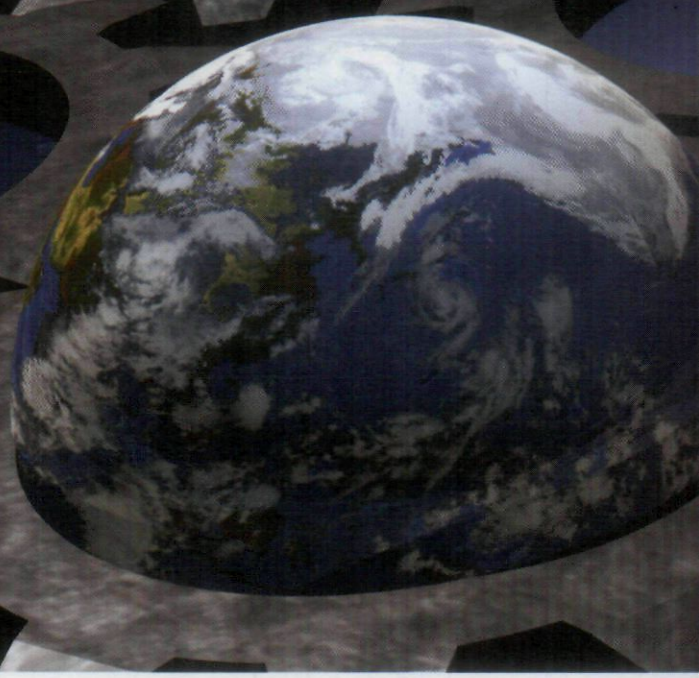
Java plus XML

A powerful new combination for SCADA systems



Java programs can be created on one platform and executed on another; likewise, XML information can be formatted on one platform and transmitted to another. Combining the two leads to the attractive dual portability of code and data.

By R. Fan, L. Cheded, and O. Toker

Java is the most common programming language for the Internet. The developers at Sun Microsystems intended it to be a platform-independent programming language so that it could run on a variety of machines under different operating systems. This was done at a time when the Internet was emerging and gaining popularity. Java allowed people working on different machines and under different operating systems to download content and applications from the Internet, and run them locally on their machines. In other words, Java provided "code portability".

The Extensible Markup Language (XML) is a method for structuring and describing information. It is a subset from the Standard Generalised Markup Language (SGML), which is a specification laid down by the World Wide Web Consortium (W3C). SGML is a generalised language for structuring information. However, it is too complex to be used for most applications. XML is more geared toward creating one type of content. It is an efficient and effective way of storing and sharing information, making it possible for a wide range of applications to easily share data in a controlled and →

