

Adobe Acrobat 5: The Professional User's Guide

DONNA L. BAKER

Apress™

Adobe Acrobat 5: The Professional User's Guide

Copyright © 2002 by Donna L. Baker

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN (pbk): 1-59059-023-6

Printed and bound in the United States of America 12345678910

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Editorial Directors: Dan Appleman, Peter Blackburn, Gary Cornell, Jason Gilmore, Karen Watterson, John Zukowski

Technical Reviewers: Susan Glinert, Carl Orthlieb

Managing Editor: Grace Wong

Copy Editor: Nicole LeClerc

Production Editor: Kari Brooks

Compositor: Susan Glinert

Indexer: Ron Strauss

Cover Designer: Tom Debolski

Marketing Manager: Stephanie Rodriguez

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States, phone 1-800-SPRINGER, email orders@springer-ny.com, or visit <http://www.springer-ny.com>.

Outside the United States, fax +49 6221 345229, email orders@springer.de, or visit <http://www.springer.de>.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710.

Phone 510-549-5930, fax: 510-549-5939, email info@apress.com, or visit <http://www.apress.com>.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

Chapter 12

Using Advanced Acrobat Activities

*This chapter contains an avalanche
of advanced activities that will
turn you into a power user.*

For All the Mighty Brains Out There

This chapter is different from many of the ones that came before. First, it doesn't cover any one specific topic. Second, the intent of this chapter is to show you some advanced ways of working with Acrobat. Some of these features are Web related; some are static. Some are functions; some are settings.

The idea is that by the time you reach this stage in the book, you should be familiar with many of the more basic and functional uses of Acrobat. This chapter's content will help to bump you to the power-user level.

In all honesty, I designed this chapter with me in mind. I have a need to be a power user for virtually any application I set my mighty brain to. So I look specifically for the types of activities that streamline my production cycle or file size. I also have a tendency to dig to the bottom of most functional areas. In this kind of book, I know in advance that I will likely cover the basics and advanced uses of some functions. I also know that I will dig too much and write too much. Because what I learn is too doggone good to keep to myself, I have to find a home for it. This is it.

This chapter, then, is dedicated to all of you who, with your own mighty brains, have a need to see what you can do with a piece of software.

What You Will Learn

As I mentioned, this chapter isn't something that is highly structured in terms of presentation. I have tried to make some sense of how I present things to you. Some of the topics are real discussions, complete with a demonstration, tutorials, and even a project. Other topics are fat Workflow Tips. Some may elicit the "Ah hah!" response. All are geared toward using Acrobat at a higher level of sophistication.

It Goes Like This

The topics in this chapter share one of two common themes. Either they are about a more advanced use of the software, or they are intended to discuss or show you how to use the software more efficiently. Enhanced efficiency means an advanced use of a piece of software, in my opinion. Here are the topics that are coming up:

- Using PDF files versus other file formats
- Working with video files (a tutorial)

- Working with destinations (discussion and project)
- Adding date stamps (a tutorial)
- Using byte-serving files on the Web
- Reducing file size

Up first is when and how to use PDF files.

Using PDFs vs. Other Media Formats

What's one of the key benefits of PDF documents? Rhetorical question. PDF formats are especially conducive to producing documents that are both cross-platform and cross-media.

What does this mean? Basically, we are looking at a format or collection process that will allow information to be used as text and graphics; for multimedia; for use on kiosks and in presentations; and for printing, online and offline. Most of these things can be done with Acrobat. But does that mean they *should* be done?

Your Basic Big Multimedia Piece

Technically, I am referring to project architecture. Consider the types of material you deal with on a regular basis: software installations, marketing pieces, learning materials . . . The list is endless. Most of these outputs include common structures. How can PDF files be added to the project, how should they be added, and what are the criteria for their use? Read on.

A large multimedia project includes these common elements:

- *Installation:* Installation is not a topic of Acrobat conversation.
- *Introduction:* Many CD pieces launch with an autorun introduction. This is usually highly visual, active content.
- *Navigation:* In addition to the ubiquitous navigation panel, there are many other navigation options—for example, thumbnailing images or video clips, or special panels to control video and talking heads.

- *Content:* The reason for opening the CD in the first place, of course, is content, and it takes virtually unlimited forms. The only comment I would like to make here is that you should consider what is required for the user to use your content. That is, will other plug-ins or extensions be required? This can be a determining factor when you're deciding on one format over another.

Let's see when to use one format or the other.

The Power of PDFs

Many times a PDF file will be the ideal medium for distribution of your work. For example:

- *When you're printing:* You can't beat the PDF format for reliable printed output.
- *When you're using source material from a variety of different authoring programs:* PDFs can be created from virtually any program that can print to Distiller or a PostScript printer.
- *When you're using cross-platform files:* PDFs work well on ISO 9660 cross-platform formatted CD-ROMs.
- *For security reasons:* PDFs are easily saved with password encryption settings.
- *For interactivity purposes:* Acrobat readily handles different types of actions smoothly and simply. These actions can be attached to a variety of handlers, including links and pages. I provide a project for adding multimedia later in this chapter.

The Power of Multimedia

Adobe Premiere and Macromedia Director are two of the big guys used for designing multimedia presentations. Depending on the structure of a piece, you may likely use both programs. Use these types of programs when the following issues are important:

- *Audio and video synchronization:* Depending on the sophistication, you will have much better results (use Acrobat to embed files that launch on specific actions only).
- *Lingo programming (Macromedia Director):* Use Director to program interactive navigation controls and other elements. Again, this depends on the level of sophistication you require.
- *Animation:* I need say no more.

CD-ROM Distribution

In other areas both of this chapter and this book, I have discussed using the Web as a medium for delivering PDFs. That is not the only option for how to deliver PDFs, and sometimes it's not even the best option. In several situations, it might be better to publish to CD-ROM. Consider these ideas:

- Suppose you have to deliver large files, movies, executable files, and the like. Internet delivery is not efficient; CD delivery is. On the same topic, Internet and/or intranet speeds may be too slow to deliver efficient and quick navigation tools and information.
- How are your users accessing material? If they ordinarily work offline, a CD is much more convenient than finding a laptop connection.
- For highly secure documents, burning a specific number of CDs for distribution will help to control access to your information.

Project Tutorial: Adding Video to PDFs Efficiently

I have neglected to show you any of the groovy things you can do with Acrobat in deference to the work-based, serious things. Time to change that. And it's a good topic to follow multimedia discussion.

Inserting a Movie into a PDF Document

You can add two types of video files to Acrobat, as you can with many applications: AVI and MOV. I will discuss how to add a movie to a PDF file and offer some tips for using video file formats.

There is a file on the CD in the Chapter 12 Projects folder named `movie.pdf`. This file, it will come as no surprise to you, is the basic framework to which you will add a movie (shown in Figure 12-1). You must insert the movie yourself to ensure that the storage locations for the PDF file and the MOV file are the same.

Note

You will require QuickTime to complete this tutorial. It is available for download at <http://www.apple.com/quicktime/download/>.

Note

The movie is a nursery rhyme—the one about the little girl with a little curl. I have no idea where the inspiration came from. Perhaps a recent shopping trip with my own little girl?



Figure 12-1
The finished document
displaying the movie

1. Open the movie.pdf file. Click the Movie tool on the Editing toolbar to select it, and then click the document page. The dialog box shown in Figure 12-2 will open.

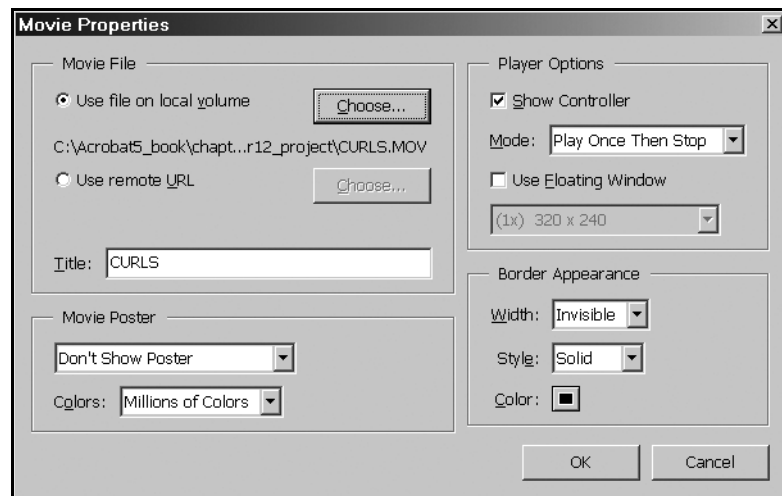


Figure 12-2
The Movie Properties
dialog box

2. Click Choose, and browse for the file to use, named CURLS.MOV (as I said, to make the file run properly on your computer, you will have to attach the file yourself).
3. In the bottom-left portion of the dialog box, Movie Poster, you can select options for the placement structure of the movie on the page. In my example, I had created the file with a window for the movie. If you are adding a movie to a document that doesn't have any placement outlines built into it, select a poster option from the drop-down list. It will display a white box the size of the movie, as shown in Figure 12-3. You can also select two color options in the same area. There is no difference in file size between the two different color options (256 versus millions of colors).

Note

You can also set border options on the right side of the dialog box to further frame out your placement guide.

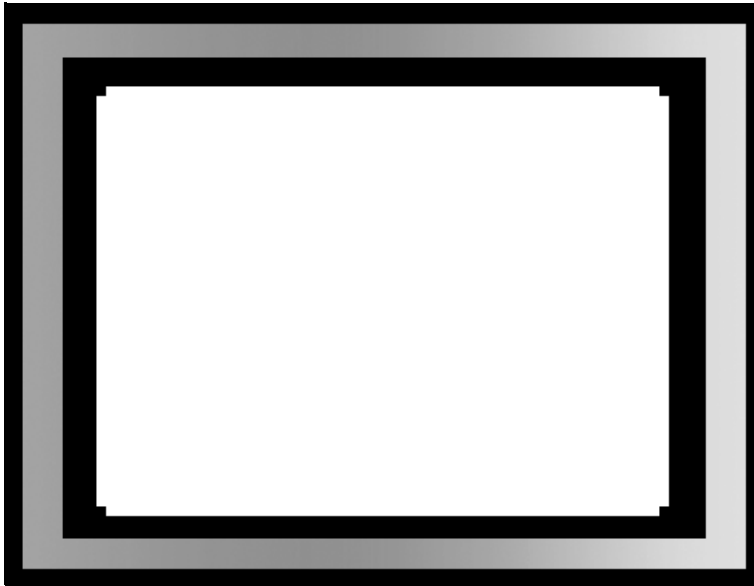


Figure 12-3
A portion of the document showing the Movie Poster placement

4. Select player options from the Mode drop-down list, as shown in Figure 12-4. There are a number of different settings depending on your use. If you leave the Show Controller option selected, the user can manipulate the movie in any case. You may also choose the floating window option—when the movie area is clicked on with the mouse, a movie player (set to one of the size options) will appear on the page.

Figure 12-4
Select a mode
for the controller.



Workflow Tip

For the Premiere Fans Out There . . .

. . . from a Premiere fan over here. I always like to know what and how a file is composed. I have this movie optimized for desktop/kiosk delivery on midrange machines. It is in QuickTime format. I used a Sorenson Video Compression codec rendered at 30fps. The frame size was 320x240pi.

5. Save the file. Deselect the Movie tool by clicking the Hand tool. Move your mouse over the movie area of the page. You will see the hand icon change to a movie icon. Click to run the movie.

So far we have looked at multimedia issues. Up next is the first of three topics dealing with time: how to save it, how to display it, and how to serve it.

Working with Destinations

A *destination* is a link represented by text in the Destinations palette. If you look at the Acrobat help files, for example, you will find that the Destinations palette has no entries. This is because the help files are all in one document.

Creating and Linking Destinations

Before destinations can be linked, they have to be created and named in the place where they are being housed (the source document). This process can be complex. In order to make using destinations more understandable, I have built a project. But first some general discussion on how Destinations work.

For these general instructions, you must have the Destinations palette open. At the lower part of the palette, you will see a message that reads “Document Not Scanned.” You must scan the document each time you open the file when working with destinations. It’s akin to setting baselines. To scan a document, open the file, and select Destination ► Scan Document from the palette’s drop-down menu. Once the scan is complete, any destinations in the document will be listed in the palette. If you have no destinations, the message that was previously displayed at the bottom of the palette will disappear.

Using destinations is a three-stage process involving a minimum of two different documents: the source document (where the links are created) and the target document (which houses the links to the source document). Follow these steps:

1. Open the target document (the document that will be linked to the source). Set the desired location and view.
2. Select Destinations ► New Destination in the Destination palette, or click the Create New Destination button at the top of the palette.
3. Name the destination, and press Enter (Return on a Mac).
4. Now open the source document. Select the Link tool.
5. Click and drag a rectangle to define a link location.
6. Set the options for the link:
 - a. Select Go To View as the action type. (This action makes a link to a destination.)
 - b. Select a magnification option.
7. With both documents open, and the Link dialog box still open, switch to the target document and select the destination to be linked. When the page is displayed, click Set Link.
8. The Link dialog box will display the file name and selected destination name from the target document.

The link is complete. Next is a project that shows how to use these destination-building ideas.

Project 12-1. Adding Destinations to the Makeup Manual

In Chapter 3, I introduced the Manual for Applying Stage Makeup project. In this project, I have made a document collection from the manual's materials. This project has two phases. First, the smaller documents will be attached to the table of contents of the main manual document. In the second phase, elements within the text of the main manual will be linked to elements in the glossary.

Workflow Tip **Why Use Destinations Instead of Regular Links?**

The name of the game is efficiency. If you added normal links to pages across documents, the links will be broken if pages are added or deleted from the target document. Destinations, which reside in source documents, are not affected by changes in target documents. It might take more time to initially create destinations. In the long run, however, they can save you lots of time.

Workflow Tip **Keeping Destinations on the Straight and Narrow**

Two points to remember. First, the destination names must be unique. Second, decide on a naming convention and stick to it. When working in the target document, or troubleshooting any errors, you will be able to understand the structure by looking at it in the source document's Destinations palette.

The CD contains a set of four “raw” files in the Chapter 12 Projects folder. These files are as follows:

- *manual.pdf*: The main manual document.
- *biblio.pdf*: A bibliography for the manual.
- *FAQ.pdf*: Guess what? A FAQ file.
- *gloss.pdf*: A glossary of terms.

The CD also includes a second set of files, named the same, but each file’s name includes “1,” as in “manual1.pdf.” These four files together make up the completed project.

You will need the Destinations palette open. Also, select Destinations ► Scan Document every time you open one of the files.

Part 1: Linking the Accessory Files to the Makeup Manual

First, the extra files will be connected to the main manual’s table of contents. The portion of the table of contents used for the destinations is shown in Figure 12-5.

<i>Glossary</i>	<i>x</i>
<i>Frequently Asked Questions</i>	<i>y</i>
<i>Bibliography</i>	<i>z</i>

Figure 12-5
The table of contents will have files attached to these entries.

Add these elements.

1. Open the first target document: *gloss.pdf*.
2. Set the magnification for the glossary. Select View ► Actual Size.
3. Select Destinations ► New Destination.
4. As shown in Figure 12-6, name the new destination *glossary*, and press Enter (Return on a Mac).



Figure 12-6
Add and name a destination in the target document.

5. Repeat this process for the other two target files:
 - a. *biblio.pdf*: Name the destination bibliography.
 - b. *FAQ.pdf*: Name the destination FAQ.
6. Don't forget to set the magnification to actual size.
7. Save the files.

Build the Links in the Source

Now we have to connect the target destinations to the manual.

1. Open *manual.pdf*.
2. Set the view mode in the bottom taskbar to Single Page.
3. Add the first link with these properties:
 - a. Click the Link tool, and then click and drag a rectangle around the *x* page number for the Glossary row.
 - b. Select Invisible as the link appearance.
 - c. Select Go To View as the action type.
 - d. Select a magnification option.
4. Leave the Link Properties palette open as it is. Now open the *gloss.pdf* document (if it isn't open).
5. In the *gloss.pdf* file, open the Destinations palette, select Scan Document, and then click the glossary destination we set earlier.
6. As you can see in Figure 12-7, which shows the lower portion of the Link Properties dialog box, the file name and the named destination are now displayed in the Link Properties palette from the *manual.pdf* document.

