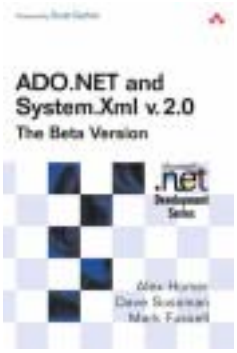
ADO.NET and System.Xml v.2.0 - The Beta Version



book review by Craig Murphy

There's no doubt about it, XML as a means of representing data has arrived. Whilst Microsoft's .NET framework 1.1 did a good job of bringing XML and data representation/access together, it's fair to say that the .NET framework 2.0 brings with it significantly more integration and ease of use.

I was pleased to be given the chance to pick up a copy of this book hot off the press direct from one of its authors: Alex Homer. It's unlikely to arrive in the UK officially until April 30th!

The UK's software legends, Dave Sussman and Alex Homer teamed up with Microsoft's Mark Fussell to produce this 528-page (1" thick!) coverage of ADO.NET, the System.Xml v.2.0 namespace and XML integration with SQL Server 2005. It's about as bang up-to-date as you can get, and you're reading about it here!

Introduction

When Microsoft introduced the .NET programming framework, many people were surprised at the fundamental changes to data access techniques that it encompassed. Not only was there a new model for both connected and disconnected relational database management, but a raft of new ways to work with XML. And now version 2.0 of the Framework is on its way, bringing with it fresh opportunities and new techniques that not only extend the reach of the technology, but also make many common tasks much easier to accomplish than ever before. This book previews and explains the features of the new versions of ADO.NET and System.Xml, based on the Beta release. It includes asynchronous commands and multiple active results sets, SQL Server 2005 integration, the universal query architecture, plus the enhancements to the DataSet, XPathDocument, the new XQuery support, and more.

Book Structure

There are 12 chapters, each of which is a good mix of narrative and code. There's nothing in the way of reference-style content, i.e. page after page of method listings and property listings that merely regurgitate a help file.

Two authors contribute to a well-balanced Foreword, kept down to a shade over four pages. Both Forewords were written by Microsoft employees working in the WebData Team: Michael Pizzo and Soumitra Sengupta. The WebData team goes back a long way, as far as I know, at least to the year 2000 and probably earlier. These guys have been so close to XML and data access, it positions them perfectly as either budding authors of this book or as readers/reviewers. I think it's safe to say that this book has been read by the very people who made ADO.NET and System.Xml v.2.0 a reality: prior to that it was written by the three people most qualified to turn a technical topic into readable narrative.

Chapter Content

New Concepts in Data Access

This first chapter wastes no time in setting the scene, both history (.NET 1.0) and current/future (.NET 2.0). On the first page, mention of the fact that the Common Language Runtime (CLR) can now be hosted inside of SQL Server 2005 (Yukon) suggests that what we are about to read is going to be a factual and straight-to-the-point account of .NET 2.0.

The chapter then goes on to provide a summary of The Evolution of Data Management in .NET, information About the .NET Version 2.0 Beta Release and New Concepts in Version 2.0. It also provides A Summary of New Features in ADO.NET and System.Xml: this is an excellent section that draws a neat map (textually) of the remainder of the book, covering such areas as ADO.NET enhancements, SQL Server 2005 Integration and XML Enhancements.

Lastly, this chapter discusses the New Data Source Controls and Data Binding Features in ASP.NET.

ADO.NET Data Management Enhancements

ADO.NET v.1.0 heralded the introduction of disconnected data access and sported a number of classes allowing us to work with data in the absence of a physical connection to the database. This chapter discusses ADO.NET v.2.0's ability to access data synchronously, its integration with the .NET System.Transaction namespace (allowing local transactions to promoted to a distributed transaction), batch updating using a DataAdaptor and the new classes available to managed code allowing the bulk copying of data (similar to SQL Server's Bulk Copy Program, BCP).

This is the first chapter to mention MARS: Multiple Active Results Sets. MARS is a SQL Server 2005 feature that allows us to open multiple result sets over a single connection – you'll need ADO.NET v.2.0 in order to make use of it. When would MARS be useful? Well, if you use a single table to hold your entire application's collection of drop-down or pop-up menu items, you may well need to fire off multiple queries at the same table in the same database.

