

# ***Using UltraDev***

---

## Trademarks

Afterburner, AppletAce, Attain, Attain Enterprise Learning System, Attain Essentials, Attain Objects for Dreamweaver, Authorware, Authorware Attain, Authorware Interactive Studio, Authorware Star, Authorware Synergy, Backstage, Backstage Designer, Backstage Desktop Studio, Backstage Enterprise Studio, Backstage Internet Studio, Design in Motion, Director, Director Multimedia Studio, Doc Around the Clock, Dreamweaver, Dreamweaver Attain, Drumbeat, Drumbeat 2000, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, FreeHand Graphics Studio, Generator, Generator Developer's Studio, Generator Dynamic Graphics Server, Knowledge Objects, Knowledge Stream, Knowledge Track, Lingo, Live Effects, Macromedia, Macromedia M Logo & Design, Macromedia Flash, Macromedia Xres, Macromind, Macromind Action, MAGIC, Mediamaker, Object Authoring, Power Applets, Priority Access, Roundtrip HTML, Scriptlets, SoundEdit, ShockRave, Shockmachine, Shockwave, Shockwave Remote, Shockwave Internet Studio, Showcase, Tools to Power Your Ideas, Universal Media, Virtuoso, Web Design 101, Whirlwind and Xtra are trademarks of Macromedia, Inc. and may be registered in the United States or in other jurisdictions including internationally. Other product names, logos, designs, titles, words or phrases mentioned within this publication may be trademarks, servicemarks, or tradenames of Macromedia, Inc. or other entities and may be registered in certain jurisdictions including internationally.

This guide contains links to third-party Web sites that are not under the control of Macromedia, and Macromedia is not responsible for the content on any linked site. If you access a third-party Web site mentioned in this guide, then you do so at your own risk. Macromedia provides these links only as a convenience, and the inclusion of the link does not imply that Macromedia endorses or accepts any responsibility for the content on those third-party sites.

## Apple Disclaimer

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE.

Copyright © 2000 Macromedia, Inc. All rights reserved. This manual may not be copied, photocopied, reproduced, translated, or converted to any electronic or machine-readable form in whole or in part without prior written approval of Macromedia, Inc.  
Part Number ZUD40M100

## Acknowledgments

Project Management: Sheila McGinn

Writing: Charles Nadeau

Editing: Anne Szabla and Lisa Stanziano

Production Management: John "Zippy" Lehnus

Multimedia Design and Production: Aaron Begley and Noah Zilberberg

Print Production: Chris Basmajian, Paul Benkman, Caroline Branch, and Rebecca Godbois

Web Editing and Production: Jane Flint DeKoven and Jeff Harmon

Special thanks to Margaret Dumas, Peter Fenczik, Nick Halbakken, S Fred Golden, Craig Jennings, Ken Karleskint, Lars Rydell, Julie Thompson, and the Dreamweaver UltraDev engineering and QA teams.

First Edition: November 2000

Macromedia, Inc.  
600 Townsend St.  
San Francisco, CA 94103

# CONTENTS

## INTRODUCTION

<a href="#">Getting Started</a> .....	7
Typographical conventions .....	7
What you need to get started.....	8
Where to start .....	10
Installing Dreamweaver UltraDev .....	10
Quick start for Windows users .....	11
Quick start for Macintosh users .....	16
Configuring your system .....	22
What's new in UltraDev 4.....	29
UltraDev resources .....	30
Learning Dreamweaver UltraDev .....	33
Web application resources .....	34
Accessibility and Dreamweaver .....	34

## CHAPTER 1

<a href="#">Dreamweaver UltraDev Tutorial</a> .....	35
Tutorial quick start for Windows users .....	36
Tutorial quick start for Macintosh users.....	41
Define a local site .....	47
Define a remote site.....	48
Configure UltraDev to work with your application server .....	49
Create a database connection.....	50
Create a simple search/results page set .....	56
Create an advanced search/results page set.....	64
Create a detail page .....	67
Create an insert page .....	72

