Delphi in Depth: ClientDataSets

by

Cary Jensen

Published by Jensen Data Systems, Inc., USA.

The code samples and example database depicted herein are fictitious, and were created for this book as copyrighted freeware to be used by this book's readers. The information and code samples contained or referred to in this book are provided as is, without any express, statutory, or implied warranties. Neither the author nor Jensen Data Systems, Inc. will be held liable in any way for any damages caused or alleged to be caused either directly or indirectly by this book or its contents, the code samples and example database, or any other related online content and/or files for any particular purpose.

ISBN-10: 1461008581 (print edition)
ISBN-13: 978-1461008583 (print edition)

Printed copies of this book are available for sale from companies listed at, and linked to from, http://www.JensenDataSystems.com/cdsbook. Any other download or sale outlet is likely to be illegal. This is not a free ebook - do not distribute it.

Project Editor: Loy Anderson
Contributing Technical Editors: Stephen Blas, Nick Hodges, and Jim Tierney
Cover Designer: Loy Anderson

For more information and links for purchasing this book, visit:
http://www.JensenDataSystems.com/cdsbook

Introduction

Before there was the ClientDataSet, Delphi's Borland Database Engine supported cached updates. When cached updates were enabled, changes to your database tables and editable queries were not written to the underlying database on a record-by-record basis. Instead, updates were delayed, permitting a group of changes to be applied to the underlying database in an all-or-none fashion (or even discarded altogether).

I wrote extensively about cached updates in the early days of Delphi, but over time, cached updates lost their appeal. There were two reasons for this. First, the Borland Database Engine became dated, and it's SQL counterpart, the Borland SQL Links for Windows became deprecated. The second is that cached updates became hopelessly broke, and this happened sometime around the release of Delphi 6.

Fortunately, there was a ready replacement — ClientDataSets. ClientDataSets supported all of the basic features of cached updates and much more. In addition to providing a mechanism for grouping updates, ClientDataSets also introduced in-memory management of data and indexes, the ability to persist data and restore it to its previous state, an intuitive interface for managing the cache, and a convenient mechanism for transferring a ClientDataSet's state between processes.

Originally ClientDataSets were only in the high-end versions of Delphi. Specifically, the Client/Server edition. Furthermore, the license to use a ClientDataSet in a distributed application was separate from Delphi, and was originally very expensive.

With the release of Kylix and subsequently Delphi 6, the ClientDataSet became available in the Professional version of Delphi, and the MIDAS (Multitier Distributed Application Services) license became affordable, and then eventually free. (The technology that was MIDAS is now called DataSnap.)

With ClientDataSets now within reach of all Delphi developers, I became an enthusiastic advocate for their use. Not only did I speak extensively about the virtues of ClientDataSets at developer conferences, I also published an extended series of articles on the Borland Developer Network detailing the use and capabilities of ClientDataSets.

At a recent Delphi conference at which I was presenting a talk on advanced ClientDataSet features, I had a revealing discussion with an attendee. His questions about an issue he was having with ClientDataSets demonstrated a fundamental misunderstanding about how they worked.

When I explained what was happening, he expressed frustration that this information was not more readily available. I directed him to my previous articles and commented, in passing, that maybe I should write a book about ClientDataSets. "I'll be the first one in line to buy it if you write that book" was his reply.

This idea has been with me for some time, and this past winter, I finally pulled myself together and started on this book in earnest.

Initially I thought that I could write this book by starting with some of my previous writings, including both the published series as well as a two-day Delphi course that I wrote and train called *Mastering ClientDataSets.* The trick, I reasoned, would be in the organization, and I had a lot of ideas about how I would go about this.

Projects like these tend to take on a life of their own, and once I was happy with the book's outline, I realized that I was committed to writing new material — a lot of new material. In the end, I estimate that at least 80 percent of this book is completely original, and that material that I salvaged from my previous writings often received extensive revision.

I am really happy about how this book turned out. I hope that you are, too.

This book takes an in-depth look at ClientDataSets. It begins with an overview of what a ClientDataSet is, and describes some of the ways that ClientDataSets can be used in your Delphi applications. It continues with a close examination of one of the more common uses for a ClientDataSet, reading and writing data from an underlying database through its interaction with a DataSetProvider.

Next you learn how to define the structure of a ClientDataSet, a process that can be undertaken both at design time as well as at runtime. The relative roles of FieldDefs and Fields are discussed, and the roles of persistent versus dynamic Fields are considered. How virtual Fields differ from data Fields is also explained.

Chapter 5 introduces indexes, including the difference between persistent and temporary indexes. The relationship between a ClientDataSet's indexes and those of an underlying database are also considered.

In Chapter 6, I take a long look at the change cache, the mechanism responsible for caching updates. Here you learn how to enable and disable the cache, how to detect changes to it, and how to modify the contents of the cache.

Chapters 7 through 9 introduce topics that will be familiar to many Delphi database developers, including how to edit, navigate, search, and filter data. These discussions, however, demonstrate the flexibility that ClientDataSets bring to these features, providing you with capabilities not available in other DataSets.

Chapters 10, 11, and 12 introduce some of the more interesting features found only in ClientDataSets. These features include aggregates, cloned cursors, and nested datasets.

