Chapter 1: Introduction

Unless you've spent the past five years in outer space, XML has probably become a regular part of your programming repertoire. Almost every industry and type of application is starting to use XML for everything from transmitting invoices to drawing pictures. There are many different APIs available for working with XML documents, but if you've never heard of SAX, it might just be the solution to problems you didn't even realize you had (yet).

Every day there are more and more ways for programmers to incorporate XML documents into the systems they are building. Unless writing an XML parser sounds appealing, in most cases you will need to use a third-party XML parser and some type of XML API.

Tree-based APIs

One of the most popular XML APIs at the moment is the *Document Object Model,* which is a standard that was developed by the World Wide Web Consortium (www.w3c.org). DOM is what is known as a *tree-based* API, which means that all of the information and content from the original document must be read into memory and stored in a tree structure before it can be accessed by a client program. Figure 1-1 shows the basic flow of an application that uses a tree-based API (like DOM) to access XML document content.

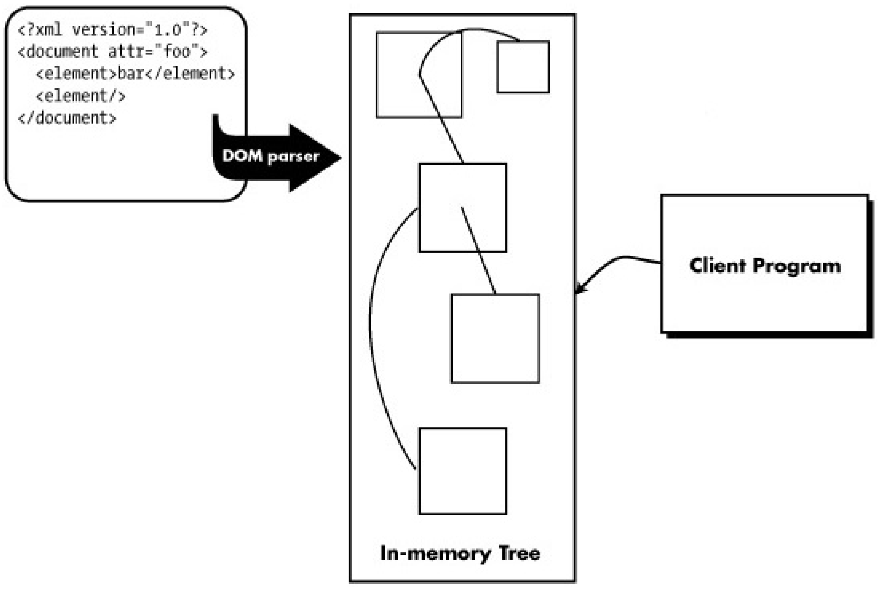

Figure 1-1: Typical flow of a tree-based XML application

Once the document has been parsed and stored as an in-memory tree structure, the client application has full access to its contents. It is simple to follow references from one part of the document to another. It is also easy to modify the document by adding and removing nodes from the tree.

While this approach has some obvious advantages, it has some equally obvious disadvantages. The size of the document affects the performance (and memory consumption) of the program. If the document is very large, it may not be possible to store the entire thing in memory at one time. Also, the whole document must be successfully parsed before any information is available to the client program.

Simple API for XML (SAX)

It was to solve these and other problems that the members of the XML-DEV mailing list (www.xml.org) developed the Simple API for XML (SAX). Unlike DOM, SAX is an *event-driven* API. Rather than building an in-memory copy of the document and passing it to the client program, the client program registers itself to receive notifications when the parser recognizes various parts of an XML document. Figure 1-2 shows the flow for a typical event-driven XML application.