<b>CHAPTER 2</b>	
UltraDev Basics.....	75
About dynamic pages .....	76
About the UltraDev workflow .....	77
About the UltraDev working environment .....	81
<b>CHAPTER 3</b>	
Connecting to a Database .....	89
Creating a database connection for an ASP application .....	89
Creating a database connection for a ColdFusion application .....	98
Creating a database connection for a JSP application .....	101
Editing or deleting database connections .....	104
Creating a connection for UltraDev use .....	105
<b>CHAPTER 4</b>	
Defining UltraDev Data Sources .....	109
Defining a recordset as a data source .....	110
Defining browser-submitted data sources for ASP pages .....	110
Defining browser-submitted data sources for ColdFusion pages .....	113
Defining browser-submitted data sources for JSP .....	117
Defining session variables as data sources .....	117
Defining application variables as data sources .....	118
Defining a stored procedure server object as a data source .....	119
Defining JavaBeans as data sources (JSP only) .....	120
Caching data sources .....	123
Changing or deleting data sources.....	123
<b>CHAPTER 5</b>	
Creating a Recordset.....	125
Defining a recordset .....	125
Invoking a stored procedure .....	131
Copying a recordset to another page .....	131
Editing or deleting a recordset as a data source .....	132

## **CHAPTER 6**

### **Adding Dynamic Content . . . . . 133**

Making text dynamic . . . . .	134
Making images dynamic . . . . .	136
Making form objects dynamic . . . . .	138
Making HTML attributes dynamic . . . . .	142
Making ActiveX, Flash, and other object parameters dynamic . . . . .	144
Changing dynamic content . . . . .	144
Deleting dynamic content . . . . .	145

## **CHAPTER 7**

### **Displaying Database Records . . . . . 147**

Creating recordset navigation links . . . . .	147
Showing and hiding regions . . . . .	151
Displaying multiple records . . . . .	151
Building a record counter . . . . .	153
Creating a master/detail page set . . . . .	156
Editing server behaviors on a page . . . . .	164

## **CHAPTER 8**

### **Building Pages That Search Databases . . . . . 165**

Creating the search page . . . . .	166
Building the results page . . . . .	167
Creating a detail page for a results page . . . . .	173
Working with related pages . . . . .	178

## **CHAPTER 9**

### **Building Pages That Edit Database Records . . . . 181**

Building a page to insert records . . . . .	182
Building a page to update records . . . . .	187
Building a page to delete a record . . . . .	193

## **CHAPTER 10**

### **Building Pages That Restrict Access to Your Site . . . . . 199**

Building a registration page . . . . .	200
Building a log-in page . . . . .	204
Building a page only authorized users can access . . . . .	207

## **CHAPTER 11**

Customizing UltraDev .....211

Editing and creating data formats .....211

Installing more server behaviors .....212

Creating server behaviors .....212

Editing server behaviors .....221

Creating other UltraDev extensions .....224

## **APPENDIX A**

Beginner's Guide to Databases ..... 225

About databases .....225

About database connections .....227

## **APPENDIX B**

Detailed Requirements for Building Web  
Applications ..... 233

Requirements for ASP developers .....233

Requirements for ColdFusion developers .....237

Requirements for JSP developers .....240

## **APPENDIX C**

Installing Microsoft Personal Web Server ..... 243

Installing PWS .....243

Configuring PWS .....244

## **APPENDIX D**

Installing Allaire ColdFusion Server ..... 245

Installing and configuring ColdFusion Server .....246

## **APPENDIX E**

Setting Up a DSN in Windows ..... 249

## **APPENDIX F**

SQL Primer .....251

Including an entire table .....251

Limiting the number of columns .....252

Limiting the number of records .....252

Sorting the records .....254

**INDEX** ..... 255



# INTRODUCTION

## Getting Started

.....

Macromedia Dreamweaver UltraDev is a professional environment for building Web applications. A Web application is a collection of pages that interact with each other and with various resources on a Web server, including databases.