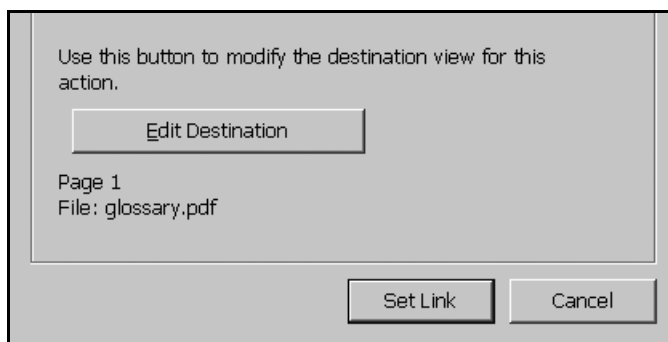


Figure 12-7
The external link has now been identified.

7. Click Set Link. The view will switch back to the manual.pdf document. Save the file and test the link.
8. Repeat the process with the destinations for the other two files.
9. Save the manual.pdf file.

Now for Part 2. Adding one destination per file is easy. What if a list of destinations is required for a list of targets, all from the same files? Read (and work) on.

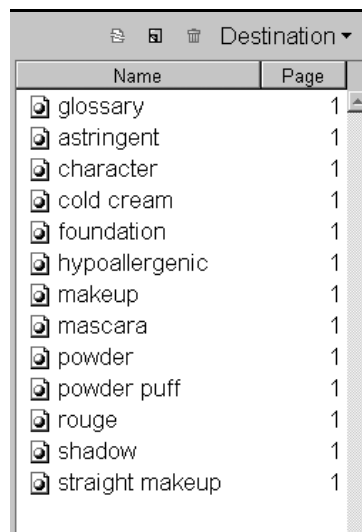
Part 2: Linking Glossary Definitions to the Manual's Pages

The stage makeup manual has a total of 14 pages. Sprinkled throughout are a number of terms that are defined in the glossary. We will set definitions for the glossary terms and then set links from the terms in the manual document.

Start with the Glossary

As the heading states, start with the glossary. This is the target document, so the destinations have to be set here first.

1. Open gloss.pdf. Select Destination ► Scan Document. The glossary destination set earlier should be displayed.
2. Set the destinations. Name each destination according to the glossary term. There will be 13 terms and the destination for the page itself, as you can see in Figure 12-8.



Name	Page
glossary	1
astringent	1
character	1
cold cream	1
foundation	1
hypoallergenic	1
makeup	1
mascara	1
powder	1
powder puff	1
rouge	1
shadow	1
straight makeup	1

Figure 12-8
The completed destinations for the glossary file

- Set the view for the destinations with the named term displayed the full width of the window and the term at the upper portion of the window (this will not work with the terms closer to the bottom of the page). I have shown one destination, “character,” in Figure 12-9.

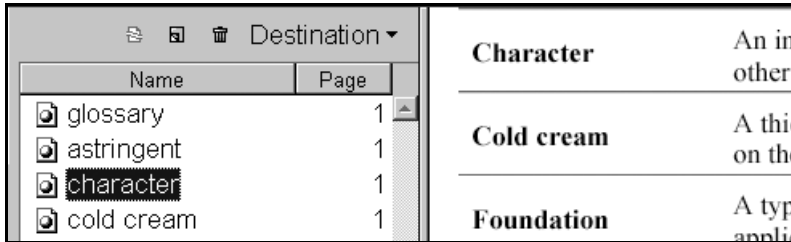


Figure 12-9
Set the view for the destinations similar to this.

- Save the gloss.pdf file.
- Open the manual.pdf document. Glossary terms are usually identified in some way based on their first occurrence. For this reason, I have listed the location where each term first appears in the manual in Table 12-1.

Note

The page numbers listed in the table are the actual page numbers of the document, not the pagination assignment.

Table 12-1 Link the Destinations from the Glossary to These Areas of the Manual

Glossary Term	Location
astringent	Page 8, second paragraph, first line
character	Page 3, last paragraph, first line
cold cream	Page 6, second row of top table
foundation	Page 7, first row of table
hypoallergenic	Page 5, second paragraph, last line
makeup	Page 3, first sentence of first paragraph
mascara	Page 7, second-to-last row of table
powder	Page 7, second row of table
powder puff	Page 13, second bullet in right column at bottom of page
rouge	Page 5, under brush heading, flat brushes listing
shadow	Page 10, lower heading
straight makeup	Page 3, first sentence of first paragraph

6. Using the information in the table, add links to the locations listed in the table. Use the same link properties and process described in the last section.
7. Save the manual.pdf file.

Part 3: Because I Can't Stand Things That Aren't Finished Properly

I simply could not leave this project without making it more convenient to use. It is far too irksome. On that note, we will next add one simple field to each glossary term that will return the user to the previous view. A portion of the finished glossary file is shown in Figure 12-10.

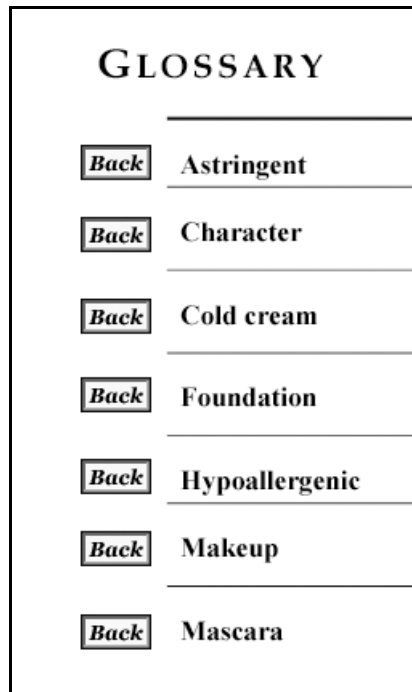


Figure 12-10
Part of the finished glossary file with back buttons added to the terms

Here's how to do it.

1. Insert the first form field. In the Field Properties dialog box, name the field **Back** and set its type as Button. Set these appearance characteristics:
 - a. Border color: dark red
 - b. Background color: custom color, RGB values of 255/251/234

- c. Border width: thin; border style: inset
 - d. Text: black, Georgia Bold Italic, size 9pt. (or substitute a similar font)
2. Click the Options tab. Set the layout as text only, and add **Back** as text for the button face in the Up position.
 3. Click the Actions tab. Select the Mouse Down state in the action list. Click Add and select Open File.
 4. When the Add An Action dialog box opens, select Open File from the type drop-down box. Click Select File and select manual.pdf. Click Set Action.
 5. When you return to the Field Properties dialog box, click OK to close the dialog box. The first button is built.
 6. Copy the form element down the page corresponding to each glossary entry. Align the buttons. I set the buttons to be roughly centered vertically with the glossary terms.
 7. Save and test the file.

Isn't that better? One last detail. Just as the Back buttons were required for the glossary terms to return to the manual, you need to add the same button to the bibliography and FAQ pages. You can add a button to the glossary page itself to make the sequence consistent. I made the same button in a larger size and changed the label, as you can see in Figure 12-11.

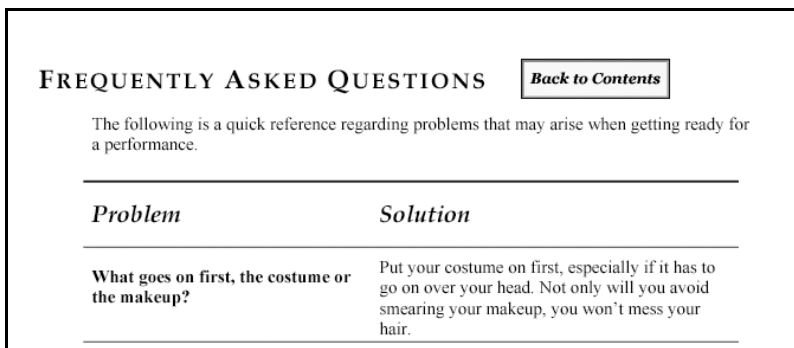


Figure 12-11
The last detail: adding navigation back from the documents to the collection

The project is finished. This is a good way to use a simple form field to make document navigation a simpler task.

Speaking of form fields, what about using a field on a document (it doesn't have to be a form) to add a date automatically? That's coming up next.

Project Tutorial: Date Stamping

One of the really interesting features you can add to a form, or any other kind of document, is a date stamp. We will look at how to add a date to a form when it is opened. To complete this tutorial, you will need the form.pdf file, which is on the CD in the Chapter 12 Projects folder.

It all hinges on a single form field. The form field will use the current date from the computer clock, and the date will be added when the form is opened. Here's how it works:

1. Add a new text form field anywhere on the document. Name it **Date**. On the Appearance tab, select Read Only under the Common Properties section. Deselect any background or border color options.
2. Click the Format tab. Select Date from the options, and choose one of the format options on the right side of the tab, as shown in Figure 12-12. Click OK, and save the file.

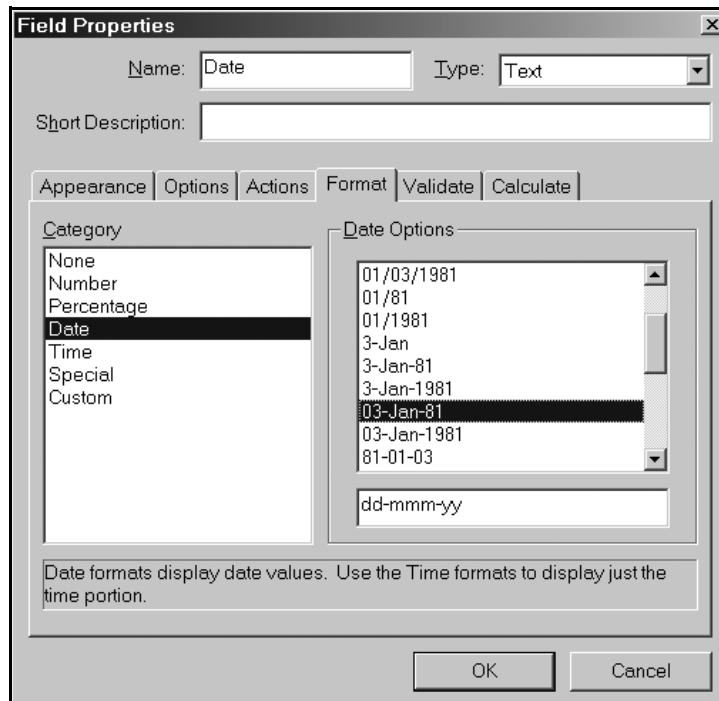


Figure 12-12

Set these date options on the Format tab of the Field Properties dialog box.

- Now we need a document-level script that will execute each time the document opens. Select Tools ► JavaScript ► Document JavaScripts. Name the script **Date** and click Add.
- Enter the following script in the JavaScript Edit dialog box within the JavaScript Functions dialog box, as shown in Figure 12-13:

```
var d = this.getField("Date");
d.value = util.printd("dd/mmm/yy", new Date());
```



Note

When the JavaScript Functions dialog box opens, some text will be present. Select and delete it.

Note

This script is made up of several parts. The Date field is bound to the variable, d, which is then calculated. The new expression is based on the date. The utility object (`util.printd`) formats the date into the format selected.

Figure 12-13
Add a document-level script.

- Click OK in the JavaScript Edit dialog box, and then click Close in the JavaScript Functions dialog box.
- Save the file. If you deselect the Form tool, you will see that the field now contains the date, as shown in Figure 12-14.

To view the date, select the Hand tool from the Acrobat toolbar; the date will be displayed in your new field.

Note

Be careful with this process. The field name must be exactly the same (including capitalization) in the Field Properties dialog box and the script; the same date format must be used in the Field properties dialog box and the script. If you receive errors when you try this, check these elements.

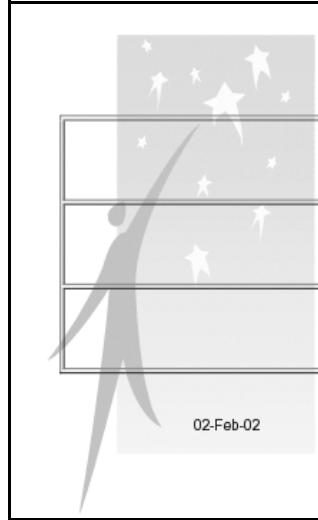


Figure 12-14
A portion of a form
showing the automatic
date entry

Variations on This Theme

You can also add different date or time formats to your form. Change the visibility on the Appearance tab to Hidden, for example, to hide the date field from view. When the file is printed, it will be displayed. As you can see in Figure 12-15, you have several options.

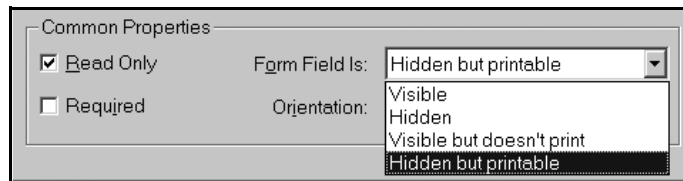
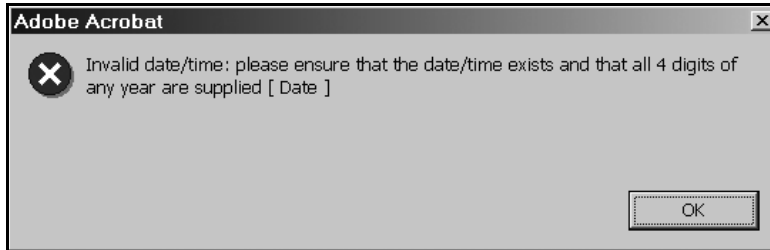


Figure 12-15
Different visibility
options for form fields

Using JavaScript in Acrobat Documents

As you have seen in several chapters, particularly with regard to forms, there are many ways to use Adobe JavaScript to configure and customize your work. There are just as many ways to botch it up. The date stamp discussion is a case in point. The process seems so simple: Configure the field and its content and then write a script to input the date. The following image shows what can happen, though.



Invalid date and time error

Well, just what does “invalid date/time” mean? Date and time certainly does exist. I know this. My system clock is running, the clock on my wall is running. So why shouldn’t a date and time exist? After considering and dismissing the thought that I had finally made it into the Twilight Zone, I discovered the scripted format differed from that selected in the Form dialog box, as you can see in the following image.



Make sure the JavaScript is the same as the Field Properties chosen.

The key is to be careful. When you receive a message, consider the obvious—such as punctuation and descriptions—before science fiction events.

And now for the last of our series of time issues.

Byte-Serving PDF Files on the Web

Byte-serving starts the view of a file before the entire file is downloaded, one page at a time by default. Depending on the Internet connection speed and the size of the file in question, unless the download is served in chunks, it can take a l-o-o-n-g time to serve an entire document—long enough, of course, for your reader to get bored and surf away.

Preparing the Files

Files should be cleaned, optimized, and saved with the Fast Web View options. Once a document is completed, save a version to be used for online viewing.

Check your preference settings. Figure 12-16 shows the preference settings that will optimize browser display. Select Edit ► Preferences ► General. Click Options. As you can see, I have selected all four of the Web Browser Options.