The final three chapters of this book take a close look at DataSnap. These chapters specifically look at DataSnap applications where ClientDataSets in a client application interact with DataSetProviders running in a DataSnap server. Here you will learn to create DataSnap servers and clients using both the classic COM-based DataSnap as well as the newer IP-based DataSnap first introduced in Delphi 2009.

Consistency is important when writing a book. And while I know that there are always some inconsistencies that slip between the cracks, a lot of effort went into using terminology consistently in this book.

One of the areas that was most challenging in this respect was how to refer to objects. By convention, Delphi classes begin with an upper case T, and in most of my Delphi writings, I have carried that convention into the discussion of objects. For example, I might talk about ClientDataSet instances as TClientDataSets.

I did not adopt that convention in this book, however. I did not like how it looked to be talking about TClientDataSets. I though that referring to instances of the TClientDataSet class as ClientDataSets read better and looked more natural. As a result, when I refer to an instance of the class, I omitted the T. In those cases where I refer to the actually class declaration, I used the T.

While this approach worked well, for the most part, there are some places were it made the prose a little more difficult to read. This is especially true in the discussion of Fields, instances of classes that descend from TField. Frankly, those discussions, especially those in Chapter 5 dealing with ClientDataSet structure, were challenging to write clearly, since sometimes I was referring to a field (lower case, and referring to a column of data from a database table), sometimes to a Field (an instance of a TField descendant), and other times referring to a TField (the actual class definition).

There was another significant challenge in this book, besides how to refer to objects, and this was related to the code samples. In short, I felt that it was extremely important to provide code samples that made few assumptions about your Delphi installation, and required little or no additional configuration.

One place where this was especially difficult was in the selection of a database. Many, but not all, uses of a ClientDataSet involve interacting with a database. Where would the database come from, and what data access mechanism should be used to interact with that data?

One solution would have been to include a sample database in the code download. Loy Anderson and I created a sample database that we have used in other books, and we considered making it available for this book. But that would have required you to both install that database, as well as any data access mechanism it required. I considered going with a free solution such as MS SQL Server Developer Edition, InterBase Developer Edition, or even the Advantage Database Server, a remote database about which Loy Anderson and I have written three books.

The main drawback to all of these solutions was that it would require you to not only download the database, but also download a client API (application programming interface). In addition, additional configuration of the data access mechanism, such as an ODBC driver or an OLE DB Provider would be necessary, and all of that would have to be done successfully before you could run the first code example.

In the end, I opted to use the Borland Database Engine (BDE) and the sample Paradox database that is shipped with every copy of Delphi. While the BDE is obsolete, it provides what you need — a pre-configured database that requires no additional files or configuration on your Delphi machine. It also supports SQL, which was one feature in its favor.

When you install Delphi, the Borland Database Engine is installed by default, and a Database alias named DBDEMOS is inserted into the BDE configuration. Assuming that you have performed a typical installation of Delphi, everything should be ready for the code examples.

One aspect of the BDE's obsolescence is that it is not Windows 7 or Vista aware, and the added layers of security and user account control (UAC) might be an issue. This might require you to run Delphi with administrative privileges. Similarly, some of the Delphi executables that you compile from the code samples may also need to run with administrative privileges. If any of your executable's fail with an error related to reading the Paradox configuration file, PDOXUSRS.NET, it is likely an issue of privilege.

One more point about the code samples deserves mention. With the exception of a few of the IP-based DataSnap examples, which require Delphi 2009 or later, most if not all of these sample projects should run in almost any version of Delphi that supports ClientDataSets. The other exception is the DataSnap servers based on COM. Many of the early versions of Delphi had difficulty using the type library created with a different version of Delphi. If you run into that problem, you will need to follow the provided steps to create new versions of those COM servers.

Note: You can download the code samples form this book's web site.
http://www.jensendatasystems.com/cdsbook/

See Appendix A for details.

About the Author

Cary Jensen

Cary Jensen is Chief Technology Officer of Jensen Data Systems, Inc., a company that provides software services, software training, documentation, and help system development. Cary is an active developer, providing companies with assistance in data modeling, software architecture, software development, and software migration. He is an award-winning, best-selling author of over 20 books, including Advantage Database Server: A Developer's Guide, 2nd Edition, Building Kylix Applications, Oracle JDeveloper, JBuilder Essentials, Delphi In Depth, and Programming Paradox 5 for Windows. A frequent speaker at conferences, workshops, and seminars throughout much of the world, he is widely regarded for his self-effacing humor and practical approaches to complex issues. Cary is an author and speaker for the 2001-2010 Delphi Developer Days tours in the US and Europe (more recently, Delphi Developer Days 2009-2011 with fellow author and Delphi expert Marco Cantù), the 2003-2006 Advantage Developer Days tours, 2000-2001 Delphi Development Seminars, the 1999-2000 Borland Developer Days, and the 1995-1999 Delphi World Tours. Cary has a Ph.D. from Rice University in Human Factors Psychology, specializing in human-computer interaction.

You can follow Cary on twitter at:
http://twitter.com/caryjensen/

Cary also writes a blog, *Let's Get Technical,* in which he typically posts magazine-length articles on Delphi and other software-related topics:
http://caryjensen.blogspot.com/

For more information about Cary's company, please visit:
http://www.JensenDataSystems.com/