UltraDev is also a professional editor for creating and managing Web sites and pages. Because it incorporates all of the Dreamweaver page design and site management tools, UltraDev makes it easy to create, manage, and edit cross-platform, cross-browser Web pages.

UltraDev is fully customizable. You can create your own objects, commands, and server behaviors, modify menus and keyboard shortcuts, and even write scripts to extend UltraDev with new actions, behaviors, and property inspectors.

## Typographical conventions

The following typographical conventions are used in this guide:

- `code font` indicates scripts, SQL statements, HTML tag and attribute names, and literal text used in examples.
- *Italic code font* indicates replaceable items in code.

## What you need to get started

To build Web applications in UltraDev, you need the following:

- A Web server
- An application server that runs on your Web server, or a Web server that doubles as an application server, such as Microsoft Personal Web Server (PWS) or Internet Information Server (IIS)
- A database or database system
- A database driver that supports your database system

The exact requirements vary depending on whether you use UltraDev to create Active Server Pages (ASP) applications, ColdFusion applications, or JavaServer Pages (JSP) applications. For more information on these technologies, see “About dynamic pages” on page 76.

This section describes some typical configurations that work with a Microsoft Access database. (A Microsoft Access database is used in the UltraDev tutorial.)

For detailed requirements for ASP, ColdFusion, or JSP developers, see “Detailed Requirements for Building Web Applications” on page 233.

### Typical system configurations for ASP developers

Here are typical system configurations of ASP developers working with Microsoft Access databases:

UltraDev system	Web server	App server	Database driver
Windows 95, 98, NT Workstation	PWS running locally	PWS running locally	Microsoft Access Driver (ODBC)
Windows NT Server, 2000	IIS running locally	IIS running locally	Microsoft Access Driver (ODBC)
Macintosh	IIS running remotely	IIS running remotely	Microsoft Access Driver (ODBC)

For more detailed information, see “Requirements for ASP developers” on page 233.

For instructions on installing PWS on your local computer, see “Installing Microsoft Personal Web Server” on page 243. If you’re a Windows 2000 user, you can install IIS 5.0, which is included in the Windows 2000 package.



## Typical system configurations for ColdFusion developers

Here are typical system configurations of ColdFusion developers working with Microsoft Access databases:

UltraDev system	Web server	App server	Database driver
Windows 95, 98, NT Workstation	PWS running locally	ColdFusion Server running locally	Microsoft Access Driver (ODBC)
Windows NT Server, 2000	IIS running locally	ColdFusion Server running locally	Microsoft Access Driver (ODBC)
Macintosh	IIS running remotely	ColdFusion Server running remotely	Microsoft Access Driver (ODBC)

For more detailed information, see “Requirements for ColdFusion developers” on page 237.

For instructions on installing PWS on your local computer, see “Installing Microsoft Personal Web Server” on page 243. For instructions on installing the single-user copy of ColdFusion Server on the UltraDev CD, see “Installing Allaire ColdFusion Server” on page 245.

## Typical system configurations for JSP developers

Here are typical system configurations of JSP developers working with Microsoft Access databases:

UltraDev system	Web server	App server	Database driver
Windows 95, 98, NT Workstation	PWS running locally	WebSphere or JRun running locally	JDBC-ODBC Bridge with Microsoft Access Driver (ODBC)
Windows NT Server, 2000	IIS running locally	WebSphere or JRun running locally	JDBC-ODBC Bridge with Microsoft Access Driver (ODBC)
Macintosh	IIS running remotely	WebSphere or JRun running remotely	JDBC-ODBC Bridge with Microsoft Access Driver (ODBC)

For more detailed information, see “Requirements for JSP developers” on page 240.

For instructions on installing PWS on your local computer, see “Installing Microsoft Personal Web Server” on page 243.

Make sure your system has a JDBC driver for Access databases or a JDBC-ODBC bridge driver. Also make sure the Java Development Kit (JDK) is installed on your computer. You can download the JDK from the Sun Web site at <http://java.sun.com/products/jdk/1.1/>.