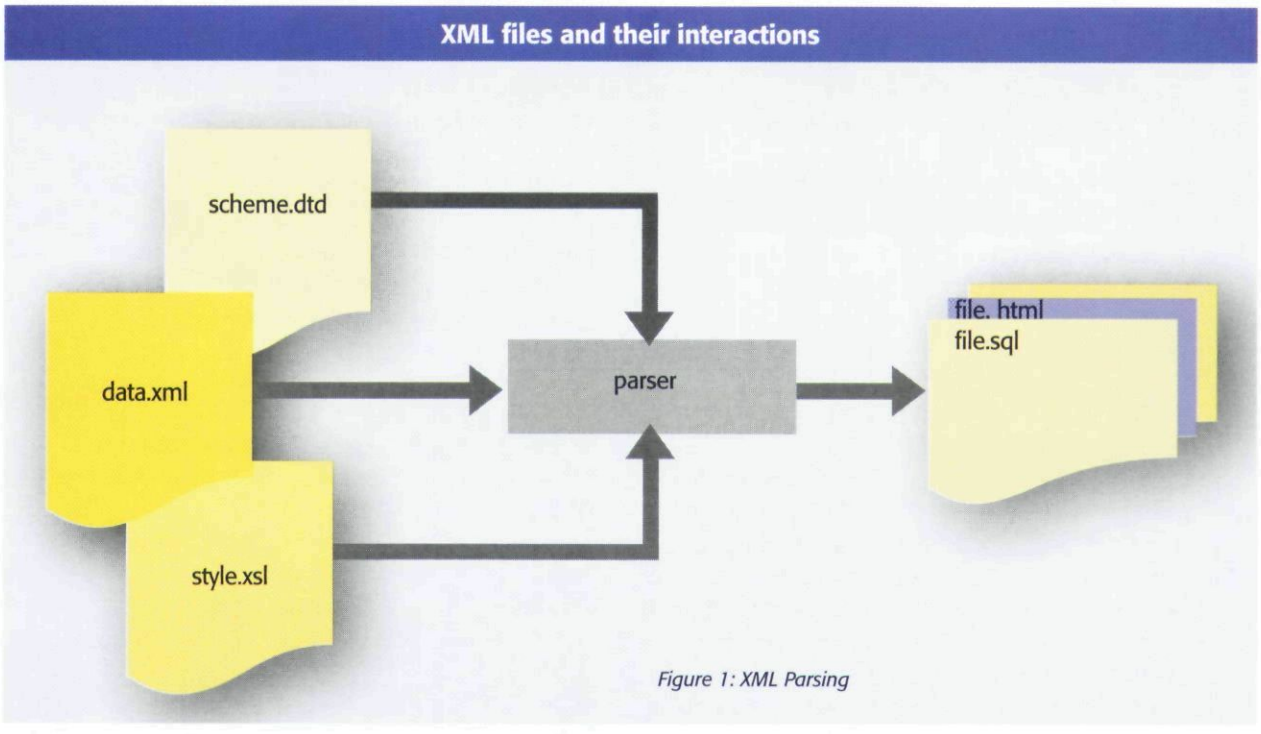


Figure 1: XML Parsing

consistent manner. Therefore, XML provides "data portability".

Since Java programs can be created on one platform and executed on another, and since XML information can also be formatted on one platform and transmitted to another, combining Java and XML leads to the attractive dual portability of code and data. Wherever Java programs can run, they can also access XML information. This enables Java and XML information to interoperate efficiently and effectively on different platforms.

The Hypertext Markup Language (HTML) is probably the most widely-known markup language. However, HTML evolved to contain tags that are used solely for formatting the display of information, thus it is considered to be limited when it comes to storing many different types of information or describing the structure of information. XML was developed to focus more on the content of information rather than on the formatting and displaying for that information. Therefore, XML separates structure from display, allowing the application to decide on how best to present the data.

XML STRUCTURE

XML consists of two parts: XML documents and XML constraints. An XML document is the file that contains the XML data. However, XML documents are not very usable without an accompanying constraint. Without document constraints, it is impossible to tell what the data in a particular document mean. There are two current standards for XML constraints: Document-Type Definitions (DTDs) and XML Schema. The DTD, or sometimes called the vocabulary, is a file that defines how the data are structured in the XML document by specifying the elements and attributes allowed in the XML document. The XML Schema is a newer standard designed to improve the DTDs by adding more typing and constructs.

XML documents are processed by special applications called parsers (Figure 1). The parser reads the document and generates output based on the document's content and the markup used to describe that content. The parser checks to make sure that the document is well-formed. Parsers that have the ability to compare the document to its DTD or schema to determine whether the document is valid, are called validating parsers. Moreover, the parser can also process a style sheet document which defines the appearance of the XML document within a browser. Because XML is an Internet and Web technology, the majority of parsers are written as Java applets.

COMBINING JAVA AND XML

Java and XML can be combined using several methods. The most basic way of accessing XML information from within a Java program is through a low-level application programming interface (API). Using these APIs, the textual content of the XML document can be directly accessed. This requires a strong XML knowledge from the developer's part. Since these low-level APIs deal with the document structure and XML rules more than with the actual data, they are commonly used either in

> **One of the main uses of Java and XML in e-business is the implementation of Web services.**

infrastructure tasks or when setting up communication in messaging.

Different approaches have been used to implement these low-level APIs. For example, the Simple API for XML (SAX) works with streamed data (it reads information from an XML input source), the Document Object Model (DOM) works with modelled data (it keeps a complete in-memory model of an XML document), and the Java API for XML Parsing (JAXP) works with abstracted data (it abstracts previous types of APIs and makes them vendor-neutral.)

The other method of combining Java and XML is to use high-level APIs. Instead of parsing and traversing the XML documents directly, the high-level APIs hide most of the details of XML structure and rules, and allow Java classes to work with the business logic implied within the document rather than with the data. This is easier and more useful when solving specific business problems, and has become quite popular with the developers of Java- and XML-based applications.

Different approaches have been used to implement these high-level APIs. For example, the mapped data approach maps data from an XML document to Java classes, while the messaged data approach uses XML as the interchange medium for data. Examples of the latter approach include the Simple Object Access Protocol (SOAP) and XML Remote Procedure Call (XML-RPC.)