The narrative in this chapter explains the need for synchronous and asynchronous access to data and ties it in nicely with its discussion about MARS. Clearly there are issues with MARS (what it does) and asynchronous data access: this chapter explains those issues succinctly.

Provider Factories, Schema Discovery, and Security

Today's modern applications don't limit themselves to a single database. In fact, from a corporate perspective, it pays to select a vendor whose application runs on a choice of database platforms. This chapter discusses the ADO.NET 2.0 APIs that we can use to help make our applications database-agnostic, or database-neutral. Such neutrality is achieved by introducing the concept of layering or interface / protocol stacking.

This chapter provides a fairly in-depth and technical look at the Provider Factory classes, the DBConnectionStringBuilder class, the Schema Discovery API, Security and Performance. Whilst this is required reading, particularly before an application is architected and a database platform is firmed up, I did think that this chapter arrived rather early in the book.

The DataSet and DataTable Classes

The performance enhancements between the DataSet class in ADO.NET 1.0 and the same class in 2.0 are presented in an eye-catching graph – the reader could hardly fail to miss it. A comparison of the 1.0 DataSet and the 2.0 DataSet during an insert rows benchmark demonstrates that large numbers of inserts in a 2.0 application should take less than half as long as a 1.0 application.

Coverage of the new features found in the DataSet class follows, making reference to future chapters, whetting the appetite for an XML datatype that we'll learn more about later.

Again, this chapter is rather in-depth and does appear as a reasonably low chapter number. However, since we cannot avoid working with the DataSet and DataTable classes, such coverage is welcomed.

ADO.NET and SQL Server 2005

This chapter covers the merging of two separate items of functionality: SQL Server 2005 and ADO.NET. It does so by focusing on three areas of functionality: MARS, SQL Server Query Notifications and SQL Server User-Defined Types (UDTs).

MARS was first mentioned in chapter two, albeit at a rather high level. This chapter goes into much more detail regarding the use of MARS, taking nine pages to do so.

SQL Server Query Notifications essentially invert the processing logic: instead of a client application periodically polling the database for changes, Query Notifications allow the client to register their interest and be told whenever a particular piece of data (obtained via a query) has been changed or invalidated. The authors spend some 15 pages covering Query Notifications and make good use of seven small examples.

Given that we've already learnt that this book explains the fact that SQL Server can host any CLR-compliant language, it should come as no surprise to learn that this chapter also discusses UDTs and how we can extend SQL Server's existing data types with those from a managed code environment. Some 12 pages and 5 code examples explain when and how UDTs should be used.

SQL Server 2005 CLR Hosting

This chapter's key takeaway is the ability to write SQL Server 2005 stored procedures using a CLR-compliant language such as Visual Basic .NET, C#. The SqlProcedure attribute allows us to mark or decorate a piece of code as being a stored procedure.

XML in SQL Server 2005

This chapter starts by making reference to SQL Server 2005 as an XML database. This is excellent news. The new XML datatype allows us to store XML in a typed form such that it conforms to an XML Schema. This kind of functionality is something we had to do manually prior to SQL Server 2005. The XML datatype also allows us to use column names in SQL statements and stored procedures.

A good discussion about the XML Schema Repository in SQL Server 2005 then follows. XML Schema is worthy of a book in its own right. The authors carefully recognise this and make excellent use of a simple example that covers the salient points, thus keeping this discussion focused on the Schema Repository.

Now that structure and schema have been covered, the authors then move on to discussing inserts and selects, using the xml datatype. It should come as no surprise to learn that XML has its own query language: XQuery (learn more about XQuery in The Delphi Magazine, Issue 75, November 2001). XQuery uses the XPath language to query an XML document that is held in a SQL Server 2005 table. Again, XQuery and XPath are topics in their own right and, again, the authors have noticed this and do not try to explain these subjects in depth. Instead, they choose to focus on the common questions / pitfalls that you and I might come across when trying to work with XML-based querying, such as XML Namespaces and the rather interesting subject of binding relational data inside XML.

The XML datatype is also implemented in ADO.NET. A brief discussion about the changes to the underlying namespaces, classes and the methods they expose follows. I was pleased to see the authors highlight the fact that certain field types and property names might change between the beta release and the final release of .NET v.2.0. Working with SQL Server 2005's XML datatype via ADO.NET includes coverage of: reading XML via a DataReader, updating an XML column with a Command, updating an XML column using an XML DML Statement, reading and updating the XML Data Type with a DataSet or DataTable, loading a DataTable containing an XML-Typed column and updating a DataTable containing an XML-Type column.