## Where to start

Start by installing UltraDev. For instructions, see “Installing Dreamweaver UltraDev” on page 10.

Next, configure your system.

The easiest way to configure your system is to obtain a trial account with a Macromedia recommended Internet service provider (ISP). For more information, see the Macromedia Web site at <http://www.macromedia.com/software/ultradev/isp/>.

If you want to configure your own system, you can quickly get started by reading the following sections:

- “Quick start for Windows users” on page 11
- “Quick start for Macintosh users” on page 16

The quick start sections assume you’re using a Microsoft Access database with a server running on a Windows computer (either locally for Windows users, or remotely for Macintosh users). If you decide not to use these configurations, read the instructions in “Configuring your system” on page 22.

## Installing Dreamweaver UltraDev

The following hardware and software is required to run Dreamweaver UltraDev.

### For Microsoft Windows:

- An Intel Pentium processor or equivalent, 166 MHz or faster, running Windows 95, Windows 98, Windows Me, Windows NT 4.0 with Service Pack 5, or Windows 2000.
- 64 MB of random-access memory (RAM) plus 170 MB of available disk space.
- Macromedia Flash Player to view the Guided Tours and Lessons. A Flash Player installer file is located on the CD, or you can download it from the Macromedia Web site at <http://www.macromedia.com/software/flashplayer/downloads/>.

**For the Macintosh:**

- A Power Macintosh running Mac OS 8.6 or 9.x.
- 64 MB of random-access memory (RAM) plus 130 MB of available disk space.
- Macromedia Flash Player to view the Guided Tours and Lessons. A Flash Player installer file is located on the CD, or you can download it from the Macromedia Web site at <http://www.macromedia.com/software/flashplayer/downloads/>.

Follow these steps to install Dreamweaver UltraDev on your computer.

**To install Dreamweaver UltraDev:**

- 1 Insert the Dreamweaver UltraDev CD into the computer's CD-ROM drive.
- 2 Choose from the following options:
  - In Windows, choose Start > Run. Click Browse and locate the UltraDev installer file on the CD. Click OK in the Run dialog box to begin the installation.
  - On the Macintosh, double-click the Dreamweaver UltraDev Installer icon.
- 3 Follow the onscreen instructions.
- 4 If prompted, restart your computer.

## Quick start for Windows users

This section describes the quickest way to start building Web applications in UltraDev for Windows. The section guides you through the following steps:

- Configuring your computer
- Configuring UltraDev
- Setting up a data source name (DSN) on your computer
- Creating a database connection

More detailed explanations of the concepts and procedures described in this section are available in the rest of this guide or in Help.

Many different system configurations are possible. For the sake of simplicity, this section assumes you use a Microsoft Access database. If you don't have a Microsoft Access database, you can use the sample database that comes with the UltraDev tutorial. With UltraDev installed on your system, the database is located on your hard disk in the Tutorial folder in the Dreamweaver UltraDev application folder.

## Configuring your computer

Here is the simplest system configuration for Windows users.

### To configure your computer:

- 1 If you're a Windows 95, 98, or NT Workstation user, install Microsoft Personal Web Server (PWS).

For detailed instructions, see “Installing Microsoft Personal Web Server” on page 243.

- 2 If you're a Windows NT Server or Windows 2000 user, make sure Internet Information Server (IIS) is installed and running on your system.

IIS is the full-featured version of PWS. It should already be installed on your system. If not, install it or ask your system administrator to install it for you.

- 3 If you want to develop a ColdFusion application, install ColdFusion Server on your system.

For instructions, see “Installing Allaire ColdFusion Server” on page 245.

- 4 If you want to develop a JSP application, install the following components on your system:

- Java 2 SDK, Standard Edition, for Windows

Sun JDBC-ODBC Bridge driver is installed automatically when you install the SDK. You can download the SDK from the Sun Web site at <http://java.sun.com/j2se/>.