The high-level API approach is generally known as the data binding approach. This usually consists of four parts: binding schema (specifying how Java classes are generated from XML constraints), class generation (creating the Java source files from XML constraints), unmarshalling (converting XML documents to Java classes), and marshalling (converting Java classes to XML documents.) This process is depicted in Figure 2.

APPLICATIONS IN E-BUSINESS

One of the main uses of Java and XML in e-business is the implementation of Web services. The term *Web service* is used to describe a service-oriented system architecture that is based on open and interoperable XML standards. It is defined as an application that exists in a distributed environment such as the Internet. It accepts a request, performs its action based on the request, and returns a response. Both the request and response usually take the form of XML, and are delivered over a wire protocol such as HTTP.

Web services can be applied in several different ways. They generally fall into three categories: (1) allowing →

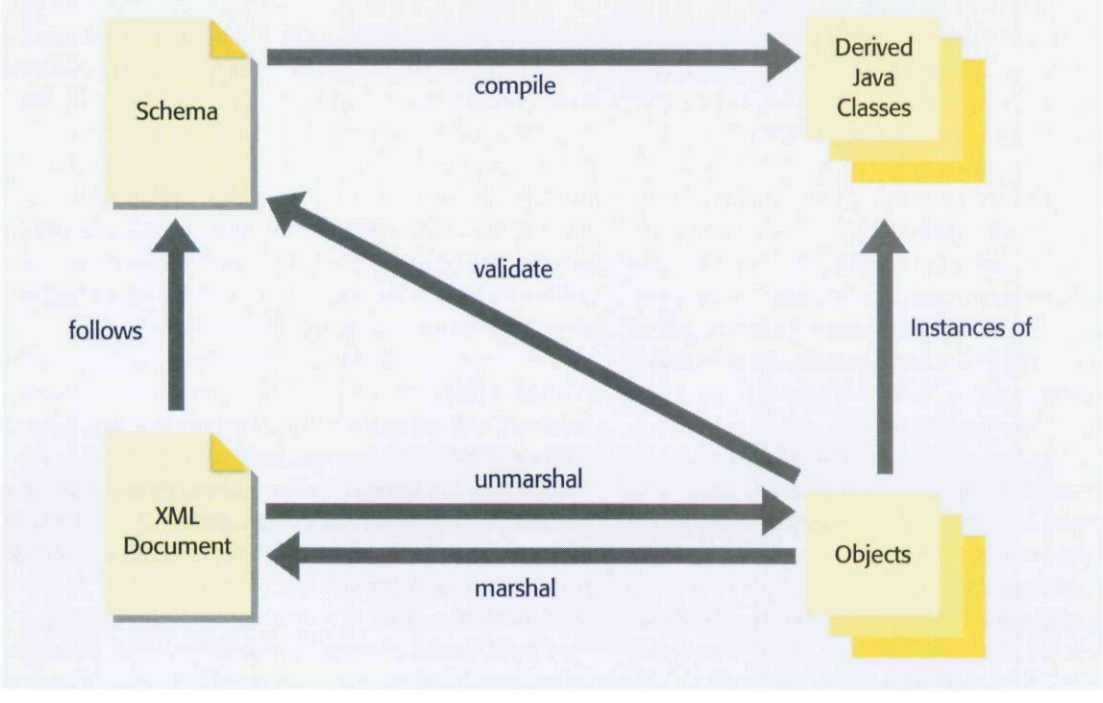


Figure 2: The high level API approach – Java and XML data binding

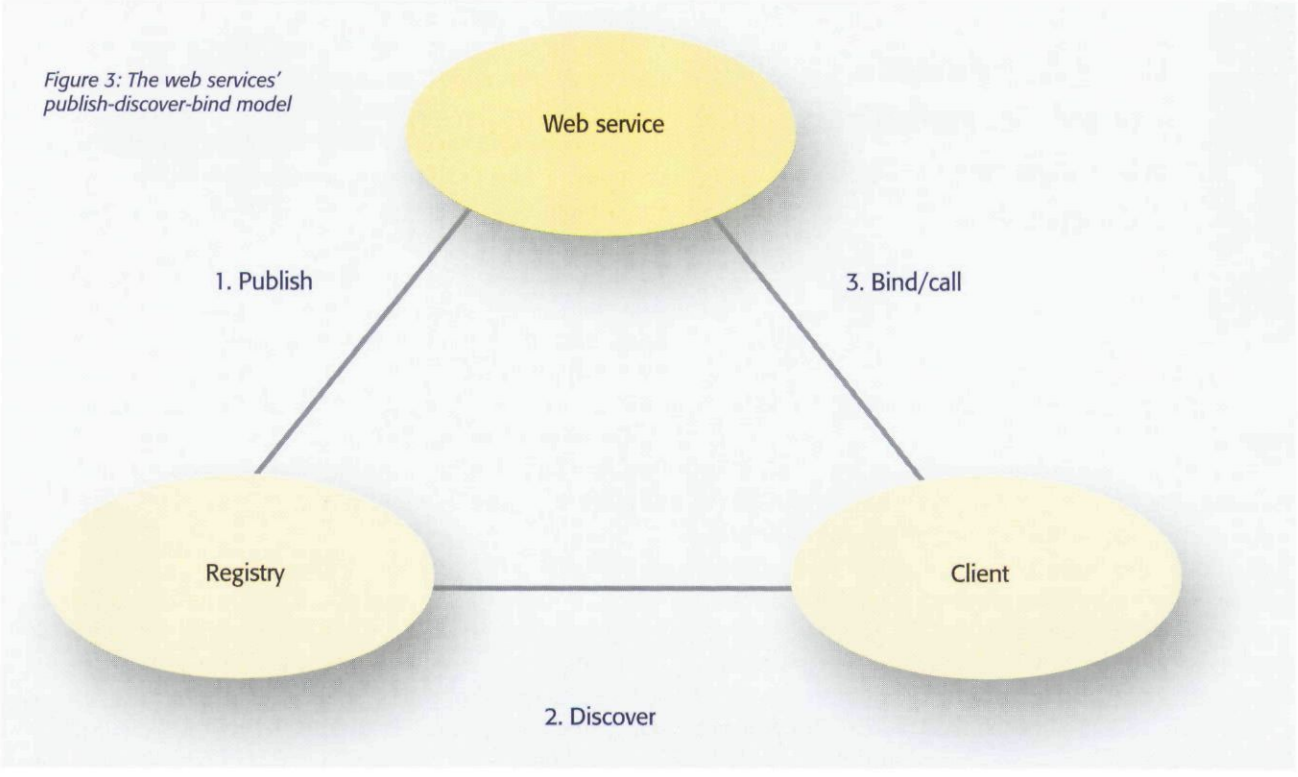


Figure 3: The web services' publish-discover-bind model

programmatic access to applications over the Internet, (2) business-to-business (B2B) integration between different organisations, and (3) application-to-application (A2A) integration within the same organisation.

This usually involves three main steps: (1) developing the Web service and publishing its features, (2) discovering and learning about the available Web services, and (3) accessing the Web services and binding to it from within the client applications. This publish-discover-bind model is depicted in Figure 3. The core technologies that underline these three main steps are WSDL, UDDI, and SOAP/HTTP.

Web Services Description Language (WSDL) is the mechanism that allows the provider of a Web service to specify the technical details of exactly which services are offered. WSDL is used to group related operations into interfaces and then provide some way to describe each of these operations. The WSDL itself is defined using XML.