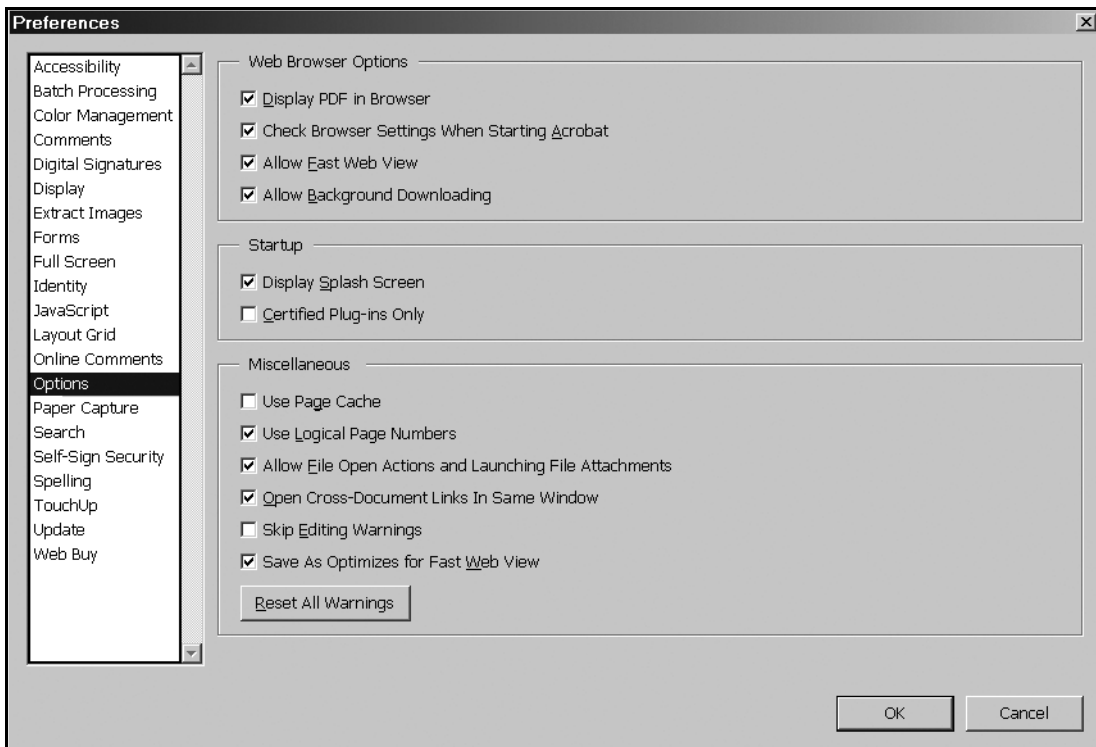


Figure 12-16
Select all of the Web Browser Options in this dialog box

While you are in this file-cleaning mode, check for the features that can be deleted from the Web-served version. Remove bookmarks, articles, and comments. This makes the file much smaller and, because these features are likely invisible to the average user, why include them?

That completes my trio of time-centric topics. For the grand finale, some tips.

Reducing File Size

Sometimes it doesn't matter how big a PDF file is. On the other hand, sometimes it is critical. Consider online files, or ones that are being shared in a workgroup. Although some elements must remain, a good understanding of how to use Acrobat will go a long way toward making file sizes more efficient. Check for these problems before throwing your hands up in disgust or moving to an isolated location without Internet access or indoor plumbing.

Here is a collection of tips for reducing the size of your files:

- Embed subsets of fonts rather than the entire font. To be safe, you can set the percentage at which the entire font is embedded.
- Watch out for crop marks and prepress information. Depending on the authoring application you use for creating source files, items such as color bars or crop marks may still be present in the converted file. Remove any of these items before converting the file to PDF.
- Consider custom PDF creators. As an experiment, export a source document using different options. If you are using a program with a custom PDF exporting utility, test it. Also create a version using Acrobat Distiller and/or PDFMaker. Check the sizes of the different versions. You may be surprised.
- Watch those graphics. Compress objects through Distiller job options. Whenever possible, use vector graphics over bitmaps. Base text on fonts, not bitmaps. Check for downsampling.
- Periodically use the Save As function instead of Save. You can use the same name and overwrite the original file. Using Save As will remove deleted objects, optimize the file, and store identical items like backgrounds, which can make an enormous difference in your file size. For example, while constructing my movie.pdf file, I saved it several times, for a final size of 61.6KB. When I used Save As, it dropped in size to 39.5KB—a drop of over 40 percent.

Workflow Tip

Beware of the Optimize Space Function

The PDF Consultant option for optimizing space analyzes the open document. When you select the Remove Unused Named Destinations option, the program doesn't check to see if the destinations in the open document are used by links in other files. In a document collection, destinations to links in other files will become invalid. Use this function only on single files with no cross-PDF links.

- Save file size for buttons by attaching actions to pages instead. Selecting Document ► Set Page Action opens the dialog box shown in Figure 12-17. The interface is the same as that in the Field Properties dialog box for mouse states.

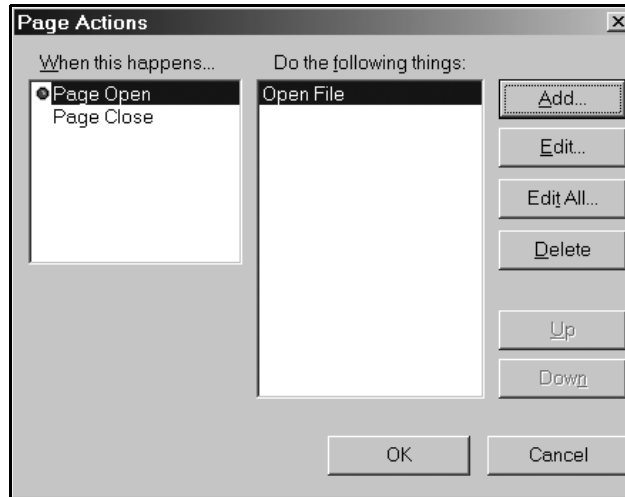


Figure 12-17
Set a page action.

- As you can see in Figure 12-18, choosing one of these actions opens the same dialog box as for other action types.

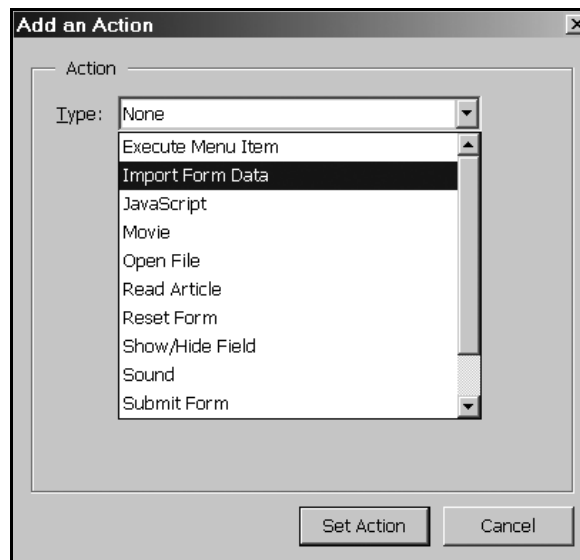


Figure 12-18
Add a page action.

- Check the named destinations set in a file. Each destination is about 100 bytes. If you are using many paragraphs, such as with an index or tables, much of the file size can be attributed to destinations.

Up Next

I introduced this chapter as a profusion of different ideas, techniques, and functions. Some of the topics were extensions of previous chapters' content or new timesaving topics. Still others presented ideas and issues rather than any specific technique.

As I mentioned, many of the concepts presented in this chapter were aimed at taking your use of Acrobat to the power-user level. Although this book isn't the forum for exhaustive discussion of any one topic, I think I gave you enough information to both stimulate your imagination and open other avenues of exploration.

And now for something completely different. Unlike this chapter, the next one is very specific and most intriguing—all about e-books.

Index

Symbols and Numbers

- \ (backslash) in Active Directory hierarchies, 420
- * (asterisks), advisory about, 434
- < (less than) column comparison symbol, role in Query Designer join types, 323
- <= (less than or equal to) column comparison symbol, role in Query Designer join types, 323
- <> (not equal to) column comparison symbol, role in Query Designer join types, 323
- = (equal to) column comparison symbol, role in Query Designer join types, 323
- == operator, using in ADO.NET connected layer, 87
- > (greater than) column comparison symbol, role in Query Designer join types, 323
- >= (greater than or equal to) column comparison symbol, role in Query Designer join types, 323
- 1NF (First Normal Form)
 - achieving, 47
 - checking against UserMan database schema, 53
- 2NF (Second Normal Form)
 - checking against UserMan database schema, 53
 - conforming to, 48
- 3NF (Third Normal Form)
 - achieving, 48–49
 - checking against UserMan database schema, 53
- 4NF (Fourth Normal Form)
 - checking against UserMan database schema, 53
 - conforming to, 49–50
- 5NF (Fifth Normal Form)
 - achieving, 51
 - checking against UserMan database schema, 53

7304 and 7309 error messages, troubleshooting in Microsoft Exchange Server 2000, 445

A

- Aborted** member of **MessageQueueTransactionStatus** Enum, description of, 483
- abstract** keyword in C#, relationship to OOP, 510–512
- AcceptChanges**
 - DataRow** class property, 246
 - DataSet** class method, 199, 215
 - DataTable** class method, 221
- AcceptChangesDuringFill** DataAdapter class property, description of, 172
- access control, securing message queuing with, 499–502
- accessibility of databases, understanding, 32
- ACL (Access Control List), role in securing message queuing, 499–500
- Active Directory
 - accessing programmatically, 415
 - accessing with OLE DB .NET Data Provider, 429–436
 - binding to objects in, 421–422
 - checking for existence of properties with, 426
 - editing exiting properties with, 427
 - examining **System.DirectoryServices** namespace in, 415–417
 - exploring, 414–415
 - identifying objects in, 420
 - introduction to, 413
 - manipulating object property values with, 426–429
 - manipulating user properties with, 428–429
 - retrieving SID for users with, 435–436
 - searching for objects of specific classes in, 423–424
 - searching objects in, 422–425

- Active Directory (*continued*)
 - specifying information for retrieval with OLE DB .NET data providers, 431–432
 - studying **DirectoryEntry** class in, 417–422
 - updating, 429
 - updating cache after making changes to, 426
 - validating users with, 585
 - Web site for, 415, 434
- Active Directory class, calling for
 - UserMan example application, 581
- Active Directory connections
 - specifying domains for LDAP access, 431
 - specifying OLE DB providers for, 430–431
- Active Directory nodes or objects,
 - returning nondefault properties from, 424–425
- Active Directory objects
 - adding to UserMan example application, 579–581
 - updating, 435
- Add Database Reference dialog box, displaying, 297–298
- Add Server dialog box, displaying, 286
- Add Table dialog box, displaying in
 - Database Designer, 307
- AddNew** method
 - of **CurrencyManager** class, 540
 - of **DataView** class, 240
- Addr or Address **ConnectionString**
 - property value, details of, 66–67
- Admin Queue, receiving message
 - rejection notification in, 493–494
- ADName column in tblUser column for
 - CUser class, details of, 514, 516
- ADO (Active Data Objects)
 - accessing ODBC data sources in, 59
 - explanation of, 55
 - resources for, 56
 - using with ADO.NET disconnected layer, 276–278
 - versus ADO.NET, 56, 59
- ADO **Recordset** classes, populating
 - DataSet** objects with, 205–206
- ADO.NET (Active Data Objects.NET)
 - versus ADO, 56, 59
 - resources for, 57
 - role of data-related namespaces in, 58–59
- ADO.NET connected layer
 - clearing OLE DB object pooling in, 93–94
 - closing connections in, 86
 - command class methods in, 138–139
 - command class properties in, 129–137
 - comparing connection objects in, 87–88
 - comparing **State** to **ConnectionState** in, 89
 - comparing with **ReferenceEquals** method in, 88
 - data providers and drivers in, 59–63
 - data providers in, 59
 - data-related namespaces in, 58–59
 - declaring and instantiating
 - XmlReader** objects in, 168
 - defining transaction boundaries in, 99
 - disabling ODBC connection pooling in, 94–96
 - disabling OLE DB connection pooling in, 93
 - disposing of connections in, 86–87
 - DNs and RDNs in, 420
 - drivers in, 59
 - executing commands in, 139–142
 - handling Command class exceptions in, 143–146
 - handling Connection class
 - exceptions in, 110–123
 - handling connection state changes in, 89–90
 - handling fill operation errors in, 184–185
 - handling messages in, 97–98
 - handling row updates in, 185–188
 - introduction to, 55
 - manipulating connection state in, 88–96
 - opening connections in, 85–86
 - pooling connections for
 - SqlConnection**, 92–93
 - pooling connections in, 90–96
 - role of Connection class in, 63–64
 - role of **ConnectionString** property in, 65–76
 - role of transactions in, 98
 - setting command properties in, 178–184
 - support for manual transactions, 99
 - using == operator in, 87
 - using command objects in, 123
 - using CommandBuilder class in, 188–191

- using Connection class events with, 85–86
- using Connection class methods with, 80–84
- using Connection class properties with, 76–80
- using DataReader class with, 146
- using **Equals** method in, 87–88
- using members of **ConnectionState** Enum with, 89
- ADO.NET disconnected layer
 - introduction to, 193
 - role of data source locking in, 265–276
 - using classic ADO and COM Interop with, 276–278
 - using cursors in, 261–265
 - using **DataColumn** class in, 249–252
 - using **DataRelation** class in, 252–261
 - using **DataRow** class in, 245–249
 - using **DataSet** class with, 193–194
 - using **DataTable** class with, 218–223
 - using **DataView** class in, 238–245
 - using XML with **DataSet** class in, 195–196
- ADSID column in tblUser column for CUser class, details of, 514
- adsPath, retrieving from Active Directory with OLE DB .NET data providers, 432
- AFTER triggers, functionality of, 406
- aggregated data, using views for, 398
- Alias grid pane column in Query Designer, 326
- AllowDBNull DataColumn** class property, description of, 250
- AllowDelete DataView** class property, description of, 239
- AllowEdit DataView** class property, description of, 239
- AllowNew DataView** class property, description of, 239
- Alt+F11 keyboard shortcut, accessing Macros IDE with, 22
- anomalies in normal forms, explanation of, 46
- Antechinus C# Programming Editor, Web site for, 15
- Append grid pane column in Query Designer, 326
- Application Name **ConnectionString** property value, details of, 66–67
- ApplicationException** class, explanation of, 356
- ApplyDefaultSort DataView** class property, description of, 239
- Apress Web site
 - accessing, 57
 - downloading UserMan classes from, 567
- ArgumentException** exception, 351
 - handling for DataReader, 160
 - throwing, 113
 - thrown by **CommandTimeout** property, 143
 - thrown by **CommandType** property, 144
 - thrown by **UpdatedRowSource** property, 145–146
- ArgumentNullException** exception, 351, 353
- ArgumentOutOfRangeException** exception, 170, 351, 354
- arguments
 - advisory about usage with **SqlXmlAdapter** and **SqlXmlCommand** classes, 617
 - creating stored procedures with, 382–384
 - using stored procedures with, 384–385
- arguments and return values, running stored procedures with, 385–387
- ASP (Active Server Pages), evolution of, 5
- ASP.NET data binding, maintaining state in, 548–549
- ASP.NET, explanation of, 5
- ASP.NET server controls, binding to data sources, 545–548
- assemblies
 - functionality of, 9–11
 - interaction of namespaces with, 13
- assembly attributes, customizing, 11
- Assert Debug** class method, 363
- assertions
 - definition of, 360
 - using with **Debug** class, 362–364
- asterisks (*), advisory about, 434
- AttachDBFilename **ConnectionString** property value, details of, 66–67
- AttributeCount XmlReader** class property, 162
- auditing, securing message queuing with, 502
- authenticated messages, sending, 494–495
- authentication, securing message queuing with, 490–495