- An application server that implements Sun JavaServer Pages 1.0 specification

For more information, see “JSP application server” on page 241.

- 5 In Windows, create a new folder to hold working copies of your site files.

Here's an example:

*c:\Sites\MyWorkingSite*

- 6 In Windows, create a subfolder in the *c:\inetpub\wwwroot* folder and give it a name that describes your site.

Here's an example:

*c:\inetpub\wwwroot\MyPublishedSite*

## Configuring UltraDev

Here is how to configure UltraDev to work with the system configuration outlined in the previous section.

**To configure UltraDev:**

- 1 Launch UltraDev, choose Site > New Site, and complete the Local Info dialog box as follows:

Site Name: *MyFirstSite*

Local Root Folder: *c:\Sites\MyWorkingSite*

- 2 Click Remote Info and complete the dialog box as follows:

Access: Local/Network

Remote Folder: *c:\Inetpub\wwwroot\MyPublishedSite*

- 3 If you want to develop an ASP site, click Application Server and complete the dialog box as follows:

Server Model: ASP 2.0

Scripting Language: VBScript or JavaScript

Page Extension: .asp

Access: Local/Network

Remote Folder: *c:\Inetpub\wwwroot\MyPublishedSite*

URL Prefix: *http://localhost/MyPublishedSite*

- 4 If you want to use ColdFusion Server as your application server, click Application Server and complete the dialog box as follows:

Server Model: ColdFusion 4.0

Scripting Language: CFML

Page Extension: .cfm

Access: Local/Network

Remote Folder: *c:\Inetpub\wwwroot\MyPublishedSite*

URL Prefix: *http://localhost/MyPublishedSite*

- 5 If you want to use a JSP application server, click Application Server and complete the dialog box as follows:

Server Model: JSP 1.0

Scripting Language: Java

Page Extension: .jsp

Access: Local/Network

Remote Folder: *c:\Inetpub\wwwroot\MyPublishedSite*

URL Prefix: *http://localhost/MyPublishedSite*

- 6 Click OK.

## Setting up a DSN on the local computer

A data source name (DSN) is a kind of shortcut used to establish a database connection.

If you're using the tutorial database, a DSN called "CompassTravel" was created for the database when you installed UltraDev.

If you're using another Microsoft Access database, set up a DSN for it. For instructions, see "Setting Up a DSN in Windows" on page 249.

## Creating a database connection

A database connection is a set of parameters you define that allows your Web application to find and use a database. You defined the parameters when you set up your DSN. Next, use the DSN to create the database connection.

You create different database connections for an ASP, ColdFusion, and JSP site.

### To create a database connection for an ASP application:

- 1 In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

- 2 Click the New button and select Data Source Name (DSN) from the pop-up menu.

The Data Source Name (DSN) dialog box appears.

- 3 Enter a name for the new connection.

- 4 Select a DSN.

If you're using the tutorial database, select CompassTravel from the list of DSNs. UltraDev created the CompassTravel DSN during installation. If you're using another database, select the DSN you created for the database.

- 5 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see "Configuring UltraDev" on page 12).

- 6 Click OK.

Your new connection should now appear in the Connections dialog box.

- 7 Click Done to close the Connections dialog box.

You can now start building your Web application. To learn more, see "Learning Dreamweaver UltraDev" on page 33.

**To create a database connection for a ColdFusion application:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
  - 2 Click the New button and select Data Source Name from the pop-up menu.  
UltraDev prompts you for your ColdFusion user name and password.
  - 3 Enter the user name and password you use to log in to the ColdFusion Administrator.  
UltraDev retrieves the ColdFusion DSNs and displays the Data Source Name dialog box.
  - 4 Enter a name for the new connection.
  - 5 Select a DSN.  
If you're using the tutorial database, select CompassTravel from the list of DSNs. UltraDev created the CompassTravel DSN during installation. If you're using another database, select the DSN you created for the database.
  - 6 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see "Configuring UltraDev" on page 12).
  - 7 Click OK.  
Your new connection should now appear in the Connections dialog box.
  - 8 Click Done to close the Connections dialog box.
- You can now start building your Web application. To learn more, see "Learning Dreamweaver UltraDev" on page 33.