The remainder of this chapter, seven pages and five screenshots, covers how to use the XML Classes in the SQL Server CLR.

XML in the .NET Framework

This is by far the most easy to read and succinct introduction to XML and its associated technologies that I have seen to date. It covers the importance of XML, why you need it, a look at the System.Xml v.1.0 namespace, what's new in the System.Xml v.2.0 namespace, XML Support in Visual Studio 2005 and an introduction to XQuery.

Regular readers of my work will probably have a good idea as to why XML is so popular and will know many reasons why we need it. This book's authors position XML within the context of the XML specifications such as XML 1.0, DOM 1.0, XPath 1.0, XSLT 1.0, SOAP 1.2, XML Schema and the XML Information Set. Whilst these specifications don't make easy reading, this chapter covers their salient points, giving new readers a good overall grounding in the history of XML.

There is good coverage of XML's relationship (no pun intended) to databases, its use in web applications, description of data via schemas, transformation and presentation via XSLT and querying via XQuery. Mention of its ability to form part of a content publishing framework is touched on but the focus quickly moves on to XML in .NET.

Through effective use of diagrams, XML's position with the .NET framework becomes clear. The boundaries between the XML world and the relational world are evident, even to the point that the notion of serialising a class as XML, for dissemination over a network, is touched upon.

Whilst not very detailed, a reasonable discussion of the System.Xml v.1.0 namespaces follows. It covers such topics as: XML reading and writing, XML document editing, XML validation and content checking, XML querying, and XSL transformation. I see this chapter as a directional chapter: it gives you the pointers and it's up to you to perform the in-depth research. This is an admirable approach and one that makes the chapter that bit easier to read.

System.Xml v.2.0 is then explained, using the same headings as noted above for v.1.0. XML support in Visual Studio 2005 is discussed, intermingled with a handful of screenshots. Importantly, with XSLT now being compiled, the fact we can now debug XSLT using the Visual Studio 2005 debugger is a gem not missed by these authors! Indeed a screenshot of the XML Schema editor, which looks rather similar to the XML Schema editor in earlier versions of Visual Studio, is also included.

This chapter closes with ten pages explaining XQuery and how it can be used for querying XML data.

Reading and Writing XML

I do like the way the authors present scenarios at the start of most chapters. This is an ideal mechanism for communicating what the chapter is about to cover and, importantly, why we might need the chapter itself.

Given that this chapter is nearly 90 pages long, it really only covers reading (loading) and writing (creating) XML documents and fragments. However, it does go into reasonable detail covering how XML can be validated (for type and structure) using Document Type Definitions and XML Schema.

Good coverage of how to deal with XML fragments, i.e. small parts of a larger XML document, and how to deal with them using code makes this chapter a good read. Similarly, there are a few good examples that cover most XML developers' bane: XML namespaces. Kudos to the authors for providing these examples. Further coverage borrows from the other chapters where we learnt that System.Xml v.2.0 now supports Typed Content Accessor methods, thus allowing us to work with XML as if it were typed.

I was pleased to see code examples of the ReadTo methods. These allow us to open an XML document/datastore and navigate to a particular element with just one line of code. Essentially, the ReadTo methods map on to XPath axes (more information about XPath axes can be found here: http://www.craigmurphy.com/bug/XML/XMLPPT1.zip, slide 30).

Creating XML is the job of the XmlWriter class. Of course, the authors cover creation of an untyped XML document and a typed XML document. They also demonstrate how to integrate the XmlReader and XmlWriter classes to create an HTML page from an RSS feed: the code was simplicity and evidence enough, no screenshot was required and none was given.

Tagged on towards the end of this chapter are six pages about Security and XML. With XML's popularity, it's only a matter of time before somebody finds a means of causing havoc using XML…and that would throw the entire industry into mayhem. Restricting DTD parsing and XSLT processing are two such security mechanisms that the authors cover, both with narrative and code examples.

The chapter wraps up with a look at how we might create an XML Schema from an existing XML document, a process known as inferring an XML Schema from an XML Document. We need this kind of functionality in order to make good use of XML Documents with SQL Server 2005 – it requires an XML Schema before it will look at XML. Whilst XML Schema inference is embedded within the Visual Studio IDE, it is also surfaced via the XmlSchemaInference Class, thus we can access it programmatically.