- AuthenticationType** property of **DirectoryEntry** class, description of, 417
- AUTO SELECT . . . FOR XML statement keyword, description of, 588
- AutoIncrement DataColumn** class property, description of, 250
- AutoIncrementSeed DataColumn** class property, description of, 250
- AutoIncrementStep DataColumn** class property, description of, 250
- automatic garbage collection, role in .NET Framework, 8
- automatic transactions
 - explanation of, 110
 - use of, 586
- B**
- backslash (\) in Active Directory hierarchies, 420
- BaseUri XmlReader** class property, 162
- BEFORE triggers, functionality of, 406
- Begin** method of **Transaction** class, description of, 109
- BeginEdit DataRow** class property, description of, 246
- BeginLoadData DataTable** class method, description of, 221
- BeginTransaction** method
 - exception handling, 111–112
 - of **OdbcConnection** class, 83, 101
 - of **OleDbConnection** class, 81
 - of **SqlConnection** class, 80, 100–101
- BINARY BASE64 SELECT . . . FOR XML statement keyword, description of, 588
- binding context, examining for Windows forms, 527–528
- BindingContext** object, role in Windows forms, 527–528, 538–539
- Bindings** property of **CurrencyManager** class, description of, 540
- Body setting for message queue encryption, explanation of, 497–498
- BOF (Beginning-Of-File), moving cursors to, 263–264
- Broken** member of **ConnectionState** Enum, description of, 89
- broken ownership chain, explanation of, 376
- browsers
 - appearance after restarting virtual directory in SQLXML 2.0, 602
 - troubleshooting in SQLXML 2.0, 609–610
- browsing scope, definition of, 19
- Bulk Load feature of SQLXML 2.0, resource for, 593
- C**
- *.cmd extension, explanation of, 303
- C (country) moniker, role in LDAP syntax, 421
- C# keywords related to OOP, list of, 510–511
- Cancel** Command class method, description of, 138
- CancelCurrentEdit CurrencyManager** class method, description of, 540
- CancelEdit DataRow** class property, description of, 246
- CanResolveEntity XmlReader** class property, 162
- Caption DataColumn** class property, description of, 250
- cascades, role in referential integrity, 44
- CaseSensitive** property
 - of **DataSet** class, 197
 - of **DataTable** class, 219
- catch** blocks
 - role in exception handling, 344, 348–349
 - role in filtering exceptions, 355
- ChangeDatabase** method
 - of **OleDbConnection** class, 81, 83
 - of **SqlConnection** class, 80
 - throws exception, 114
- Chaos** member of **IsolationLevel** Enum, description of, 102
- child tables
 - explanation of, 39
 - and referential integrity, 44–45
- ChildColumns DataRelation** class property, description of, 254
- ChildKeyConstraint DataRelation** class property, description of, 254
- ChildRelations DataTable** class property, description of, 219
- Children** property of **DirectoryEntry** class, description of, 417
- ChildTable DataRelation** class property, description of, 254
- Class View in Solution Explorer, switching to, 507
- classes, wrapping as components, 582
- clear **DataSet**, explanation of, 209
- clear **DataTable**, explanation of, 227
- Clear** method
 - of **DataSet** class, 199
 - of **DataTable** class, 221

- ClearErrors DataRow** class property, description of, 246
- client-side cursors, explanation of, 262
- clients, creating for UserMan example application, 582–583
- close connection, explanation of, 86
- Clone** method
 - of **DataSet** class, 199
 - of **DataTable** class, 221
 - using with **DataSet** objects, 210
 - using with **DataTable** objects, 228
- Close** method
 - of **DataReader** class, 152
 - of **DirectoryEntry** class, 419
 - of **OdbcConnection** class, 83
 - of **OleDbConnection** class, 81
 - of **SqlConnection** class, 80
 - of **XmlReader** class, 164
- Closed** member of **ConnectionState** Enum, description of, 89
- CLR (common language runtime)
 - handling exceptions with, 359
 - versus JVM, 9
 - .NET Framework adherence to, 6–8
- CLR tasks, built-in type of, 7
- CLS (Common Language Specification), .NET Framework adherence to, 4, 6
- CN (common name) moniker, role in LDAP syntax, 420–421
- code, running simple stored procedures from, 380–382, 401–405
- column and row change events, examining order of, 232–233
- column attributes for tables, displaying with Database Designer, 310
- column changes, handling in **DataTable** objects, 233–234
- Column grid pane item in Query Designer, 326
- ColumnChanged DataTable** class event, description of, 224
- ColumnChanging DataTable** class event, description of, 224
- ColumnMapping DataColumn** class property, description of, 250
- ColumnName DataColumn** class property, description of, 250
- columns
 - adding to tables with Table Designer, 315–316
 - checking for null values in, 149–150
 - choosing in Data Form Wizard, 532
 - in database table rows, definition of, 36
 - role in databases, 36
 - sorting with Query Designer, 324
- Columns DataTable** class property, description of, 219
- COM components, using from within .NET Framework, 276–278
- COM Interop
 - resource for, 56
 - using with ADO.NET disconnected layer, 276–278
- COM types, generating metadata for, 277–278
- Command class
 - methods, 129–137
 - properties, 138–139
- command files, adding for scripts, 303–304
- command mode, role in VS .NET IDE, 18–19
- command object parameters, adding to UserMan example application, 575–576
- Command objects
 - instantiating for UserMan database, 573–574
 - using in ADO.NET connected layer, 123
- command properties, setting in ADO.NET connected layer, 178–184
- Command Window in VS .NET IDE, modes for, 18–19
- CommandBuilder class, using in ADO.NET connection layer, 188–191
- commands, executing in ADO.NET connected layer, 139–142
- CommandText** property
 - of **OdbcCommand** class, 135
 - of **OleDbCommand** class, 132
 - of **SqlCommand** class, 130
- CommandTimeout** property
 - of **OdbcCommand** class, 135
 - of **OleDbCommand** class, 132
 - of **SqlCommand** class, 130
 - throws **ArgumentException** exception, 143
- CommandType** property
 - of **OdbcCommand** class, 135–136
 - of **OleDbCommand** class, 132–133
 - of **SqlCommand** class, 130
 - throws **ArgumentException** exception, 144
- Commit** method of Transaction class, description of, 109
- CommitChanges DirectoryEntry** class method, description of, 419

- Committed** member of **MessageQueueTransactionStatus** Enum, description of, 483
- common language runtime. *See* CLR
- Common Language Specification. *See* CLS
- common type system. *See* CTS
- compare connections, explanation of, 87–88
- comparisons, performing in ADO.NET connected layer, 87
- components, wrapping classes as, 582
- composite keys
 - in databases, definition of, 43
 - role in databases, 43
- Computer Management MMC versus Server Explorer, advisory when creating public message queues, 457
- Compute DataTable** class method, description of, 221
- concurrency violations
 - handling, 273–275
 - ignoring, 271–273
 - occurrence of, 269–270
- Connect As dialog box, displaying, 287
- Connect Timeout or Connection Timeout **ConnectionString** property value, details of, 68–69
- Connecting** member of **ConnectionState** Enum, description of, 89
- Connection class
 - events, 84–85
 - exceptions handling, 110–123
 - explanation of, 63
 - functionality of, 63–64
 - methods of, 80–84
 - properties of, 76–79
- Connection Lifetime **ConnectionString** property value, details of, 68–69
- Connection objects, comparing in ADO.NET connected layer, 87–88
- Connection OdbcCommand** class property, details of, 136
- Connection OleDbCommand** class property, details of, 133
- connection oriented programming, explanation of, 450
- connection pools
 - explanation of, 90–96
 - resetting connections in, 93
- Connection** property of Transaction class, description of, 108
- Connection Reset **ConnectionString** property value, details of, 70–71
- Connection Reset value, resetting connections in connection pools with, 93
- Connection SqlCommand** class property, details of, 130
- connection state, manipulating in ADO.NET connected layer, 88–96
- connection state messages, handling in ADO.NET connected layer, 97–98
- connectionless programming, explanation of, 450
- connections
 - choosing in Data Form Wizard, 530
 - closing in ADO.NET connection layer, 86
 - deleting with Server Explorer, 285
 - disposing in ADO.NET connected layer, 86–87
 - handling with Server Explorer, 282–285
 - opening in ADO.NET connected layer, 85–86
 - removing from pools, 92
 - setting with VS .NET IDE, 26
- ConnectionState**, comparing **State** to, 89
- ConnectionString** property
 - examining after setting, 76
 - example of white space in, 91
 - exception handling, 112–113
 - functionality of, 65
 - values for, 66–75
- ConnectionTimeout** property
 - exception handling, 113
 - of **OdbcConnection** class, 79
 - of **OleDbConnection** class, 78
 - of **SqlConnection** class, 77
- constraints, adding with Table Designer, 319–320
- Constraints DataTable** class property, description of, 219
- constructors, adding to CUser class for wrapped databases, 518–519
- Contains** public property and method of **BindingContext** class, description of, 538
- ContinueUpdateOnError** DataAdapter class property, description of, 172

- copy **DataSet**, explanation of, 209
 - copy **DataSet** structure, explanation of, 210
 - copy **DataTable**, explanation of, 227
 - copy **DataTable** rows, explanation of, 230–232
 - copy **DataTable** structure, explanation of, 227
 - Copy** method
 - of **DataSet** class, 200
 - of **DataTable** class, 222
 - CopyTo DataView** class method, 240
 - CopyTo DirectoryEntry** class method, description of, 419
 - Count DataView** class property, description of, 239
 - Count** property of **CurrencyManager** class, description of, 540
 - Create Command File dialog box, displaying, 303
 - Create Database dialog box, displaying, 292
 - Create Message Queue dialog box, displaying, 290
 - CreateCommand** method
 - of **OdbcConnection** class, 83
 - of **OleDbConnection** class, 81
 - of **SqlConnection** class, 80
 - CreateParameter** Command class method, description of, 138
 - Criteria grid pane column in Query Designer, 326
 - cross-language inheritance, role in .NET Framework, 8
 - Ctrl+Alt+J keyboard shortcut, opening Object Browser with, 19
 - Ctrl+Alt+K keyboard shortcut, accessing Task List window with, 27
 - Ctrl+Alt+S keyboard shortcut, accessing Server Explorer with, 281
 - Ctrl+S keyboard shortcut
 - saving changes to stored procedures with, 381, 383
 - saving database diagrams with, 310
 - saving SQL Editor scripts with, 336
 - saving triggers with, 408
 - saving views with, 399
 - Ctrl+Shift+C keyboard shortcut, switching to Class View in Solution Explorer with, 507
 - Ctrl+Shift+N keyboard shortcut, displaying New Project dialog box with, 296
 - CTS (common type system), .NET Framework adherence to, 4, 6
 - Culture, Processor, OS, manifest components of assemblies, description of, 10
 - CurrencyManager** class methods and properties, 540–541
 - CurrencyManager** object, role in Windows forms, 527, 538–544
 - Current Language **ConnectionString** property value, details of, 70–71
 - Current** property of **CurrencyManager** class, description of, 540
 - cursor locations, options for, 262–263
 - cursor types, 263–265
 - cursors, using in ADO.NET disconnected layer, 261–265
 - CUser class
 - adding command object parameters to, 575–576
 - adding constructors and destructors to, 518–519
 - adding error event class to, 520–521
 - adding events to, 519–524
 - creating, 513–524
 - declaring event and method raising event for, 522
 - declaring event delegate for, 522
 - displaying method receiving event in, 523
 - displaying private database constants and variables for, 567–568
 - filling data sets in, 576
 - hooking event receiver with event in, 523
 - hooking up public properties to data sets in, 576–578
 - instantiating and initializing data adapters in, 574–575
 - instantiating command objects in, 573–574
 - instantiating **DataSet** objects in, 574
 - locating code for, 579
 - making connection object shared in, 570
 - opening database connection for, 570–572
 - setting maximum column length variables for, 569–570
 - specifying parent classes in, 578
 - triggering invalid LoginName error event in, 523
- ## D
- DAP (Directory Access Protocol), introduction to, 413

- data
 - displaying in databases, 397
 - manipulating in views from code, 402–405
- data adapters, instantiating and initializing in UserMan example application, 574–575
- data binding, suspending and resuming, 543–544
- data-bound controls
 - choosing right data storage for, 549–551
 - communication with data sources, 528
 - controlling validation of edit operations in, 543–544
 - versus manual data hooking, 525–526
 - retrieving current row of data sources for, 542
 - retrieving and setting current position in data sources for, 542–543
 - retrieving number of rows in data sources for, 542
 - using Data Form Wizard with, 528–536
 - using with Web forms, 545–551
 - using with Windows forms, 527
- data-bound Web forms, creating and updating, 551–561
- data-bound Windows form controls, creating, 544
- data classes for .NET Data Providers, list of, 63
- data connections
 - choosing in Data Form Wizard, 530
 - closing in ADO.NET connection layer, 86
 - deleting with Server Explorer, 285
 - disposing in ADO.NET connected layer, 86–87
 - handling with Server Explorer, 282–285
 - opening in ADO.NET connected layer, 85–86
 - removing from pools, 92
 - setting with VS .NET IDE, 26
- data distribution, performance resources optimization of, 369
- Data Form Wizard
 - choosing data connections in, 530
 - choosing data display in, 533
 - choosing data sets in, 529
 - choosing tables and columns in, 532
 - choosing tables and views in, 531
 - creating data-binding forms with, 551–557
 - creating forms for data-bound controls with, 528–536
 - examining code created by, 536–537
 - examining frmUser.cs file in, 537
 - examining objUser object declaration in, 536–537
 - opening frmUser form with, 536
 - role of **Dispose** methods in, 536
 - sample form created by, 534
- data integrity
 - in databases, explanation of, 44
 - role in databases, 44
- Data Link Properties dialog box, displaying, 283–284
- data providers
 - in ADO.NET connected layer, 59–62
 - role in ADO and ADO.NET, 57
- data-related exceptions, handling, 357–359
- data-related namespaces, role in ADO.NET connected layer, 58–59. *See also* namespaces
- data sets
 - choosing in Data Form Wizard, 529
 - filling in UserMan example application, 576
- Data Source **ConnectionString** property value, details of, 66–67
- data source locking, role in ADO.NET disconnected layer, 265–276
- data source messages, handling in ADO.NET connection layer, 97–98
- data sources
 - binding ASP.NET server controls to, 545–548
 - binding Windows form controls to, 537–538
 - locking data at, 266–268
 - retrieving and setting current position for data-bound controls, 542–543
 - retrieving number of rows for data-bound controls, 541–542
 - updating from **DataSet** objects, 618–619
 - updating with DataAdapter classes, 206–209
 - updating with **SqlXmlCommand** class, 631–635
- data storage, choosing for data-bound controls, 549–551
- data wrappers
 - definition of, 505

- reasons for use of, 505
- DataAdapter classes
 - explanation of, 170–171
 - events, 174–175
 - methods of, 173–174
 - populating **DataSet** objects with, 203–206
 - preparing for **CommandBuilder** class, 189–191
 - properties of, 172
 - role in ADO.NET, 59, 170–172
 - setting command properties, 178–184
 - updating data sources with, 206–209
- DataAdapter object, instantiating, explanation of 176–178
- Database **ConnectionString** property
 - value, details of, 66–67
- DATABASE **ConnectionString** property
 - value, details of, 66–67
- Database Designer
 - adding relationships with, 311
 - adding tables with, 309
 - creating tables with, 309–313
 - deleting and removing tables with, 309
 - deleting relationships with, 311
 - designing databases with, 306–313
 - displaying column attributes with, 310
 - editing database properties with, 311
- database diagrams
 - changing table views for, 313
 - changing viewable area of, 312
 - creating with Database Designer, 306–313
 - moving view ports in, 312–313
 - overview of, 311–312
 - saving, 314
 - showing relationship names in, 312
 - sizing tables automatically in, 313
 - zooming in, 311–312
- database objects
 - adding to database projects, 300–304
 - creating with Server Explorer, 285
 - identifying, 37
- database projects
 - adding database objects to, 300–304
 - creating, 296–298
 - creating folders for, 298–299
 - definition of, 296
 - deleting folders from, 299
- database properties, editing with Database Designer, 311
- Database property
 - of **OdbcConnection** class, 79
 - of **OleDbConnection** class, 78
 - of **SqlConnection** class, 77
 - throws exceptions, 114
- Database Query database object template, description of, 301
- database tables versus message queues, 452
- databases. *See also* SQL Server databases
 - benefits of, 32–33
 - columns in, 36
 - composite keys in, 43
 - data integrity in, 44
 - declarative referential integrity (DRI) in, 45
 - definition of, 31–32
 - designing with Database Designer, 306–313
 - designing with Visio for Enterprise Architect, 305
 - determining objects for, 37–38
 - displaying data in, 397
 - domain integrity in, 45
 - entity integrity in, 44
 - fields in, 36
 - foreign keys in, 42–43
 - hierarchical, description of, 33–34
 - indexes in, 43
 - keys in, 42
 - lookup keys in, 42
 - normalization in, 46
 - null values in, 36
 - primary keys in, 42–43
 - procedural referential integrity (PRI) in, 45
 - reasons for use of, 32–33
 - records in, 36
 - referential integrity in, 44–45
 - rows in, 36
 - tables in, 36
 - wrapping, 513–524
- DataBind** method, role in binding ASP.NET server controls to data sources, 545, 547–548
- DataBindings dialog box, displaying, 560
- DataColumn** class
 - properties of, 250–252
 - using in ADO.NET disconnected layer, 249–252
- DataColumn** objects, declaring and instantiating, 252
- DataGrid Properties dialog box, displaying, 553
- DataGridCancelCommand** event procedure, displaying, 554–555
- DataGridEditCommand** event procedure, displaying, 553