**To create a database connection for a JSP application:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select "ODBC Database (Sun JDBC-ODBC Driver)" from the pop-up menu.  
The ODBC Database (Sun JDBC-ODBC Driver) dialog box appears.
- 3 Enter a name for the new connection.
- 4 If you're using the tutorial database, replace the [odbc\_dsn] placeholder in the URL box with CompassTravel.  
The URL box should look like this:  
`jdbc:odbc:CompassTravel`

5 If you're using another database, replace the `[odbc dsn]` placeholder in the URL box with the DSN you created for the database.

6 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see "Configuring UltraDev" on page 12).

7 Click OK.

Your new connection should now appear in the Connections dialog box.

8 Click Done to close the Connections dialog box.

You can now start building your Web application. To learn more, see "Learning Dreamweaver UltraDev" on page 33.

## Quick start for Macintosh users

This section describes the quickest way to start building Web applications in UltraDev for Macintosh. The section guides you through the following steps:

- Configuring your server
- Configuring UltraDev on the Macintosh
- Creating a database connection

More detailed explanations of the concepts and procedures described in this section are available in the rest of this guide or in Help.

Many different system configurations are possible. For the sake of simplicity, this section assumes you use a Microsoft Access database. If you don't have a Microsoft Access database, you can use the sample database that comes with the UltraDev tutorial. With UltraDev installed on your system, the database is located on your hard disk in the Tutorial folder in the Dreamweaver UltraDev application folder.

Because popular Web servers and application servers do not yet support the Macintosh, you need another computer to run the server software. This section assumes you have access to a Windows NT Server or Windows 2000 computer running Internet Information Server (IIS), a common commercial Web server.



## Configuring your server

This section assumes you have access to a Windows NT Server or Windows 2000 computer.

### To configure the server:

- 1 If not already done, install Internet Information Server (IIS) on the server.  
IIS should already be installed on the system. If not, install it or ask your system administrator to install it for you.
- 2 If you want to develop a ColdFusion application, install ColdFusion Server on the server.  
For instructions, see “Installing Allaire ColdFusion Server” on page 245.
- 3 If you want to develop a JSP application, install the following components on the Windows server:
  - Java 2 SDK, Standard Edition, for Windows  
Sun JDBC-ODBC Bridge driver is installed automatically when you install the SDK. You can download the SDK from the Sun Web site at <http://java.sun.com/j2se/>.
  - An application server that implements Sun JavaServer Pages 1.0 specification  
For more information, see “JSP application server” on page 241.
- 4 On the Windows server, create a subfolder in the c:\Inetpub\wwwroot folder and give it a name that describes your site.  
Here's an example:  
c:\Inetpub\wwwroot\MyPublishedSite
- 5 If you want to use the tutorial's database, copy the database file from the Macintosh to the server.  
The Microsoft Access database file, compasstravel.mdb, is located on your Macintosh hard disk in the Tutorial folder in the Dreamweaver UltraDev application folder.  
You can place the database file anywhere on the server's hard disk.
- 6 Set up a DSN pointing to the database on the server.  
A DSN is a kind of shortcut used to establish a database connection. For instructions, see “Setting Up a DSN in Windows” on page 249.  
If you're using the tutorial database, call your DSN “CompassTravel”.

## Configuring UltraDev on the Macintosh

Here is how to configure UltraDev on the Macintosh to work with the system configuration outlined in the previous section.

### To configure UltraDev on the Macintosh:

- 1 Create a new folder to hold working copies of your site files.