XML Serialization Enhancements

As the authors rightly point out in this chapter's opening salvo, serialization (or serialisation for us Brits), isn't something you might find yourself doing much. Or so you thought. If you've been building web service-based applications using either Visual Studio or Delphi, under the hood, the web service architecture has been serialising your data and classes on your behalf.

Pre-generation of serialisation assemblies is covered, albeit rather lightly. The command-line syntax for the Serialisation Generation Tool, sgen, is given, as is a worked example covering two pages. A further three pages explain the enhanced operation of the IXMLSerializable interface, using two pages of code to demonstrate it in practice. This chapter, whilst useful, has a high code to narrative ratio, which isn't necessarily a bad thing – serialisation is important so perhaps it's best demonstrated via a reasonable sized code example.

XML Document Stores

I'm surprised that this chapter appears so late in this book: I would have thought it would have been presented earlier. It spends much of its 45 pages setting the scene for XML, using the XmlDocument class, limitations of the XML DOM, design guidelines for exposing XML from your classes, the XPathNavigator with cursor-editing model, using XPath queries, inserting attributes and elements using the XPathNavigator and adding and removing elements using the XPathNavigator. I think we can conclude that the XPathNavigator plays a major part in the System.Xml v.2.0 namespace. Indeed, it was read-only in the v.1.0 namespace, now it has been opened up for editing.

The XPathNavigator API is covered using a series of small examples – these demonstrate the salient points without too much noise. The XmlDocument class is now capable of understanding XML Schema (previously we had to use a validating XmlReader). This has the advantage of making our XmlDocument instances type aware. Obviously some sort of mapping between XML Schema types (XSDs) and CLR types needs to be defined. This is a subject the authors choose to explain via a mapping table and a code example making use of the CLR type Double.

A further eight pages cover the ins and outs of validation against an XML Schema, making good use of short examples to convey the salient points. The remainder of the chapter covers XPath queries, drawing upon the authors' knowledge and experience to answer many frequently asked questions (FAQs from newsgroups).

Transforming XML Documents

Chapter 12's great revelation is the introduction of compiled XSL Transforms, introduced by the XslCompiledTransform class. Naturally, the authors waste no time in benchmarking the performance of the compiled XslTransform class and the .NET 1.1 XslTransform class. You will be interested to note that the new XslCompiledTransform class offers 200% to 400% performance gains depending upon your scenario. How is this performance gain realised? Well, XSLT is now compiled into .NET's intermediate language: MSIL. If XSLT can be compiled into MSIL, you can perhaps imagine the incredible side-effect that this has: creation of a program database (PDB) file is now possible, as is debugging via an XSLT debugger!

With XSLT now being compiled, it should come as no surprise that Microsoft have allowed XSLT to be extended to call out to code that has been written outside of the .xsl file itself. This concept isn't new and has been around since the days of Microsoft's Core XML Services 3 (MSXML3) – it was possible to call methods written using JavaScript, VBScript, or even a dll. A concise example is presented, demonstrating the ability to encapsulate functionality within a separate Visual Basic.net class, instantiate the class from a .xsl transform and use methods declared within the .xsl transform.

This chapter also provides a few words revolving around the subject of when to use the eXtensible Stylesheet Language for Transformation (XSLT). This is a welcome addition and it answers one of my most popular after-presentation questions.

Lastly, this chapter discusses XSLT security. It's perhaps something not a lot of us think about, but with script hacking becoming increasingly popular, it is worthy of a mention. Whilst basic .xsl transformations are reasonably harmless (watch this space, I'll eat my hat!), XSLT's scripting capabilities introduce the need for security. Two settings are discussed: XsltSettings.EnableScript and UnmanagedCodePermission. Like most Microsoft offerings these days, the EnableScript property is set to false. I would like to have seen a little more written about the XSLT security concerns and solutions ("go and write it yourself", I hear you cry!). However, I'm sure that .NET v.2.0 has been designed with security in mind, besides security sometimes has to be retrospective: until there's an attack, we don't know what to secure!

