take as input values from an HTML form and

output XML. Without this or a similar mechanism, XSLT cannot access these form values.

- In order to apply an XSLT stylesheet, a developer needs a mechanism to apply the stylesheet to the XML input. The Java programming language can incorporate this process into complex applications, providing the programming logic which determines which processes should be run when, and with what parameters.

3. Approaches to Java/XSLT Development

The demonstration which concludes this presentation shows the components of a possible approach to a Java/XSLT implementation strategy. The application shown is essentially XSLT-driven, and has been designed to be self-contained and hence portable. Its purpose is to allow forms to be specified and created independently of the form user's interface device and of the user's language, and to create XML output data based on the user's input which conforms to any required DTD. As much as possible of the application code has been written using XML and XSLT, including:

- specification of form structure
- validation of user input
- specification of XSLT processing requirements
- creation of XML output.

Java components have been used principally to provide the process control, and data interchange between the HTML interface and the XML data structures.

3.1. XML and XSLT Components

The principal XML components of the application are:

- A series of XML resource files, including data about

- form labels in multiple languages
- data types and validation requirements
- XSLT processing requirements
- input and output data item locations
- user settings and preferences
- The XSLT stylesheets perform transformations which include:
 - generating secondary stylesheets which are used at run time
 - creating a user interface in the appropriate format
 - creating XML output based on user input
 - validating user input
 - updating user preferences.

3.2. Java Components

The Java components used in the application are not intended as an optimal application of Java technology, but as a demonstration of different ways in which Java can be used to support XML applications. They are as follows:

- Java servlets, which run on the server and can be called from a web page. They contain Java code which can, for example, run an XSLT process, and can include parameters obtained from the web page.
- JavaServer™ pages (JSPs), which are themselves web pages, and which can contain the same Java code as a servlet.
- Java XML components including parsers and XSLT processors such as Xalan, which can implement the XSLT processing fired by the servlet or JSP.

- Compiled XSLT stylesheets or 'translets', which are Java classes containing all the processing specified in the source XSLT file. Because these classes do not require an XSLT processor in order to run, they have a performance advantage over normal XSLT stylesheets.

4. Demonstration: An XSLT / Java Application

The demonstration of the application will:

- show the functionality
- describe the overall architecture, and the parts played by each of the key components
- look at examples of XML, XSLT and Java code.

5. Summary: What are the Benefits?

Combining XSLT and Java allows the developer to:

- choose where to allocate code between the two environments, depending on the specific requirements of the application
- go beyond the boundaries of XSLT
- process XML using an XML processing language
- modularise an application in such a way as to reflect maintenance and data lifecycle needs
- maximise portability without compromising efficiency.

These benefits have already attracted huge interest in the Java community, opening the way for widespread incorporation of XSLT components into Java applications. Conversely, XML and XSLT developers can take advantage of this interest, and of the availability of Java components, to make their applications capable of easy

implementation in Java environments.

Biography

Antony Scott

RivCom

Swindon

United Kingdom

Email: antony.scott@rivcom.com

Antony Scott - Antony Scott is Director of Professional Services at RivCom, a consultancy and services company specializing in helping businesses adopt XML technologies to meet their information management and distribution needs. RivCom has been involved in the development of the XML family of standards as active members of W3C (including the XSL Working Group), OASIS; as joint project leader of the STEP/SGML harmonisation initiative under ISO; as editor of the NewsML standard; and as software development lead in the European XML/EDI Pilot Project. Antony has been developing XML applications at RivCom for the past three years, and has many years experience in writing and publishing structured documentation. During 1999, he was a co-developer on the European XML/EDI Pilot Project to develop good practice guidelines and prototype applications to show how the XML family of standards can be used to develop the next generation of EDI applications. He has spoken at a number of industry conferences over the past three years, and has run tutorials in XML and XSLT.