00FMADOR2.qxd 2/18/05 9:37 AM Page xxv

Forewords

By Michael Pizzo*
Software Architect, WebData Team
Microsoft Corporation

It's 2:30 p.m. and I'm on a flight bound for Michigan, wondering why I agreed to spend the first hours of my five-day vacation writing the foreword for a book about data and XML.

The setting reminds me of a flight to Sydney a few years ago to give a presentation on Microsoft's latest set of data interfaces. In preparing for the presentation I thought back to how I had first gotten involved with data access.

It all started innocently enough about 12 years ago when I was a fresh young program manager working on Microsoft Excel. While we had invested heavily in making Excel a great tool for analyzing data in the spreadsheet, a vast amount of corporate data was locked away in relational databases—outside of the reach of applications like Excel. The challenge of working with disparate databases led to my involvement with Microsoft's Open Data Base Connectivity (ODBC) specification, a common call-level interface for issuing queries to and retrieving results from relational SQL databases. ODBC afforded me the opportunity to participate in various committees working toward an international standard for how applications could communicate with relational databases. Untold hours of work by industry leaders from different companies across the world, united in a common vision of data accessibility, cumulated in the approval

* Foreword by Michael Pizzo first published in *A First Look at ADO.NET and System.Xml v. 2.0* (Boston, MA: Addison-Wesley, 2004).

00FMADOR2.qxd 2/18/05 9:37 AM Page xxvi

of an extension to the ANSI/ISO SQL 92 specification known as SQL/CLI (SQL Call-Level Interface).

But before the ink was dry on the new standard, we were already thinking about how to take data access to the next level. By viewing a database as a set of services, all connected through a common interface, we hoped to provide common relational services over any type of store. The result was OLE-DB, which, along with the higher-level ActiveX Data Objects (ADO), became the core of Microsoft's Universal Data Access platform and the topic of my presentation in Sydney.

This reflection brought me to the sudden realization that I actually found accessing and working with data interesting. This was a somewhat disturbing discovery. Up until that point, I had always considered myself pretty "hip"; I had been active in sports, a letterman in track and field, seemingly fun at parties, and I loved the outdoors. Yet I feared that this recent discovery meant that, somewhere along the line, I had become a nerd.

But why not take an interest in data? Just about any application, when you get down to it, is about accessing, manipulating, or storing data of one type or another. Information (data) is what makes the Internet so powerful, and harnessing that power by building better ways not just to access but also to work with and manage that data is the key to harnessing that power.

Still, I thought to myself, this newly realized love for data was not something to bring up at a party or dwell overly on during a first date.

Since that flight, however, I've noticed that I'm not alone in my affliction. There are others equally passionate about enabling new and innovative ways to bring data to life. My coworkers, individuals within the standards community, people I meet at conferences, the gurus who assist others on newsgroups and mailing lists, and even the guy currently sitting across from me engrossed in his book on building data-oriented Web sites all do what they do because they share that same passion for data.

And so do the authors of this book.

I first met Mark Fussell when he was interviewing for Microsoft. It was immediately apparent that Mark was a fellow data-phile through his enthusiasm for finding ways to enrich how customers access and work with data. Understanding the incredible potential in XML's ability to model data in a human-readable, self-describing format, and the tools and opportunities this enabled, Microsoft had merged the XML team into the Data team responsible for more traditional relational data access, and Mark quickly took on a lead role in that team. In his three years at Microsoft his enthusiasm and customer focus have contributed significantly to the design and

00FMADOR2.qxd 2/18/05 9:37 AM Page xxvii

development of ADO.NET and the `System.Xml` classes that shipped in version 1.0 and beyond.

I first became acquainted with Alex Homer's name through the ADO.NET 1.0 beta newsgroups. ADO.NET 1.0 provided an unprecedented level of control to the developer. Rather than hide data access logic behind higher-level concepts, ADO.NET defines explicit interactions between connected objects optimized for accessing a database and an in-memory object specifically designed for working with a disconnected set of data. This new factoring, however, required a conceptual shift for a number of developers familiar with the previous ADO objects. It was interesting to watch the early beta newsgroups as developers caught on to the power of this new factoring and started evangelizing the model to others. Alex was one such evangelist who took an early interest in ADO.NET, ultimately writing one of the first books dedicated to the subject.

Dave Sussman interacts with customers daily as a consultant, trainer, and writer. He, too, has been working with the .NET Framework since before its release and shares the same enthusiasm and excitement over the Framework.

The benefits of these three authors' personal experiences, as well as those of the customers they represent, show through in this book's unique customer perspective of the evolution of the ADO.NET and XML features described in this book.

This is why, I realize now, I agreed to write this foreword. If these authors can share their insights into some of the exciting new features that comprise the next evolutionary step in our ability to access and interact with data, then maybe more people will discover a hidden passion for data. And who knows; perhaps in time I can even feel cool again.

But in the meantime, I wouldn't leave this book around during your next cocktail party.

By Soumitra Sengupta
Product Unit Manager
WebData XML Team
Microsoft Corporation

Mark Fussell demonstrates a special combination of attributes while taking the reader through the XML stack in the .NET Framework. He is a skilled practitioner who has a deep understanding of the design philosophy and goals (he should know as he was involved in the design and implementation of the core classes in System.Xml) and a remarkable determination to help

00FMADOR2.qxd 2/18/05 9:37 AM Page xxviii

other practitioners grasp how to take advantage of what he helped build. I am glad that Mark took the time to think about the challenges other software practitioners face when using the core XML stack in the .NET Framework. The result is the four chapters in this book that clearly explain the design goals and choices the product team made, with code examples that illustrate how best to use the API to build XML-enabled applications. For someone like me, who got a crash course in the .NET Framework after spending more than five years working with Java APIs, Mark's chapters speed up the learning curve faster than the specs that his team wrote. Don't get me wrong—the specs are pretty good too.

Use of XML is growing every day, and developers are using XML in new scenarios. It is not easy to write about a set of APIs that can be used in such diverse scenarios. However, Mark shows the same passion in these chapters as he did when leading the team that designed these APIs. His empathy for and deep understanding of the needs of practitioners come through as he takes the time to explain the scenarios that drove the design decisions and follows it up with sample code, which I believe will help both experienced developers and newcomers as well. Having read his chapters from beginning to end, I recommend that you do the same because the scenarios provide valuable context to the code samples Mark provides. I found it useful to annotate sections with additional comments drawing parallels to scenarios I am familiar with and then to revisit the code samples to see how they are different from the APIs in Java, which are more familiar to me.

I especially like Mark's focus on explaining the subtle differences between the version 1.x APIs and the upcoming version 2.0 APIs. He not only enumerates the differences in detail but also explains the reasons these changes were made. This can come only from someone who has spent countless hours talking to customers, designing the changes, and then justifying these decisions to the demanding internal Microsoft customers. For example, take the change in version 2.0 to `XmlReaderSettings` and `XmlSchemaSet` from `XmlValidatingReader` and `XmlSchemaCollection` in version 1.x. He explains the rationale for this change: to enhance performance, to extend validation over any `XmlReader` as opposed to just the `XmlTextReader`, and to enable validation of in-memory documents. I hope that when you get your hands on the code, your experience will vindicate the decisions Mark and the team made.

Finally, there is no substitute for learning by doing. Mark demonstrates that in the chapters he wrote for this book. I believe that all readers of this

00FMADOR2.qxd 2/18/05 9:37 AM Page xxix

book and real practitioners in the trenches developing applications on the .NET Framework will benefit from Mark's experience and his ability to communicate that well.