XJ: Integrating XML and Java

Mukund **Raghavachari**

Abstract

Current mechanisms for developing XML-based applications in Java provide minimal support for ensuring that programs are correct with respect to the XML schemas governing the XML data. With data-binding approaches such as Java Architecture for XML Binding (JAXB), a programmer must understand how the specification maps XML schemas to Java classes and not program merely in terms of the XML data. Furthermore, there is no support for ensuring the correctness of XPath expressions evaluated against documents. In these systems, a mistyped XPath expression (where one of the element names is misspelled) would not raise an error at run time --- it would silently return no results. Finally, run-time library approaches to XML, such as DOM, cannot take advantage of compiler techniques in order to optimize the performance of XML processing. The high cost of processing XML is a common complaint.

The XML Enhancements for Java (XJ) project extends Java with first-class support for XML. In XJ, one can import XML schemas just as one does Java classes. All the element declarations in the XML schema are then available to programmers as if they were Java classes. Programmers can write inline XPath expressions on these classes, and the compiler checks them for correctness with respect to the XML schema. In addition, it performs optimizations to improve the evaluation of XPath expressions. A programmer may construct new XML documents by writing XML directly inline. Again, the compiler ensures correctness with respect to the appropriate schema. By integrating XML and Java, XJ allows programmers to reuse existing Java libraries in the development of XML code and vice-versa.

The XJ language has the following advantages:

1. Familiarity (for the XML Programmer) : XML processing in XJ is consistent with open XML standards.

2. Robustness : XJ programs are strongly typed with respect to XML Schemas. The XJ compiler can detect errors in uses of XPath expressions and construction of XML data.

3. Easier Maintenance: Since XJ programs are written in terms of XML and not low-level APIs such as DOM or SAX, they are easier to maintain and modify if XML Schemas change.

4. Performance: Since the compiler is aware of the use of XML in a program, it can optimize the runtime representation, parsing, and XPath evaluation of XML.

We have released a prototype compiler and runtime system on Alphaworks (http://www.alphaworks.ibm.com/tech/xj). We will show how XJ can reduce the complexity of developing XML applications.

Table of Contents

1. Late-breaking Talk

The author did not prepare a paper for the proceedings.

Biography

Mukund **Raghavachari**
Research Staff Member
IBM Corporation [http://www.ibm.com]
Hawthorne
New York
United States of America

Mukund is a Research Staff Member in the T.J. Watson Research Center of IBM Corporation. He received a PhD from Princeton University in Computer Science. He has been leading the XJ project -- a language extension to Java that incorporates XML as a first-class constructs. Several publications in leading conferences and journals. More information is available at http://www.research.ibm.com/people/m/mrm