- DataGridUpdateCommand** event procedure, displaying, 555–556
- DataReader** classes
 - closing, 149
 - exception handling, 160
 - handling multiple results with, 150
 - instantiating, 176–178
 - methods of, 151–160
 - properties of, 151
 - reading rows in, 148
 - usage of, 160–161
 - use of server-side cursors by, 263
 - using with ADO.NET connected layer, 146
- DataReader** objects, declaring and instantiating, 147–148
- DataRelation** class
 - properties of, 254
 - using in ADO.NET disconnected layer, 252–261
- DataRelation** objects
 - declaring and instantiating, 254–256
 - exceptions, 253
- DataRow** and **DataColumn** classes, functionality of, 232–233
- DataRow** class, using in ADO.NET disconnected layer, 245–249
- DataRow** objects
 - building, 249
 - declaring and instantiating, 249
- DataSet** class
 - events of, 202–203
 - methods of, 199–201
 - properties of, 197–199
 - using with ADO.NET disconnected layer, 193–194
- DataSet Designer**, creating typed data sets with, 338–340
- DataSet** objects
 - accepting or rejecting changes to data in, 214–218
 - building UserMan database schema as, 256–261
 - clearing data from, 209
 - copying data and structures of, 209–210
 - detecting and handling changes to data in, 212–214
 - instantiating, 203
 - instantiating in CUser class, 574
 - merging data between, 210–212
 - populating with DataAdapter class, 203–206
 - populating with **SqlXmlAdapter** class, 614–616
 - updating data sources from, 618–619
 - using **AcceptChanges** method with, 215
 - using **RejectChanges** method with, 216–218
 - using with data-bound controls, 550
- DataSet** property
 - of **DataRelation** class, 254
 - of **DataTable** class, 219
- DataSetName DataSet** class property, description of, 197
- DataSource** property
 - of **OdbcConnection** class, 79
 - of **OleDbConnection** class, 78
 - of **SqlConnection** class, 77
- DataTable** class
 - events of, 223–224
 - methods of, 220–223
 - properties of, 218–220
 - using with **DataSet** class, 194, 218–223
- DataTable** objects
 - building, 225–227
 - clearing data from, 227
 - copying, 227–229
 - copying rows in, 230–232
 - declaring and instantiating, 224–225
 - examining order of column and row change events in, 232–233
 - handling column changes in, 233–234
 - handling row changes in, 235–236
 - handling row deletions in, 236–238
 - merging **DataSet** objects with, 212
 - populating, 227
 - searching and retrieving filtered data views from, 229–230
 - using with **CommandBuilder** class, 188–189
- DataType DataColumn** class property, description of, 250
- DataView** class
 - events of, 241
 - methods of, 240
 - properties of, 239
 - using in ADO.NET disconnected layer, 238–245
- DataView** objects
 - declaring and instantiating, 242
 - searching, 243
 - sorting, 244–245
- DataViewManager DataView** class
 - property, description of, 239
- DBCC** (Database Consistency Checker), functionality of, 371–372
- DBMS** (database management system), definition of, 35

- dbo (database owner), role in creating SQL Server stored procedures, 376
 - DBObject virtual name, querying objects with, 605–606
 - DBQ **ConnectionString** property value, details of, 66–67
 - DC (domain component) moniker, role in LDAP syntax, 421
 - dead-letter message queues
 - explanation of, 485–486
 - rejecting messages in, 492
 - debug assertions, using, 363–364
 - Debug** class
 - advisory about, 364
 - methods and properties of, 366
 - using, 359–366
 - using error messages with, 364–366
 - debugger in VS .NET IDE, explanation of, 20–21
 - debugging, enabling and disabling, 360–362
 - DefaultValue DataColumn** class property, description of, 250
 - DefaultView DataTable** class property, description of, 219
 - DefaultViewManager DataSet** class property, description of, 197
 - Delete DataRow** class property, description of, 246
 - Delete DataView** class method, 240
 - DELETE queries, using Query Designer for, 331–332
 - DeleteCommand** DataAdapter class property, description of, 172
 - DeleteTree DirectoryEntry** class method, description of, 419
 - denormalization, explanation of, 51
 - deployment tools, upgrades in VS .NET IDE, 23
 - Depth** property of DataReader class, description of, 151
 - Depth XmlReader** class property, 162
 - DesignTimeVisible** property
 - of **OdbcCommand** class, 136
 - of **OleDbCommand** class, 133
 - of **SqlCommand** class, 130
 - destructors, adding to CUser class for wrapped databases, 518–519
 - deterministic finalization versus automatic garbage collection, 8
 - diagram pane of Query Designer, functionality of, 322–325
 - DiffGrams
 - advisory about using with SQLXML 2.0 Managed classes, 619
 - generating and executing, 633–635
 - direct URL queries in SQLXML 2.0, advisory about, 606
 - DirectoryEntries** class in **System.DirectoryServices** namespace, description of, 416
 - DirectoryEntry** class
 - description of, 416
 - methods of, 419–420
 - studying in Active Directory, 417–422
 - in **System.DirectoryServices** namespace, 415
 - DirectorySearcher** class
 - description of, 416
 - in **System.DirectoryServices** namespace, 415
 - using with objects in Active Directory, 422–423
 - DisplayExpression DataTable** class property, description of, 219
 - Dispose** methods, role in Data Form Wizard code, 536
 - dispose of connection, explanation of, 86–87
 - distributed applications versus stand-alone applications, 368
 - DLL (dynamic link library) classes versus assemblies, 10
 - DN (distinguished name), role in Active Directory, 420
 - domain integrity in databases, explanation of, 45
 - DRI (declarative referential integrity) in databases, explanation of, 45
 - DRIVER ConnectionString** property value, details of, 66–67
 - driver messages, handling in ADO.NET connection layer, 97–98
 - Driver OdbcConnection** class property, description of, 79
 - drivers. *See* OLE DB drivers
 - DSN **ConnectionString** property value, details of, 68–69
 - dynamic cursors, explanation of, 265
 - Dynamic Help, role in VS .NET IDE, 21–22
- ## E
- edit mode, generating data-bound Web forms in, 555
 - edit operations, controlling validation for data-bound controls, 543–544
 - ELEMENTS SELECT . . . FOR XML statement keyword, description of, 588

- encapsulation, role in OOP, 510
 - encrypted messages, sending and receiving, 498–499
 - encryption
 - securing message queuing with, 495–499
 - using views for, 398
 - EndCurrentEdit CurrencyManager** class method, description of, 540
 - EndEdit DataRow** class property, description of, 246
 - EndLoadData DataTable** class method, description of, 222
 - EnforceConstraints DataSet** class property, description of, 197
 - Enlist **ConnectionString** property value, details of, 70–71
 - entity integrity
 - in databases, explanation of, 44
 - role in databases, 44
 - enums, tip when passing arguments to methods and properties, 522
 - EOF XmlReader** class property, 162
 - equal to (=) column comparison symbol, role in Query Designer join types, 323
 - Equals** method, using in ADO.NET connected layer, 87–88
 - Equals** method
 - of **OdbcConnection** class, 83
 - of **OleDbConnection** class, 82
 - of **SqlConnection** class, 80
 - ER/Studio, Web site for, 37
 - ErrorCode** property of **OleDbException** class, details of, 117
 - Errors** property
 - of **OdbcException** class, 121
 - of **OleDbException** class, 117
 - ERwin, Web site for, 37
 - events, adding to CUser class for wrapped databases, 519–524
 - Exception** class
 - examining, 346–348
 - methods of, 348
 - exception handlers
 - handling exceptions in, 348–354
 - using two or more in single procedures, 345–346
 - exception handling
 - enabling, 345
 - functionality of, 343–344
 - role of **catch** blocks in, 344
 - role of **finally** blocks in, 344–345
 - role of **try** statement in, 344
 - types of handlers in, 345
 - Exception** type, 350
 - exceptions
 - CLR handling of, 359
 - creating, 356
 - filtering, 354–356
 - handling data-related type of, 357–359
 - throwing, 356–357
 - types of, 350–351
 - understanding generation of, 111
 - Exchange Server 2000
 - accessing, 436–447
 - accessing as linked server from SQL Server, 442–447
 - retrieving contacts for UserMan user from, 438–439
 - reviewing content of sample folder with, 440–441
 - ExecuteNonQuery** method
 - description of, 138
 - usage of, 140–141, 631–632
 - ExecuteReader** method
 - description of, 138
 - usage of, 140–141
 - ExecuteScalar** method
 - description of, 138
 - usage of, 140, 142
 - ExecuteXmlReader** method
 - description of, 139
 - usage of, 140, 142
 - Executing** member of **ConnectionState** Enum, description of, 89
 - Exists DirectoryEntry** class method, description of, 419
 - ExOLEDB (Microsoft OLE DB Exchange Server Provider), using, 438–439
 - EXPLICIT SELECT . . . FOR XML statement keyword, description of, 588
 - Expression DataColumn** class property, description of, 251
 - ExtendedProperties** property
 - of **DataColumn** class, 251
 - of **DataRelation** class, 254
 - of **DataSet** class, 197
 - of **DataTable** class, 219
 - external transactions, role in message queues, 480
- ## F
- Fail** method, role in using error messages with Debug class, 365
 - FAQs (Frequently Asked Questions), about databases, list of, 35–51
 - fault handlers, role in exception handling, 345

- Fetching** member of **ConnectionState** Enum, description of, 89
 - FieldCount** property of **DataReader** class, description of, 151
 - fields, role in databases, 36
 - FIFO (First In, First Out) principle versus message queues, 453
 - FIL **ConnectionString** property value, details of, 70–71
 - file-based XML templates, executing from code with SQLXML 2.0, 617
 - File Name **ConnectionString** property value, details of, 70–71
 - File Table manifest component of assemblies, description of, 10
 - Fill** method
 - of **DataAdapter** class, 173
 - populating **DataSet** objects with, 203–206
 - fill operation errors, handling in ADO.NET connection layer, 184–185
 - FillError** **DataAdapter** class event, description of, 175
 - FillSchema** **DataAdapter** class method, description of, 173
 - filter handlers, role in exception handling, 345
 - finally** blocks, role in exception handling, 344–345
 - finally** handlers, role in exception handling, 345
 - Find DataView** class method, 240
 - FindRows DataView** class method, 240
 - FirstName** column in **tblUser** column for **CUser** class, details of, 514
 - folders
 - creating for database projects, 298–299
 - deleting for database projects, 299
 - foreign keys
 - in databases, definition of, 42
 - role in databases, 42–43
 - ForeignKeyConstraint** objects, advisory when using **DataSet** objects, 215
 - format names, binding to existing message queues with, 460–461
 - formatters, using with message queues, 464–467
 - forms, creating for data binding, 551–557
 - forms for data-bound controls, creating with Data Form Wizard, 528–536
 - forward-only cursors, explanation of, 263–264
 - friendly names, binding to existing message queues with, 459–460
 - frmUser** form, opening with Data Form Wizard, 536
 - frmUser.cs** file, examining in Data Form Wizard, 537
- ## G
- garbage collection, role in .NET Framework, 7–8
 - Generate Dataset dialog box, displaying, 559
 - Get*** methods of **DataReader** class, 152–159
 - GetAttribute XmlReader** class method, 164
 - GetBaseException** method
 - of **OdbcException** class, 123
 - of **OleDbException** class, 119
 - GetBoolean** method of **DataReader** class, 152
 - GetByte** method of **DataReader** class, 152
 - GetBytes** method of **DataReader** class, 152
 - GetChanges** method
 - of **DataSet** class, 200
 - of **DataTable** class, 222
 - GetChar** method of **DataReader** class, 153
 - GetChars** method of **DataReader** class, 153
 - GetChildRows DataRow** class property, description of, 247
 - GetColumnError DataRow** class property, description of, 247
 - GetColumnsInError DataRow** class property, description of, 247
 - GetDataTypeName** method of **DataReader** class, 153
 - GetDateTime** method of **DataReader** class, 153
 - GetDecimal** method of **DataReader** class, 153
 - GetDouble** method of **DataReader** class, 154
 - GetEnumerator DataView** class method, 240
 - GetErrors DataTable** class method, description of, 222
 - GetFieldType** method of **DataReader** class, 154
 - GetFillParameters** **DataAdapter** class method, description of, 174

- GetFloat** method of **DataReader** class, 154
 - GetGuid** method of **DataReader** class, 154
 - GetHashCode** method
 - of **OdbcConnection** class, 83
 - of **SqlConnection** class, 80
 - GetInt16** method of **DataReader** class, 154
 - GetInt32** method of **DataReader** class, 154
 - GetInt64** method of **DataReader** class, 155
 - GetName** method of **DataReader** class, 155
 - GetOleDbSchemaTable**
 - OleDbConnection** class
 - method, details of, 82
 - GetOrdinal** method of **DataReader** class, 155
 - GetParentRow DataRow** class property, description of, 247
 - GetParentRows DataRow** class property, description of, 248
 - GetSchemaTable** method of **DataReader** class, 155
 - GetSqlBinary** method of **DataReader** class, 155
 - GetSqlBoolean** method of **DataReader** class, 156
 - GetSqlByte** method of **DataReader** class, 156
 - GetSqlDateTime** method of **DataReader** class, 156
 - GetSqlDecimal** method of **DataReader** class, 156
 - GetSqlDouble** method of **DataReader** class, 156
 - GetSqlGuid** method of **DataReader** class, 156
 - GetSqlInt16** method of **DataReader** class, 157
 - GetSqlInt32** method of **DataReader** class, 157
 - GetSqlInt64** method of **DataReader** class, 157
 - GetSqlMoney** method of **DataReader** class, 157
 - GetSqlSingle** method of **DataReader** class, 157
 - GetSqlString** method of **DataReader** class, 157
 - GetSqlValue** method of **DataReader** class, 158
 - GetSqlValue** method of **DataReader** class, 158
 - GetSqlValues** method of **DataReader** class, 158
 - GetString** method of **DataReader** class, 158
 - GetTimeSpan** method of **DataReader** class, 158
 - GetType** method
 - of **OdbcConnection** class, 83
 - of **OleDbConnection** class, 82
 - of **SqlConnection** class, 80
 - GetValue** method of **DataReader** class, 158
 - GetValues** method of **DataReader** class, 159
 - GetXml DataSet** class method, description of, 200
 - GetXmlSchema DataSet** class method, description of, 200
 - greater than (>) column comparison symbol, role in Query Designer join types, 323
 - greater than or equal to (>=) column comparison symbol, role in Query Designer join types, 323
 - grid pane of Query Designer, functionality of, 325–327
 - Group By grid pane column in Query Designer, 326
 - guarded blocks of code, using exception handlers with, 345
 - Guid** property of **DirectoryEntry** class, description of, 417
- ## H
- Has . . . a inheritance, explanation of, 507
 - HasAttributes XmlReader** class property, 162
 - HasChanges DataSet** class method, description of, 200
 - HasErrors** property
 - of **DataRow** class, 245
 - of **DataSet** class, 198
 - of **DataTable** class, 219
 - HasValue XmlReader** class property, 162
 - HasVersion DataRow** class property, description of, 248
 - Help system, integration in VS .NET IDE, 21–22
 - HelpLink** property
 - of **Exception** class, 347
 - of **OdbcException** class, 121
 - of **OleDbException** class, 117
 - hierarchical databases, description of, 33–34