Universal Description, Discovery, and Integration (UDDI) is the global registry of Web services. It allows providers of Web services to advertise their offerings in a standard way on the Internet. It also allows developers of client applications to search the available Web services and learn which interfaces they provide in order to be able to build those client applications.

Simple Object Access Protocol (SOAP) is a protocol that allows clients to invoke operations on an interface once that interface has been defined. It provides a way to identify which operation to invoke, to convey that operation's inputs as XML data, and also to return any output as XML data. SOAP forms an envelope around the XML data, and then carries it over HTTP.

The novel idea of Web services was created by a group of companies including Microsoft, IBM, Sun, Oracle, BEA, and many others. They have all endorsed the core Web services technologies of SOAP, WSDL, and UDDI. Most of these technologies have been submitted to the W3C and have become official standards of the web. The W3C standards can be referred to at http://www.w3.org.

APPLICATIONS IN PROCESS AUTOMATION

The effect of XML and Web services technologies on the process automation field has been shown in some applications; they can generally be summarised into two categories: Web enabling and Web integration.

The term Web enabling is used here to refer to the process of applying a Web interface to a control process. Java and HTML have been the main technologies to develop Web-enabled process automation systems.

The term Web integration is used here to refer to the process of integrating between different distributed subsystems using standard Web technologies. Java and XML can play a major role in this area.

In the next article, we demonstrate these concepts by describing the design of a Web-enabled control system, as well as the design of a Web- integrated SCADA system. ■