Introduction

It was about five years ago, a few days after I finished my first book, when the publisher came to me with a rather enticing proposal: "Why don't you start thinking about a new book?" Now I realize that all publishers make this sort of proposition, but at the time the proposal was definitely alluring, and a clear signal—I thought—of appreciation. "Because you seem to do so well with new technologies," they said, "we'd like you to have a look at this new stuff called XML." It was the first time I had heard about XML, which was not yet a W3C recommendation.

A lot of things have happened in the meantime, and XML did go a long way. You can be sure that, as I write this, a thousand or more IT managers are giving presentations that include XML in one way or another. Not many years ago, at a software conference, I heard a product manager emphasize the key role played by XML in the suite of products he was presenting. After the first dozen sentences to the effect that "this feature wouldn't have been possible without XML," one of the attendees asked a candid question: "Is there a function in which you didn't use XML?" The presenter's genuine enthusiasm led everyone there (including myself) to believe that programming would no longer be possible without a strong knowledge of XML. We were more than a little reassured by the speaker's answer: "Oh no, we didn't use XML in the compiler."

Regardless of the hype that often accompanies it, XML truly is a key element in software. Today, XML is more than just a software technology. XML is a fundamental aspect of all forms of programming, as essential as water and air to every human being. Just as human beings realistically need some infrastructure to take advantage of water and air, programming forms of life must be supported by software tools to be effective and express their potential in terms of interoperability, flexibility, and information. For XML, the most important of these tools is the parser.

An XML parser reads in XML text and outputs a memory representation of the contents. The input for an XML parser is always plain and platform-independent text, although potentially encoded in a variety of character sets, whereas the output of an XML parser is strictly tied to the underlying hardware and software platform. Depending on the operating system and the programming environment of choice, an XML parser can generate a Component Object Model (COM) object as well as a Java or a JScript class. No matter the kind of output, however, the end result is XML data in a programmable form.

The growing level of integration and orchestration that partner applications need makes the exchanged XML code more and more sophisticated and often requires the use of specialized dialects like Simple Object Access Protocol (SOAP) and XPath. As a result, XML programming requires ad hoc tools for reading and writing in these dialects; all the better if the tools are tightly integrated into some sort of programming framework.

Effective XML programming requires that you be able to generate XML in a more powerful way than merely concatenating strings. The XML API must be extensible enough to accommodate pluggable technologies and custom functionalities. And it must be serializable and integrate well with other elements of data storage and exchange, including databases, complex data types (arrays, tables, and lists), and—why not?—visual user interface elements. In simple terms, XML must no longer be a distinct API bolted onto the core framework, but instead be a fully integrated member of the family. This is just what XML is in the Microsoft .NET Framework. And this book is about XML programming with the .NET Framework.

What Is This Book About?

This book explores the array of XML tools provided by the .NET Framework. XML is everywhere in the .NET Framework, from remoting to Web services, and from data access to configuration. In the first part of this book, you'll find in-depth coverage of the key classes that implement XML in the .NET platform. Readers and writers, validation, and schemas are discussed with samples and reference information. Next the book moves on to XPath and XSL Transformations (XSLT) and the .NET version of the XML Document Object Model (XML DOM).

The final part of this book focuses on data access and interoperability and touches on SQL Server 2000 and its XML extensions and .NET Remoting and its cross-platform counterpart—XML Web services. You'll also find a couple of chapters about XML configuration files and XML data islands and browser/deployed managed controls.

What Does This Book Cover?

This book attempts to answer the following common questions:

- Can I read custom data as XML?
- What are the guidelines for writing custom XML readers?
- Is it possible to set up validating XML writers?
- How can I extend the XML DOM?
- Why should I use the XPath navigator object whenever possible?
- Can I embed my own managed classes in an XSLT script?
- How can I serialize a *DataSet* object efficiently?
- What is the DiffGram format?
- Are the SQL Server 2000 XML Extensions (SQLXML) worth using?
- Why does the XML serializer use a dynamic assembly?
- When should I use Web services instead of .NET Remoting?
- How can I embed managed controls in Web pages?
- How can managed controls access client-side XML data islands?
- How do I insert my own XML data in a configuration file?

All of the sample files discussed in this book (and even more) are available through the Web at the following address: *http://www.microsoft.com/mspress/books/6235.asp*. To open the Companion Content page, click on the Companion Content link in the More Information box on the right side of the page.

Although all the code shown in this book is in C#, the sample files are available both in C# and in Microsoft Visual Basic .NET. Here are some of the more interesting examples:

- An XML reader that reads CSV files and exposes their contents as XML
- An extended version of the XML DOM that detects changes to the disk file and automatically refreshes its data
- A Web service that offers dynamically created images
- An XML reader class with writing capabilities
- A class that serializes *DataTable* objects in a true binary format
- A tool to track the behavior of the XML serializer class
- A *ListView* control that retrieves its data from the host HTML page

These and other samples will get you on your way to XML in the .NET Framework.

What Do I Need to Use This Book?

Most of the examples in this book are Windows Forms or console applications. The key requirements for running these applications are the .NET Framework and Microsoft Visual Studio .NET. You also need to have SQL Server 2000 installed to make most of the samples work, and a few examples make use of Microsoft Access 2000 databases. The SQLXML 3.0 extensions are required for the samples in Chapter 8. The code has been tested with the .NET Framework SP1.

The SQL Server examples in this book assume that the sa account uses a blank password, although the use of such a blank password is strongly discouraged in any professional development environment. If your SQL Server sa account doesn't use a blank password, you'll need to add the sa password to the connection strings in the source code. For example, if your sa password is "Hello", the following connection string provides access to the Northwind database:

```
string nwind =
"SERVER=localhost;UID=sa;pswd=Hello;DATABASE=northwind;";
```

Some of the applications in this book require SOAP Toolkit 2.0 and SQLXML 3.0. These products are available at the following locations:

- **SOAP Toolkit 2.0**
  http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/MSDN-FILES/027/001/580/msdncompositedoc.xml
- **SQLXML 3.0**
  http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/MSDN-FILES/027/001/824/msdn-compositedoc.xml

Contacting the Author

Please feel free to send any questions about this book directly to the author. Dino Esposito can be reached via e-mail at one of the following addresses:

- <dinoe@wintellect.com>
- <desposito@vb2themax.com>

In addition, you can contact the author at the Wintellect (*http://www.win-tellect.com*) and VB2-The-Max (*http://www.vb2themax.com*) Web sites.

Support

Every effort has been made to ensure the accuracy of this book and the contents of the sample files. Microsoft Press provides corrections for books through the Web at the following address:
http://www.microsoft.com/mspress/support/

To connect directly to the Microsoft Press Knowledge Base and enter a query regarding a question or issue that you might have, go to:
http://www.microsoft.com/mspress/support/search.asp

If you have comments, questions, or ideas regarding this book or the sample files, please send them to Microsoft Press using either of the following methods:

Postal mail:

Microsoft Press
Attn:*Microsoft .NET XML Programming* Editor
One Microsoft Way
Redmond, Wa 98052-6399

E-mail:

<MSPINPUT@MICROSOFT.COM>

Please note that product support is not offered through the above mail addresses. For support information, please visit the Microsoft Product Support Web site at *http://support.microsoft.com*