- hierarchical versus relational databases, 33–35
- HResult** property
 - of **OdbcException** class, 121
 - of **OleDbException** class, 118
- I**
- ICT Database Designer Tool, Web site
 - for, 37
- Id column in tblUser column for CUser class, details of, 514–515
- Identity manifest component of assemblies, description of, 10
- IDEs (Integrated Development Environments). *See also* VS .NET IDE
 - running scripts in, 302
 - running simple stored procedures from, 379–380
 - running stored procedures with arguments from, 384
 - running views from, 400–401
- IDL (Interface Definition Language) files,
 - role in language compilation, 8
- IDs, retrieving for message queues, 458–459, 468–469
- immediate mode, role in VS .NET IDE, 18–19
- implementation inheritance, role in OOP, 509–510
- ImportRow** method, using with **DataTable** objects, 232
- ImportRow DataTable** class method,
 - description of, 222
- IMS (Information Management System),
 - Web site for, 34
- indexes
 - adding with Table Designer, 317–319
 - in databases, definition of, 43
 - role in databases, 43
- IndexOutOfRangeException** exception, 350, 352, 355
- InferXmlSchema DataSet** class method,
 - description of, 200
- InfoMessage** Connection class event,
 - details of, 85
- information hiding, role in OOP, 510
- inheritance, role in OOP, 507–510
- Initial Catalog **ConnectionString** property value, details of, 70–71
- Initial File Name **ConnectionString** property value, details of, 70–71
- Initialized** member of **MessageQueueTransactionStatus** Enum, description of, 483
- InnerException** property
 - of **Exception** class, 347
 - of **OdbcException** class, 122
 - of **OleDbException** class, 118
- INSERT queries, using Query Designer
 - for, 333
- INSERT triggers, functionality of, 406
- InsertCommand** DataAdapter class
 - property, description of, 172
- integrated debugger in VS .NET IDE,
 - explanation of, 20–21
- Integrated Security **ConnectionString** property value, details of, 70–71
- interface inheritance, role in OOP, 507–509
- interface** keyword in C#, relationship to OOP, 511
- interface modes for VS .NET IDE, explanation of, 16
- interfaces, creating and implementing with OOP, 508–509
- internal transactions, role in message queues, 480–481
- Interop Assemblies, location of, 278
- InvalidCastException** exception
 - handling for **DataReader** class, 160
 - thrown by **XmlReader** methods, 164
- InvalidConstraintException** exception
 - thrown by **DataRelation** class, 253
- InvalidOperationException** exception, 351–352
 - handling for **DataReader** class, 160
 - handling for **XmlReader** class, 170
 - throwing, 111–112, 115, 144–145
- Invoke DirectoryEntry** class method,
 - description of, 419
- Is a . . . inheritance, explanation of, 507, 509
- ISAPI extension
 - configuring for SQLXML 2.0, 593–598
 - testing for SQLXML 2.0, 599
- IsClosed** property of **DataReader** class,
 - description of, 151
- IsDBNull** method of **DataReader** class, 159
- IsDefault XmlReader** class property, 162
- IsEmptyElement XmlReader** class
 - property, 162
- IsName XmlReader** class method, 164
- IsNameToken XmlReader** class method, 164
- IsNull DataRow** class property,
 - description of, 248

isolation levels, determining for running transactions, 107–108

IsolationLevel Enum, members of, 102

IsolationLevel property of Transaction class, description of, 108

IsReadOnly public property and method of **BindingContext** class, description of, 538

IsStartElement XmlReader class method, 164

ItemArray DataRow class property, description of, 245

J

JIT (Just in Time) compiler, functionality of, 9

join dependence, role in 5NF, 51

join types, changing with diagram pane of Query Designer, 323–324

journal storage

controlling size of, 489–490

retrieving messages from, 489

using with message queues, 486–490

JScript, implications of CLR for, 6–7

JVM (Java Virtual Machine)

versus CLR, 9

K

keys

adding with Table Designer, 317–319

in databases, definition of, 42

role in databases, 42

keyset cursors, explanation of, 265

L

labels, binding to existing message queues with, 461

LastName column in tblUser column for CUser class, details of, 514

LDAP display names for users, Web site for, 425

LDAP (Lightweight Directory Access Protocol), introduction to, 413–414

LDAP query filters, Web sites for, 425

LDAP syntax, examining, 420–421

less than (<) column comparison symbol, role in Query Designer join types, 323

less than or equal to (<=) column comparison symbol, role in Query Designer join types, 323

line numbers, using in VS .NET IDE text editors, 24

link tables, using with many-to-many relationships, 41

linked servers

accessing Microsoft Exchange Server as, 442–447

creating, 443–444

creating views on, 447

dropping, 446

setting up, 442

List property of **CurrencyManager** class, description of, 540

LoadDataRow DataTable class method, description of, 222

local processing resources, performance resources optimization of, 369

Locale property

of **DataSet** class, 198

of **DataTable** class, 219

LocalName XmlReader class property, 162

locking types, explanations of, 265–276

LoginName column in tblUser column for CUser class, details of, 514,

518, 523

lookup keys, role in databases, 42

LookupNamespace XmlReader class method, 164

M

machine names, creating message queues with, 454–455

Macros IDE, accessing, 22–23

mail messages, sending with SMTP, 437

mail, retrieving from Active Directory with OLE DB .NET data providers, 432

Make Table queries, using Query Designer for, 332

Managed Classes in SQLXML 2.0, introduction to, 611

managed code

definition of, 6

role in .NET Framework, 7

manifest components of assemblies, list of, 10

manipulation of databases, understanding, 33

manual data hooking versus data-bound controls, 525–526

manual transactions

aborting, 106

ADO.NET connection layer support for, 99

committing, 107

summary of, 109–110

many forms, 506–507

- many-to-many relationships, explanation of, 41
- mapping schema for XPath queries, displaying, 629–630
- Max Pool Size **ConnectionString** property value, details of, 70–71
- Max Pool Size value names, role in pooling connections for **SqlConnection**, 92
- MaxLength DataColumn** class property, description of, 251
- MDAC (Microsoft Data Access Components)
 - 2.6, required on your system, 61
 - 2.7, distributed with VS .NET, 61
- MDI (multiple document interface) mode, explanation of, 16
- merge **DataSet**, explanation of, 210
- Merge DataSet** class method, description of, 200
- merge failures, handling with **DataSet** class, 202–203
- MergeFailed DataSet** class event, 200
- Message Exception** class property, description of, 347
- message formatters, setting up for message queues, 464–467
- message journaling, explanation of, 488–489
- Message** property
 - of **OdbcError** class, 121
 - of **OdbcException** class, 122
 - of **OleDbError** class, 117
 - of **OleDbException** class, 118
- message queues
 - aborting transactions for, 483
 - assigning labels to, 458
 - binding to existing types of, 459–461
 - binding with labels, 461
 - clearing messages from, 472–473
 - committing transactions for, 482–483
 - controlling storage size of, 479–480
 - creating programmatically, 454–456
 - creating with Server Explorer, 289–290
 - deleting with Server Explorer, 290
 - displaying or changing properties in, 456–459
 - enabling authentication on, 491
 - enabling journaling on, 487
 - ending transactions for, 482–483
 - locating, 475–477
 - making transactional, 480–485
 - peeking at messages in, 467
 - picking specific messages from, 468–469
 - prioritizing messages in, 473–475
 - rejecting nonencrypted messages in, 496–497
 - removing, 477–479
 - removing messages after specified time elapse, 493
 - retrieving all messages from, 470
 - retrieving IDs of, 458–459
 - retrieving messages from, 463–467
 - sending and retrieving messages from, 466–467
 - sending messages to, 461–463
 - setting up message formatters for, 464–467
 - starting transactions for, 481
 - usage of, 449, 451–453
 - using journal storage with, 486–490
 - using system-generated type of, 485–490
 - using system journals with, 486–487
 - using transactions with, 484–485
- message queuing
 - using access control with, 499–502
 - using auditing with, 502
 - using authentication with, 490–495
 - using encryption with, 495–499
- MessageQueue** class, examining, 450
- MessageQueueTransaction** class, using, 483–485
- messages
 - deleting messages from, 290
 - enabling journaling on, 487–488
 - rejecting nonauthenticated type of, 491
 - removing from message queues, 472–473
 - removing from message queues after specified time elapse, 493
 - retrieving from journal storage, 489
 - sending and retrieving asynchronously, 470–472
 - sending and retrieving from message queues, 466–467
- metadata, generating for COM types, 277–278
- Microsoft Access, advisory about, 367–368
- Microsoft Data Access Components. *See* MDAC.
- Microsoft Exchange Server
 - accessing, 436–447
 - accessing as linked server from SQL Server, 442–447
 - retrieving contacts for UserMan user from, 438–439
 - reviewing content of sample folder with, 440–441

- Microsoft OLE DB Exchange Server Provider (ExOLEDB), using, 438–439
 - Microsoft OLE DB Provider for Internet Publishing (MSDAIPP), using, 439–441
 - Microsoft Visual Studio .NET documentation, role in VS .NET IDE Help system, 22
 - Microsoft.Data.Odbc** namespace, description of, 58
 - Microsoft.Jet.OLEDB.4.0 provider, description of, 61
 - Min Pool Size **ConnectionString** property value, details of, 72–73
 - Min Pool Size value names, role in pooling connections for **SqlConnection**, 92
 - MinimumCapacity DataTable** class property, description of, 220
 - MissingMappingAction** DataAdapter class property, description of, 172
 - MissingSchemaAction** DataAdapter class property, description of, 172
 - monikers, role in LDAP syntax, 420–421
 - Move* XmlReader** class methods, 165–166
 - MoveTo DirectoryEntry** class method, description of, 419
 - MoveToAttribute XmlReader** class method, 165
 - MoveToContent XmlReader** class method, 165
 - MoveToElement XmlReader** class method, 165
 - MoveToFirstAttribute XmlReader** class method, 165
 - MoveToNextAttribute XmlReader** class method, 166
 - MSDAIPP (Microsoft OLE DB Provider for Internet Publishing), using, 439–441
 - MSDAORA provider, description of, 61
 - MSIL (Microsoft intermediate language), compilation of code to, 8
 - MSMQ (Microsoft Message Queue Server), explanation of, 449–450
 - multivalued dependencies, role in 4NF, 49
 - MySQL, advisory about, 368
- ## N
- n-tier applications, relationship to ADO.NET, 56
 - Name** property
 - of **DirectoryEntry** class, 418
 - of **SqlParameter** class, 612
 - of **XmlReader** class, 162
 - named parameters, executing SQL queries with, 623–625
 - named SQL transactions, beginning, 103
 - named transactions, tip about, 107
 - namespace pollution, definition of, 12
 - Namespace** property
 - of **DataColumn** class, 251
 - of **DataSet** class, 198
 - of **DataTable** class, 220
 - namespaces. *See also* data-related namespaces
 - determining for classes, 123
 - functionality of, 11–13
 - public status of, 13
 - NamespaceURI XmlReader** class property, 163
 - NameTable XmlReader** class property, 163
 - NativeError** property
 - of **OdbcError** class, 121
 - of **OleDbError** class, 117
 - NativeGuid** property of
 - DirectoryEntry** class, description of, 418
 - NativeObject** property of
 - DirectoryEntry** class, description of, 418
 - Nested DataRelation** class property, description of, 254
 - nested OleDb transactions, beginning, 105–106
 - nesting, definition of, 397
 - .NET Data Providers
 - Connection class in, 63–64
 - explanation of, 62–63
 - .NET-enabled applications, connecting to, 64
 - .NET Framework
 - components of, 4–9
 - cross-language inheritance in, 8–9
 - data-related namespaces in, 58
 - description of, 3–4
 - garbage collection in, 7–8
 - managed code in, 7
 - using COM components from within, 276–278

- .NET Framework class library, contents of, 13–14
- Net or Network Library
 - ConnectionString** property value, details of, 72–73
- Network Address **ConnectionString** property value, details of, 72–73
- network directory services, explanation of, 413
- network resources, performance
 - resources optimization of, 369
- New Project dialog box, displaying, 296
- New Value grid pane column in Query Designer, 326
- NewRow DataTable** class method, description of, 222
- NextResult** method of **DataReader** class, 159
- nodes, separating in Active Directory hierarchy, 420
- NodeType XmlReader** class property, 163
- non-row returning SQL statements, executing, 632
- non-row-returning string templates, executing, 632–633
- non-row-returning templates, executing, 633
- normal forms, explanation of, 46–51
- normalization in databases, explanation of, 46
- not equal to (<>) column comparison symbol, role in Query Designer join types, 323
- NotSupportedException** exception, handling for **DataReader**, 160
- null values
 - advisory when using
 - CommandBuilder** class, 190
 - checking in columns, 149–150
 - in databases, definition of, 36
 - role in databases, 36
- NullReferenceException** exception, explanation of, 350, 352
- O**
- O (organization) moniker, role in LDAP syntax, 421
- Object Browser, role in VS .NET IDE, 19–20
- object property values, manipulating with Active Directory, 426–429
- objects
 - binding to in Active Directory, 421–422
 - checking structural integrity with
 - DBCC, 371–372
 - determining for databases, 37
 - querying with **DObject** virtual name, 605–606
 - searching in Active Directory, 422–425
- objectSid retrieving from Active Directory with OLE DB .NET data providers, 432
- objUser object declaration, examining in Data Form Wizard, 536–537
- ODBC connection pooling, disabling, 94–96
- ODBC connection strings, setting up, 61–62
- ODBC object pooling, clearing, 96
- ODBC (Open Data Base Connectivity), Web site for, 96
- ODBC (Open Data Base Connectivity) data sources, accessing in ADO, 59
- ODBC standard escape sequences, Web sites for, 137
- OdbcCommand** class
 - instantiating, 126–127
 - properties of, 79, 135–137
 - usage of, 124
- OdbcConnection** class,
 - BeginTransaction** method of, 101
- OdbcConnection** data type, pooling connections of, 94–96
- OdbcConnection** exceptions, handling, 119–123
- OdbcConnection** managed connection, explanation of, 63–64
- OdbcDataAdapter** class, setting command properties in, 182–184
- OdbcDataReader** class, use of, 147
- OdbcError** class, examining, 119–120
- OdbcException** class, examining, 121–123
- ODBC .NET data provider, downloading, 58
- OdbcTransaction** classes, nesting transactions with, 106
- OLE DB connection pooling, disabling, 93
- OLE DB drivers
 - in ADO.NET connected layer, 59
 - specifying when connecting, 61–62
- OLE DB .NET data provider, accessing Active Directory with, 429–436

- OLE DB (Object Linking and Embedding Data Base) specification, explanation of, 59
 - OLE DB object pooling, clearing, 93–94
 - OLE DB providers
 - compatibility with OLE DB.NET providers, 61
 - selecting for data sources with Server Explorer, 283
 - specifying for Active Directory connections, 430–431
 - specifying when connecting, 61–62
 - OLE DB.NET providers compatible with OLE DB providers, list of, 61
 - OleDbCommand** class
 - properties of, 132–134
 - usage of, 124
 - OleDbCommand** object, instantiating, 124–126
 - OleDbConnection** class
 - methods of, 81–84
 - properties of, 78
 - OleDbConnection** data type, pooling connections of, 93–94
 - OleDbConnection** exceptions, throwing, 115–119
 - OleDbConnection** managed connection, explanation of, 63–64
 - OleDbDataAdapter** class, setting command properties in, 181–182
 - OleDbDataReader** class, use of, 147
 - OleDbError** class, examining, 116–117
 - OleDbException** class
 - examining, 117–119
 - methods of, 119
 - traversing, 116
 - OleDbTransaction** class, nesting transactions with, 105–106
 - one-to-many relationships, explanation of, 40
 - one-to-one relationships, explanation of, 39–40
 - OOP (Object Oriented Programming)
 - C# keywords related to, 510–511
 - overriding properties in derived classes with, 512–513
 - overview of, 506–513
 - resources for, 506
 - role of encapsulation in, 510
 - role of implementation inheritance in, 509–510
 - role of inheritance in, 507–510
 - role of polymorphism in, 506–507
 - open connection, explanation of, 85–86
 - Open** member of **ConnectionState** Enum, description of, 89
 - Open** method
 - of **OdbcConnection** class, 83
 - of **OleDbConnection** class, 82
 - of **SqlConnection** class, 81
 - throwing exceptions, 115
 - optimistic locking, explanation of, 265, 268–276
 - optimization issues, explanation of, 368–373
 - Option setting for message queue encryption, explanation of, 498
 - Options dialog box, accessing, 24–25
 - Options dialog box in SQL Editor, displaying, 334
 - Or . . . grid pane column in Query Designer, 326
 - Oracle, advisory about, 368
 - Oracle Database Project, downloading, 392
 - Oracle stored functions, running, 391–392
 - Oracle stored procedures
 - retrieving result set from, 396
 - running, 392–397
 - order systems, typical objects for, 38
 - Ordinal DataColumn** class property, description of, 251
 - organization of databases, understanding, 32
 - orphaned child tables, preventing with referential integrity, 45
 - OU (organizational unit) moniker, role in LDAP syntax, 420–421
 - Output grid pane column in Query Designer, 326
 - override** keyword in C#, relationship to OOP, 511
 - ownership chain, definition of, 391
- ## P
- Packet Size **ConnectionString** property value, details of, 72–73
 - PacketSize SqlConnection** class property, details of, 77
 - Page_Load** event procedure, displaying, 556–557
 - Parameters** property
 - of **OdbcCommand** class, 136
 - of **OleDbCommand** class, 133
 - of **SqlCommand** class, 130
 - parent classes, specifying in CUser class, 578
 - Parent** property of **DirectoryEntry** class, description of, 418
 - parent tables, explanation of, 39