Conclusions

I have only two gripes about this book: the code samples are written using Visual Basic.net and the book does assume some prior knowledge. Luckily, the authors have been kind enough to provide C# versions of the code on their web-site! Prior knowledge shouldn't put you off; .NET has been with us publicly since 2000. Unless you've been living under a rock for the last five years, it's very likely that you've found yourself reading some .NET 1.0 and 1.1 material.

Now that Dave and Al have moved to Addison Wesley, we can say adios to large tomes comprising three-fifths code and narrative and two-fifths appendices and sometimes the other way around. This is a good read; it is straight to the point and hits the nail on the head for a number of potentially scary subjects. If you are thinking of buying a book with the title: ADO.NET and System.Xml v.2.0, the beta version, I think it's fair to say that you're coming from a reasonably technical background and that you have an interest in where the technology's come from and going to.

Whilst the code samples in the book have been edited down, full versions are available from Dave and Al's web-site. I would recommend having the full code samples available to you, even if you don't compile/run them, but to use them for reference.

If you have enjoyed some success with .NET 1.1 and you are now planning a .NET/XML project and intend to use .NET 2.0, I can strongly recommend this book as a great introduction to the salient points: the narrative is there, but has not been padded out. This book is clearly targeted at a developer audience, particularly an audience with some .NET 1.0 or 1.1 exposure and a moderate amount of XML awareness.

About This Book

Expect to pay £22.39 from Amazon for **Ado.Net and System XML V. 2.0: The Beta Version** in paperback, 528 pages (marked as 400 at Amazon), published April 30, 2005 by Addison Wesley, ISBN: 0321247124.

Other Resources

- http://www.daveandal.net
- http://www.awprofessional.com/title/0321247124#
- Learn more about XQuery in The Delphi Magazine, Issue 75, November 2001
- http://www.craigmurphy.com/bug/XML/XMLPPT1.zip

InstallAWARE Product Promotion

InstallAWARE 33% discount offer - all editions

Upgrade your installation experience to InstallAWARE: Partial web deploy saves you bandwidth, and your customers download time, by isolating runtimes (or rarely used application features) from your installation, while still producing a fully self contained setup.exe that works without requiring an Internet connection. Genuine scripting for Windows Installer makes authoring MSI based packages a snap and isolates you from all aspects of Windows Installer - you visually build your setup script, including conditional branching, and InstallAWARE converts it to an MSI package automatically at build time. The dialog editor is very similar to Delphi/C++Builder and has advanced controls like interactive Flash/HTML billboards, and makes it very easy to pass values between your setup script and user interface. InstallAWARE also has many more unique features - LZMA/BCJ2 compression, one-click patch creation, debugging - all to save you time and effort while building your setups.

Just click www.installaware.com/buy.asp to order, and choose any crossgrade option for an instant 33% discount.

InstallAWARE Free copies

We are raffling three Express and one Developer licenses at each of our meetings in May, June and July. These normally sell at £55.55 (£38.72 at the discount price above) and £195.80 (£139.70). Have a look at the excellent web site www.installaware.com for a table showing the features of each edition. You can also compare InstallAWARE with other products. [Ed]

About InstallAWARE

InstallAWARE has been designed from the ground up to meet today's setup authoring needs. It provides a perfect fusion of Windows Installer, Web Deployment, and Scripting technologies. Developers can author Windows Installer setups in complete isolation from all aspects of MSI tables, databases, sequences, and so on. They just create a conditional setup script, and at build time, InstallAWARE magically converts that script into a legal, logo-certifiable MSI database. Moreover, developers can break up their setups into multiple online and offline parts, removing rarely used application features and runtimes from the main setup executable, but still making sure that the essential application data is contained within the main setup executable. With other products, its all or never web deployment, with no choice on which features go where.

Other unique InstallAWARE features are: Flash/HTML billboards for interactive progress dialogs, a choice of 12 (twelve) setup themes, a sophisticated dialog editor with a rich set of controls, superior compression which does wonders in eliminating the shared overhead in executable files, and also tightly compresses runtimes (the entire .NET runtime gets reduced to 11MB, compared to the regular 23MB), and an innovative two-way linked IDE which has a visual view that generates the installation script automatically, along with the code view that provides complete access to the heart of the installation.

What developers love about InstallAWARE is that it lets them break through the confines of Windows Installer, and author a setup script the intelligent way - they spend time coding their setup logic, instead of worrying about populating MSI database tables; but the end result is always a logo compliant Windows Installer installation.