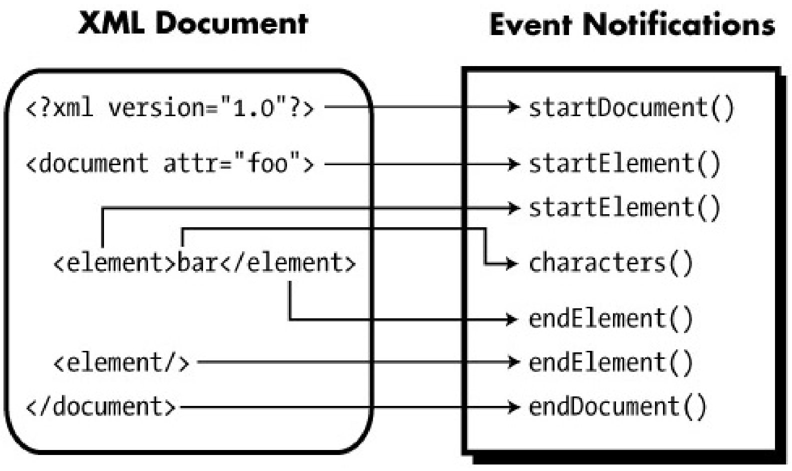

Figure 1-2: Event-based XML application flow

In the event-driven scenario, the API itself doesn't allocate storage for the contents of the document. The required content is passed to the event notification method, and then forgotten. This means that whether the document was 10 kilobytes or 10 megabytes, the application's memory usage and relative performance will remain constant.

Unlike in the tree-based approach, notifications are received as the document is parsed. This means that the client application can begin processing before the entire document has been read. For many Internet-based applications, where bandwidth may be an issue, this can be extremely useful.

There are, of course, drawbacks to this approach. The application developer is responsible for creating his own data structures to store any document information he will need to reference later. Since no comprehensive model of the document is available in memory, SAX is unsuitable for sophisticated editing applications. Also, for applications where random access to arbitrary points of the document is required (such as an XSLT implementation), a tree-based API would be more appropriate.

About This Book

This book is meant to be a complete introduction, tutorial, and reference for the Simple API for XML. The chapters in Part I progressively introduce the interfaces and classes that make up a standard SAX distribution. They should be read in order, as each chapter builds on the concepts explained in the preceding chapters.

This book attempts to introduce concepts in the order that you need to use them in a real-world application. Chapter 2 starts by showing you how to build a complete, functioning SAX application. Chapter 3 explains the SAX error handling process. Chapter 4 introduces the concept of the SAX `InputSource` class, which enables more sophisticated handling of document content. Chapter 5 explains how to capture the meta-information from the XML document type definition (DTD).

Chapter 6 explains the SAX support for the Namespaces in XML standard. Chapter 7 explains advanced SAX concepts, such as the support for SAX filter classes. Chapter 8 is intended to assist programmers who have developed applications for SAX version 1.0 to migrate to the new (and very different) SAX 2.0 API. The final tutorial chapter, Chapter 9, explains the issues that face programmers who want to write their own SAX implementations.

Part II is a complete reference to the `org.xml.sax` and `org.xml.sax.helpers` packages. Each of the classes, interfaces, and exceptions that make up a standard SAX distribution are documented. Part II is organized into two major sections: Chapter 10, "SAX 2.0 API Reference" and Chapter 11, "Deprecated SAX 1.0 API Reference."

SAX is still a very young—though well-supported—API, and the differences between versions 1.0 and 2.0 reflect the changes in XML itself. The one major change between 1.0 and 2.0 is the incorporation of full support for XML namespaces. When writing new SAX applications, only the classes and interfaces in Chapter 10 of Part II should be used. The other section is provided for historical purposes, and includes tips for quickly converting old applications to the new 2.0 API.

How to Use This Book

If you are new to SAX and have never written a SAX application before, you will probably want to continue directly to Chapter 2 and build your first application. Since each subsequent chapter introduces progressively more specialized functionality, you may find everything you need within the first two or three chapters you read. Don't feel compelled to read the entire tutorial section at once before getting started. The best way to learn SAX (or any programming concept) is by trying it out.

Once you become familiar with SAX, you will probably be more interested in using the reference material in Part II. This book makes sure to include language-specific bindings for both Java and the Microsoft MSXML implementations of SAX for all of the classes and interfaces in Part II. There are also functioning examples that demonstrate the use of each object and method given.

Now let's get started and build our first SAX application!