- ParentColumns DataRelation** class
 - property, description of, 254
- ParentKeyConstraint DataRelation**
 - class property, description of, 254
- ParentRelations DataTable** class property, description of, 220
- ParentTable DataRelation** class property, description of, 254
- Password column in **tblUser** column for **CUser** class, details of, 514
- Password or Pwd **ConnectionString** property value, details of, 72–73
- Password** property of **DirectoryEntry** class, description of, 418
- Path** property of **DirectoryEntry** class, description of, 418
- PE (portable executable) files, storage of compiled code in, 8
- Peek** and **Receive** methods of **MessageQueue** class, explanation of, 464
- Pending** member of **MessageQueueTransactionStatus** Enum, description of, 483
- perfmon (Performance Monitor), functionality of, 372
- performance degradation, troubleshooting, 371–373
- performance resources optimization table, 369
- Permissions manifest component, description of, 10
- Persist Security Info **ConnectionString** property value, details of, 72–73, 76
- pessimistic locking, explanation of, 266–268
- polymorphism, role in OOP, 506–507
- pooling connections
 - in ADO.NET connected layer, 90–96
 - of data type **OdbcConnection**, 94–96
 - of data type **OleDbConnection**, 93–94
 - of data type **SqlConnection**, 92–93
- Pooling **ConnectionString** property value, details of, 74–75
- pools
 - removing connections from, 92
 - testing for, 92
- populate **DataSet** with **DataAdapter** class, 203–206
- populate **DataTable**, explanation of, 227
- Position** property of **CurrencyManager** class, description of, 540
- positional parameters, executing SQL queries with, 625–627
- Prefix** property
 - of **DataColumn** class, 251
 - of **DataSet** class, 198
 - of **DataTable** class, 220
 - of **XmlReader** class, 163
- Prepare** Command class method, description of, 139
- PRI (procedural referential integrity) in databases, explanation of, 45
- primary keys
 - in databases, definition of, 42
 - role in databases, 42–43
 - setting with Table Designer, 316–317
- PrimaryKey DataTable** class property, description of, 220
- privacy for message queue encryption, explanation of, 497
- private message queues
 - creating on local machines, 454–456
 - versus public message queues, 453–454
 - retrieving, 475–476
 - using with Server Explorer, 288
- private variables, creating for **CUser** class, 515–516
- procedures, using multiple exception handlers in, 345–346
- processing resources, determining for optimization, 370
- programming concepts, review of, 4–14
- properties
 - adding to Active Directory, 427–429
 - checking existence with Active Directory, 426
 - displaying and changing for message queues, 456–459
 - editing with Active Directory, 427
 - overriding in derived classes with OOP, 512–513
 - using local cache for, 426
- Properties** property of **DirectoryEntry** class, description of, 418
- Property Pages dialog box in Table Designer, displaying, 318–319
- PropertyCollection** class in **System.DirectoryServices** namespace, description of, 416
- PropertyValueCollection** class in **System.DirectoryServices** namespace, description of, 416
- Provider **ConnectionString** property value, details of, 74–75

- provider messages, handling in ADO.NET connection layer, 97–98
 - Provider OleDbConnection** class property, details of, 78
 - providers. *See* OLE DB providers
 - public message queues
 - creating, 455
 - versus private message queues, 453–454
 - retrieving, 476–477
 - using with Server Explorer, 288
 - public properties, hooking up to data sets in UserMan example application, 576–578
 - public status of namespaces, advisory about, 13
 - PWD **ConnectionString** property value, details of, 72–73
- Q**
- queries
 - executing from browsers using file-based templates, 606–610
 - executing with Query Designer, 330
 - Query Analyzer, functionality of, 372
 - Query Builder, accessing, 335
 - Query Designer
 - changing join types with, 323–324
 - creating select query with, 321
 - creating SQL statements with, 325
 - diagram pane in, 322–325
 - executing queries with, 330
 - grid pane in, 325–327
 - grouping and sorting output in, 324–325
 - hiding and showing panes in, 328–329
 - removing tables with, 323
 - Results pane in, 328
 - sorting columns with, 324
 - SQL pane in, 327–328
 - using DELETE queries with, 331–332
 - using INSERT queries with, 333
 - using Make Table queries with, 332
 - using SELECT queries with, 331
 - using UPDATE queries with, 331
 - using Verify SQL Syntax facility in, 329–330
 - Query Editor, producing SQL statements with, 335–336
 - query types, examining in Query Designer, 330–334
 - QuoteChar XmlReader** class property, 163
- R**
- RAW SELECT . . . FOR XML statement keyword, description of, 588
 - RCWs (Runtime Callable Wrappers), using with COM types, 277
 - RDBMSes (relational database management systems), planning move to, 375
 - RDN (relative distinguished name), role in Active Directory, 420
 - Read** method of **DataReader** class, 159
 - Read** method of **XmlReader** class, 166
 - read-only data, retrieving from views in code, 401–402
 - read-only mode, generating data-bound Web forms in, 554
 - Read* XmlReader** class methods, 166–167
 - ReadAttributeValue** method of **XmlReader** class, 166
 - ReadCommitted** member of **IsolationLevel** Enum, description of, 102
 - ReadElementString** method of **XmlReader** class, 166
 - ReadEndElement** method of **XmlReader** class, 166
 - ReadInnerXml** method of **XmlReader** class, 166
 - ReadOnly DataColumn** class property, description of, 251
 - ReadOuterXml** method of **XmlReader** class, 167
 - ReadStartElement** method of **XmlReader** class, 167
 - ReadState XmlReader** class property, 163
 - ReadString** method of **XmlReader** class, 167
 - ReadUncommitted** member of **IsolationLevel** Enum, description of, 102
 - ReadXml DataSet** class method, description of, 200
 - ReadXmlSchema DataSet** class method, description of, 201
 - Receive** and **Peek** methods of **MessageQueue** class, explanation of, 464
 - records, role in databases, 36
 - RecordsAffected** property of **DataReader** class, description of, 151
 - Recordset** class versus **DataSet** class, 194–196
 - reference points, saving in transactions, 104–105
 - Referenced Assemblies manifest component, description of, 10
 - ReferenceEquals** method, comparing with, 88

- referential integrity in databases, explanation of, 44–45
- Refresh CurrencyManager** class
 - method, description of, 541
- RefreshCache DirectoryEntry** class
 - method, description of, 419
- Register SQL Server Instance dialog box, displaying, 291
- Registry, explaining absence of, 4
- RejectChanges** method
 - of **DataRow** class, 248
 - of **DataSet** class, 201
 - of **DataTable** class, 223
 - using with **DataSet** objects, 216–218
- relational database design, explanation of, 37–38
- Relational Database Model, explanation of, 36
- relational databases, key aspects of, 35–51
- relational versus hierarchical databases, 33–35
- RelationName DataRelation** class property, description of, 254
- Relations DataSet** class property, description of, 199
- relationship names, showing in database diagrams, 312
- relationships
 - adding with Database Designer, 311
 - creating with Table Designer, 320
 - deleting with Database Designer, 311
 - explanation of, 39
- ReleaseObjectPool** method
 - clearing ODBC object pooling with, 96
 - clearing OLE DB object pooling with, 93–94
 - of **OdbcConnection** class, 84
 - of **OleDbConnection** class, 82
- RemoveAt CurrencyManager** class
 - method, description of, 541
- Rename DirectoryEntry** class method, description of, 420
- RepeatableRead** member of **IsolationLevel** Enum, description of, 102
- Reset** method
 - of **DataSet** class, 201
 - of **DataTable** class, 223
- ResetCommandTimeout** Command
 - class method, description of, 139
- ResolveEntity** method of **XmlReader** class, 167
- result sets
 - appending to streams, 621–622
 - representation in ADO.NET, 261
 - retrieving from Oracle stored procedures, 396
 - retrieving in **XmlReader** class, 620
 - saving to streams, 620–621
- ResultPropertyCollection** class in **System.DirectoryServices** namespace, description of, 416
- ResultPropertyValueCollection** class in **System.DirectoryServices** namespace, description of, 416
- Results pane of Query Designer, functionality of, 328
- ResumeBinding CurrencyManager** class method, description of, 541
- return values and arguments, running stored procedures with, 385–387
- RETURN_VALUE, retrieving from stored procedures, 390–391
- Rollback** method
 - role in aborting manual transactions, 106
 - of Transaction class, 109
- row and column change events, examining order of, 232–233
- row changes, handling in **DataTable** objects, 235–236
- row deletions, handling in **DataTable** objects, 236–238
- row updates, handling in ADO.NET connection layer, 185–188
- RowChanged DataTable** class event, description of, 224
- RowChanging DataTable** class event, description of, 224
- RowDeleted DataTable** class event, description of, 224
- RowDeleting DataTable** class event, description of, 224
- RowError DataRow** class property, description of, 245
- RowFilter DataView** class property, description of, 239
- rows
 - appending to tables with INSERT Results queries, 333
 - copying in **DataTable** objects, 230–232
 - reading in **XmlReader** class, 168–169
 - retrieving from stored procedures, 382
 - retrieving from stored procedures with input arguments, 384–385

- rows (*continued*)
 - retrieving in views, 402
 - role in databases, 36
 - in database tables, definition of, 36
 - Rows DataTable** class property,
 - description of, 220
 - rows of data sources
 - navigating in bound controls, 543
 - retrieving for data-bound controls, 542
 - RowState DataRow** class property,
 - description of, 245
 - RowStateFilter DataView** class property,
 - description of, 239
 - RowUpdated** DataAdapter class event,
 - description of, 175
 - RowUpdating** DataAdapter class event,
 - description of, 175
 - Run On dialog box, displaying, 302
 - running transactions, determining
 - isolation levels of, 107–108
- S**
- sa (system administrator), role in SQL Servers, 293
 - samAccountName retrieving from
 - Active Directory with OLE DB .NET data providers, 432
 - Save** method of Transaction class,
 - description of, 109
 - save points, using with **SqlTransaction** class, 104–105
 - scalar values, returning with stored procedures, 389–391
 - SchemaClassName** property of **DirectoryEntry** class,
 - description of, 418
 - SchemaEntry** property of **DirectoryEntry** class,
 - description of, 418
 - SchemaNameCollection** class in System.DirectoryServices namespace, description of, 416
 - SchemaPath** property, role in executing XPath queries, 630
 - script editing with SQL Editor, performing, 334–337
 - script templates, editing and using with SQL Editor, 336
 - scripts
 - placing in command files, 303–304
 - running in IDEs, 302
 - saving with SQL Editor, 336
 - sealed** keyword in C#, relationship to OOP, 511
 - search Active Directory, explanation of, 422–424
 - search **DataTable**, explanation of, 229–230
 - search **DataView**, explanation of, 243
 - SearchResult** class in
 - System.DirectoryServices** namespace, description of, 416
 - SearchResultCollection** class in
 - System.DirectoryServices** namespace, description of, 416
 - security, using views for, 398
 - Select DataTable** class method,
 - description of, 223
 - SELECT . . . FOR XML AUTO, ELEMENTS output, 589–590
 - Select** method, finding rows in **DataTable** objects with, 229–230
 - SELECT queries, using Query Designer for, 331
 - SelectCommand** DataAdapter class
 - property, description of, 172
 - SelectCommand** property, setting, 178, 189
 - SELECT . . . FOR XML AUTO, XMLDATA output, 590–591
 - SELECT . . . FOR XML RAW output, 590
 - SELECT . . . FOR XML statement keywords, list of, 588–589
 - Serializable** member of **IsolationLevel** Enum, description of, 102
 - Server **ConnectionString** property
 - value, details of, 74–75
 - SERVER **ConnectionString** property
 - value, details of, 74–75
 - Server Explorer
 - adding servers with, 286–287
 - Connect As dialog box options in, 287
 - connecting servers as different users with, 286–287
 - creating database objects with, 285
 - creating SQL Server databases with, 292–295
 - creating triggers with, 407–412
 - deleting and dropping SQL Server databases with, 295
 - deleting data connections with, 285
 - deleting servers with, 295–296
 - explanation of, 25–26
 - handling data connections with, 282–285
 - introduction to, 281–282

- registering SQL Server instances with, 291
 - selecting message queues in, 478
 - unregistering SQL Server instances with, 292
 - using message queues with, 288–290
 - using server resources with, 288–295
 - using with SQL Server databases, 290–295
- server processing resources, performance resources
 - optimization of, 369
- server resources, using with Server Explorer, 288–295
- server-side cursors, explanation of, 262–263
- server-side processing, definition of, 367
- servers
 - adding with Server Explorer, 286–287
 - connecting as different users with Server Explorer, 286–287
 - deleting with Server Explorer, 295–296
- ServerVersion** property
 - of **OdbcConnection** class, 79
 - of **OleDbConnection** class, 78
 - of **SqlConnection** class, 77
- SET ANSI_DEFAULTS T-SQL statement, role in RDBMS migration from SQL Server, 375
- set**, property arguments, checking length of, in **CUser** class, 517
- SetColumnError DataRow** class property, description of, 248
- SetParentRow DataRow** class property, description of, 248
- SetPermissions** method, using with access control for message queues, 501–502
- SimpleStoredProcedure**, displaying output from, 380
- Skip XmlReader** class method, 167
- SMTP (Simple Mail Transfer Protocol), sending mail messages with, 437
- SOAP (Simple Object Access Protocol), based on XML, 5
 - uses HTTP protocol, 5
 - XML Web services based on, 5
- Solution Explorer
 - finding database projects in, 298–299
 - locating Show All Files button in, 535
 - running SQL scripts with, 336–337
 - switching to Class View in, 507
 - sort **DataView**, explanation of, 244–245
 - Sort DataView** class property, description of, 239
 - Sort Order grid pane column in Query Designer, 326
 - Sort Type grid pane column in Query Designer, 326
 - SortOption** class in **System.DirectoryServices** namespace, description of, 416
 - Source** property
 - of **Exception** class, 347
 - of **OdbcError** class, 121
 - of **OdbcException** class, 122
 - of **OleDbError** class, 117
 - of **OleDbException** class, 118
 - sp_addlinkedserver** system stored procedure, resource for, 442–443
 - sp_dropserver** system stored procedure, resource for, 446
 - SQL Dialect, Web site for, 434
 - SQL Editor
 - editing and using script templates with, 336
 - running SQL scripts with, 336–337
 - saving scripts with, 336
 - using, 334–337
 - SQL pane of Query Designer, functionality of, 327–328
 - SQL queries
 - executing with named parameters, 623–625
 - executing with positional parameters, 625–627
 - executing with **SqlXmlCommand** class, 620–624
 - SQL Script database object template, description of, 301
 - SQL scripts, running with SQL Editor, 336–337
 - SQL Server 7.0, advisory about, 367
 - SQL Server databases. *See also* databases
 - accessing Microsoft Server as linked server from, 442–447
 - creating nondefault values and properties for, 294
 - creating simple stored procedures with, 376
 - creating with Server Explorer, 292–295
 - deleting and dropping with Server Explorer, 295
 - functionality of stored procedures in, 392
 - resources for, 371

- SQL Server databases (*continued*)
 - retrieving from, with **SqlXmlAdapter** class, 614–619
 - updating with **SqlXmlAdapter** class, 618–619
 - using Server Explorer with, 290–295, 290–295
 - using stored procedures with, 375–379
- SQL Server Enterprise Manager, viewing stored procedure dependencies with, 391
- SQL Server instances
 - registering with Server Explorer, 291
 - unregistering with Server Explorer, 292
- SQL Server .NET Data Provider versus SQLXML 2.0 Managed classes, 611
- SQL standards, Web site for, 327
- SQL statements
 - creating in SQL pane of Query Designer, 325
 - executing non-row-returning type of, 632
 - producing with Query Editor, 335–336
- SQL syntax, verifying with Query Designer, 329–330
- SqlCommand** class
 - properties of, 130–131
 - usage of, 124
- SqlCommand** object, instantiating, 128–129
- SqlCommandBuilder** class, using, 190
- SqlConnection** class, **BeginTransaction** method for, 100–101
- SqlConnection** class exceptions
 - catching, 357
 - handling, 110–115
 - methods of, 80–81
 - properties of, 77
- SqlConnection** data type, pooling connections for, 92–93
- SqlConnection** managed connection, explanation of, 63–64
- SqlDataAdapter** class
 - instantiating, 176–178
 - setting command properties of, 179–180
- SqlDataReader** class and object, instantiating, 147–148
- SqlInfoMessageEventArgs** class
 - argument, 97–98
 - Message** property, 98
 - Source** property, 98
- SQLOLEDB provider, description of, 61
- SQLState** property
 - of **OdbcError** class, 121
 - of **OleDbError** class, 117
- SqlTransaction** class
 - nesting transactions with, 104–105
 - using transaction save points with, 104–105
- SQLXML 2.0
 - advisory about direct URL queries in, 606
 - advisory about using template string method with, 617
 - configuring ISAPI extension for, 593–598
 - executing directory queries from browser in, 601–606
 - executing file-based XML templates from code in, 617
 - executing queries from browsers with file-based templates, 606–610
 - installing, 592–593
 - introduction to, 591
 - introduction to Managed Classes in, 611
 - querying tables in, 601–603
 - sample connection string in, 614
 - sample template query with parameters in, 608–609
- SQLXML 2.0 Managed Classes, using DiffGrams and UpdateGrams with, 619
- SqlXmlAdapter** class
 - examining, 612–619
 - retrieving SQL Server data with, 614–619
 - updating SQL Server data with, 618–619
- SqlXmlAdapter** objects, instantiating, 613–614
- SqlXmlCommand** class
 - executing SQL queries with, 620–623
 - introduction to, 619
 - populating **DataSets** with, 615–616
 - updating data sources with, 631–635
- SqlXmlCommand** objects, instantiating, 619
- SqlXmlParameter** class, examining, 612
- StackTrace** property
 - of **Exception** class, 347
 - of **OdbcException** class, 122
 - of **OleDbException** class, 118
- stand-alone applications versus distributed applications, 368
- standard HTML programming model for ASP.NET, explanation of, 5

- State**, comparing to **ConnectionState**, 89
 - State** property
 - of **OdbcConnection** class, 79
 - of **OleDbConnection** class, 78
 - of **SqlConnection** class, 77
 - StateChange** **Connection** class event, details of, 85
 - static cursors, explanation of, 264
 - storage capability of databases, understanding, 32
 - Stored Procedure Script database object template, description of, 301
 - stored procedures, 372–373
 - creating, 375–379
 - creating with arguments, 382–384
 - creating with arguments and return values, 385–387
 - introduction to, 373
 - reasons for use of, 374–375
 - renaming, 391
 - retrieving rows and output values from, 388–389
 - retrieving rows from, 382
 - running from code, 380–382
 - running from IDE, 379–380
 - running with arguments from IDE, 384
 - running with Oracle, 392–397
 - syntax testing of, 387
 - using RETURN statement with, 389–391
 - using with arguments, 384–385
 - viewing dependencies for, 391
 - streams
 - appending result sets to, 621–622
 - saving result sets to, 620–621
 - strong typing versus weak typing, example of, 196
 - SuspendBinding CurrencyManager** class
 - method, description of, 541
 - syntax for creating namespaces, 12
 - system-generated message queues, examining, 485–490
 - system journals, using with message queues, 486–487
 - system message queues, using with Server Explorer, 288
 - System** .NET root namespace, explanation of, 11–12
 - System Stored Procedures. *See* stored procedures, 372–373
 - System*** data-related namespaces, list of, 58
 - System.Data** namespace, description of, 58
 - System.Data.OleDb** namespace, description of, 58
 - System.Data.SqlClient** namespace, description of, 58
 - System.DirectoryServices** namespace, examining in Active Directory, 415–417
 - SystemException** exception type, 350
- T**
- *.tlb (type libraries), role in generating metadata for COM types, 277
 - T-SQL, adding linked Microsoft Exchange Server with, 442–443
 - tabbed documents mode, explanation of, 16
 - Table Designer
 - adding columns to tables with, 315–316
 - adding constraints with, 319–320
 - adding indexes and keys with, 317–319
 - creating relationships with, 320
 - creating tables with, 314–315
 - setting primary keys with, 316–317
 - using triggers with, 320
 - Table grid pane column in Query Designer, 326
 - Table** property
 - of **DataColumn** class, 251
 - of **DataRow** class property, description of, 245
 - of **DataView** class, 239
 - Table Script database object template, description of, 301
 - table views, changing for database diagrams, 313
 - TableMappings** **DataAdapter** class property, description of, 172
 - TableName DataTable** class property, description of, 220
 - tables
 - adding and showing in tables with Query Designer, 322
 - adding with Database Designer, 309
 - creating with Database Designer, 309–313
 - creating with Table Designer, 314–315
 - deleting and removing with Database Designer, 309
 - in databases, definition of, 36
 - querying with SQLXML 2.0, 601–603
 - removing with diagram pane of Query Designer, 323
 - role in databases, 36
 - sizing automatically in database diagrams, 313

- tables and columns, choosing in Data Form Wizard, 532
- tables and views, choosing in Data Form Wizard, 531
- Tables DataSet** class property, description of, 199
- TargetSite Exception** class property, description of, 347
- TargetSite** property of **OdbcException** class, details of, 122
- TargetSite** property of **OleDbException** class, details of, 118
- Task List in VS .NET IDE, explanation of, 27–28
- tblLog table in UserMan database schema, explanation of, 52
- tblRights table in UserMan database schema, explanation of, 52
- tblUser columns for CUser class, list of, 514
- tblUser table in UserMan database schema, explanation of, 52
- tblUser_Update trigger, displaying, 409
- tblUserRights table in UserMan database schema, explanation of, 52
- TCP/IP (Transmission Control Protocol/Internet Protocol), role in connectionless and connection oriented programming, 450
- TDS (tabular data stream) protocol, role in .NET Data Provider for SQL Server, 59
- template string method, advisory in SQLXML 2.0, 617
- templates
 - advisory about, 606
 - executing as string templates, 632–633
 - specifying with SQLXML 2.0, 602–603
- text editors
 - using in VS .NET IDE, 24
 - using SQL Editor, 334–337
- throw** statements, using, 356–357
- Toolbox in VS .NET IDE, explanation of, 27
- tools, modifying in VS .NET IDE, 24–25
- ToString** method
 - of **OdbcException** class, 123
 - of **OdbcConnection** class, 84
 - of **OleDbConnection** class, 82
 - of **OleDbException** class, 119
 - of **SqlConnection** class, 81
- transaction boundaries, defining in ADO.NET connected layer, 99
- Transaction class
 - methods of, 109
 - properties of, 108
- Transaction** property
 - of **OdbcCommand** class, 136
 - of **OleDbCommand** class, 133
 - of **SqlCommand** class, 131
- transaction save points, using with **SqlTransaction** class, 104–105
- transactional message queues, creating, 481
- transactional private message queues, creating, 456
- transactions
 - aborting for message queues, 483
 - beginning with nondefault isolation levels, 102
 - committing for message queues, 482–483
 - ending for message queues, 482–483
 - functionality of, 98
 - locking data at data sources with, 266–268
 - nesting with **OdbcTransaction** class, 106
 - nesting with **OleDbTransaction** class, 105–106
 - nesting with **SqlTransaction** class, 104–105
 - saving reference points in, 104–105
 - starting for message queues, 481
 - using with message queues, 484–485
- trigger and catch concurrency violation, example of, 269–270
- Trigger Script database object template, description of, 301
- triggers
 - advisory about, 406
 - creating, 407–412
 - introduction to, 403–404
 - invoking and catching exception raised example, 410–411
 - locating after saving, 409
 - reasons for use of, 406
 - using with Table Designer, 320
 - viewing source for, 412
- Trusted_Connection **ConnectionString** property value, details of, 74–75
- try** blocks, role in filtering exceptions, 355
- try** statements, role in exception handling, 344
- Type Reference manifest component, description of, 10

- typed data sets
 - in ADO.NET disconnected layer, 196
 - creating with DataSet Designer, 338–340
 - creating with XML Designer, 338
- U**
- UID **ConnectionString** property value, details of, 74–75
- Unique DataColumn** class property, description of, 251
- unmanaged code, definition of, 6
- Unspecified** member of **IsolationLevel** Enum, description of, 102
- untyped data sets in ADO.NET disconnected layer, explanation of, 196
- update data source using **DataAdapter** class, 206–209
- Update** **DataAdapter** class method, description of, 174
- UPDATE queries, using Query Designer for, 331
- UpdateCommand** **DataAdapter** class property, description of, 172
- UpdatedRowSource** property
 - of **OdbcCommand** class, 137
 - of **OleDbCommand** class, 134
 - of **SqlCommand** class, 131
 - throws **ArgumentException** exception, 145–146
- UpdateGrams, advisory about using with SQLXML 2.0 Managed classes, 619
- URLs (uniform resource locators), advisory about querying with, 606
- UsePropertyCache** property of **DirectoryEntry** class, description of, 418
- User ID **ConnectionString** property value, details of, 74–75
- user permissions, setting programmatically, 501
- user properties
 - adding to Active Directory, 428
 - editing with Active Directory, 427
 - manipulating with Active Directory, 428–429
- UserMan Properties dialog box, displaying, 433–434
- UserMan database schema, 51–52
- UserMan example application
 - adding Active Directory objects to, 579–581
 - adding command object parameters to, 575–576
 - building as **DataSet** object, 256–261
 - calling Active Directory class in, 581
 - conformity to 5NF, 53
 - creating additional objects for, 581–582
 - creating select query in, 321
 - creating stored procedures for
 - accessing database tables in, 584
 - creating triggers with, 407–412
 - creating views in, 399–400
 - creating Web client for, 583
 - creating Windows client for, 582
 - downloading classes for, 567
 - exception handling in, 586
 - exposing functionality with Web services, 585–586
 - filling data sets in, 576
 - hooking up public properties to data sets in, 576–578
 - identifying information and objects in, 565–566
 - instantiating and initializing data adapters in, 574–575
 - instantiating command objects in, 573–574
 - instantiating **DataSet** objects in, 574
 - introduction to, 51–53
 - logging to event log in, 583–584
 - opening and closing connection to, 570–573
 - optimizing, 583–586
 - passing connection object to various classes in, 584
 - retrieving contacts from Exchange Server 2000, 438–439
 - schema for, 51–52
 - securing Password column in user table of, 583
 - setting up database security for, 584–585
 - setting up triggers for enforcing business rules in, 584
 - specifying parent classes in, 578
 - using constants for table and column names in, 585
 - using local transactions with, 585
 - Web site for, 57
- Username** property of **DirectoryEntry** class, description of, 418
- userPrincipalName retrieving from Active Directory with OLE DB .NET data providers, 432
- users, retrieving SID with Active Directory, 435–436

using statement
 importing namespaces into classes
 with, 58
 uspGetRETURN_VALUE stored procedure, displaying, 390
 uspGetUsers stored procedures, displaying, 381
 uspGetUsersAndRights stored procedure, displaying, 386–388
 uspGetUsersByLastName stored procedure, displaying, 383

V

Value property
 of **SqlParameter** class, 612
 of **XmlReader** class, 163
 VBA (Visual Basic for Applications),
 implications of CLR for, 7
 VBScript, implications of CLR for, 7
 view ports, moving in database diagrams, 312–313
 View Script database object template,
 description of, 301
 views
 choosing in Data Form Wizard, 531
 creating, 399–400
 creating on linked servers, 447
 introduction to, 397
 manipulating data from code in,
 402–405
 reasons for use of, 398
 restrictions of, 398
 running from IDE, 400–401
 saving, 399
 using from code, 401–405
 virtual directory in SQLXML 2.0
 managing with SQLXML 2.0, 593–599
 restarting after making changes to,
 600
virtual keyword in C#, relationship to
 OOP, 511
 Virtual Name Configuration dialog box,
 displaying in SQLXML 2.0, 605
 Visible Analyst DB Engineer, Web site
 for, 37
 Visio for Enterprise Architect, designing
 databases with, 305
 viwUser view, displaying, 403
 VS .NET IDE (Integrated Development
 Environment). *See also* IDEs
 built-in Object Browser for, 19–20
 built-in Web browser functionality
 for, 17
 Command Window modes for, 18–19
 integrated debugger in, 20–21
 integrated Help system in, 21–22

interface modes for, 16
 macros in, 22–23
 modifying, 24–25
 Server Explorer in, 25–26
 setting up data connections with, 26
 sharing with all .NET languages, 13
 Task List in, 27–28
 text editors in, 24
 Toolbox in, 27
 upgraded deployment tools in, 23
 VSA (Visual Studio for Applications),
 implications of CLR for, 7

W

weak typing versus strong typing, example
 of, 196
 Web browser functionality feature of VS
 .NET IDE, explanation of, 17
 Web clients, creating for UserMan example
 application, 583
 Web Forms programming model for
 ASP.NET, explanation of, 5
 Web forms, using data-bound controls
 with, 545–551
 Web services, exposing functionality
 with, 585–586
 Web sites
 Active Directory, 415, 434
 Antechinus C# Programming Editor,
 15
 Apress, 57
 DiffGrams, 633
 IMS (Information Management
 System), 34
 LDAP display names for users, 425
 LDAP query filters, 425
 modeling relational databases, 37
 ODBC (Open Data Base
 Connectivity), 96
 ODBC standard escape sequences,
 137
 Oracle Database Project, 392
 SQL Dialect, 434
 SQL standards, 327
 SQLXML 2.0 download, 592
 UserMan, 57
 UserMan Web client, 583
 UserMan Windows client, 582
 for VS .NET IDE Command Window,
 19
 X.500 directory standard, 413
 XPath Web site, 605
 Web user controls, data binding in,
 557–561
 Windows form controls, binding to data
 sources, 537–538

- Windows forms
 - creating data-bound controls for, 544
 - examining binding context for, 527–528
 - role of **BindingContext** object in, 527, 538–539
 - role of **CurrencyManager** object in, 527
 - using data-bound controls with, 527
 - Workstation ID **ConnectionString** property value, details of, 74–75
 - WorkstationId SqlConnection** class property, details of, 77
 - wrappers
 - definition of, 505
 - reasons for use of, 505
 - wrapping databases, 513–524
 - WriteXml DataSet** class method, description of, 201
 - WriteXmlSchema DataSet** class method, description of, 201
- X**
- X.500 directory standard, Web site for, 413
 - XML Designer, creating typed data sets with, 338
 - XML documents, writing to disk, 617
 - XML (eXtensible Markup Language), using with **DataSet** class in ADO.NET disconnected layer, 195–196
 - XML Parser feature of SQLXML 2.0, resource for, 593
 - XML templates, sample for updating data sources, 632
 - XML Web services programming model for ASP.NET, explanation of, 5
 - XMLDATA SELECT . . . FOR XML statement keyword, description of, 588
 - XmlException**, handling for **XmlReader** class, 170
 - XmlLang XmlReader** class property, 163
 - XmlNodeReader** class, usage of, 161
 - XmlReader** class
 - closing, 169
 - exception handling, 170
 - methods of, 164–167
 - properties of, 161–163
 - reading rows in, 168–169
 - retrieving result sets in, 620
 - usage of, 161–168
 - XmlReader** objects, declaring and instantiating, 168
 - XmlSpace XmlReader** class property, 163
 - XmlTextReader** class, usage of, 161
 - XmlValidatingReader** class, usage of, 161
 - XPath queries, executing, 627–631
 - XPath, querying database tables with, 603
 - XPath Web site, 605
 - .xsd files, creation by Data Form Wizard, 534