Here's an example:

Macintosh HD:Sites:*MyWorkingSite*

- 2 Launch UltraDev, choose Site > New Site, and complete the Local Info dialog box as follows:

Site Name: *MyFirstSite*

Local Root Folder: Macintosh HD:Sites:*MyWorkingSite*

- 3 Click Remote Info and complete the dialog box as follows:

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyPublishedSite/*

Login: *MyUserName*

Password: *MyPassword*

- 4 If you're interested in developing an ASP site, click Application Server and complete the dialog box as follows:

Server Model: ASP 2.0

Scripting Language: VBScript or JavaScript

Page Extension: .asp

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyPublishedSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyPublishedSite*

- 5 If you want to use ColdFusion Server as your application server, click Application Server and complete the dialog box as follows:

Server Model: ColdFusion 4.0

Scripting Language: CFML

Page Extension: .cfm

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyPublishedSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyPublishedSite*

- 6 If you want to use a JSP application server, click Application Server and complete the dialog box as follows:

Server Model: JSP 1.0

Scripting Language: Java

Page Extension: .jsp

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyPublishedSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyPublishedSite*

- 7 Click OK.

## Creating a database connection

A database connection is a set of parameters you define that allows your Web application to find and use a database. You defined the parameters when you set up your DSN. Next, use the DSN to create the database connection.

You create different database connections for an ASP, ColdFusion, and JSP site.

**To create a database connection for an ASP application:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
  - 2 Click the New button and select Data Source Name (DSN) from the pop-up menu.  
The Data Source Name (DSN) dialog box appears.
  - 3 Enter a name for the new connection.
  - 4 Enter a DSN.  
If you're using the tutorial database, enter CompassTravel. If you're using another database, enter the DSN you created.
  - 5 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see "Configuring UltraDev on the Macintosh" on page 18).
  - 6 Click OK.  
Your new connection should now appear in the Connections dialog box.
  - 7 Click Done to close the Connections dialog box.
- You can now start building your Web application. To learn more, see "Learning Dreamweaver UltraDev" on page 33.

**To create a database connection for a ColdFusion application:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select Data Source Name from the pop-up menu.  
UltraDev prompts you for your ColdFusion user name and password.
- 3 Enter the user name and password you use to log in to the ColdFusion Administrator.  
UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name dialog box.
- 4 Enter a name for the new connection.
- 5 Select a DSN.  
If you're using the tutorial database, select CompassTravel from the pop-up menu. If you're using another database, select the DSN you created for the database.

**6** Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see “Configuring UltraDev on the Macintosh” on page 18).

**7** Click OK.

Your new connection should now appear in the Connections dialog box.

**8** Click Done to close the Connections dialog box.

You can now start building your Web application. To learn more, see “Learning Dreamweaver UltraDev” on page 33.

**To create a database connection for a JSP application:**

**1** In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

**2** Click the New button and select “ODBC Database (Sun JDBC-ODBC Driver)” from the pop-up menu.

The ODBC Database (Sun JDBC-ODBC Driver) dialog box appears.

**3** Enter a name for the new connection.

**4** If you’re using the tutorial database, replace the [odbc dsn] placeholder in the URL box with CompassTravel.

The URL box should look like this:

```
jdbc:odbc:CompassTravel
```

**5** If you’re using another database, replace the [odbc dsn] placeholder in the URL box with the DSN you created for the database.

**6** Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see “Configuring UltraDev” on page 12).

**7** Click OK.

Your new connection should now appear in the Connections dialog box.

**8** Click Done to close the Connections dialog box.

You can now start building your Web application. To learn more, see “Learning Dreamweaver UltraDev” on page 33.

## Configuring your system

Before you can develop Web applications using UltraDev, you need to configure your system.

This section provides general procedures to help you configure your system. If you successfully configured your system using the procedures in one of the quick start sections (“Quick start for Windows users” on page 11 or “Quick start for Macintosh users” on page 16), you can skip this section.

Configuring your system consists of the following tasks:

- Configuring your Web server
- Defining a local site in UltraDev
- Defining a remote site in UltraDev
- Specifying a server technology in UltraDev
- Specifying an application server in UltraDev
- Specifying a URL prefix in UltraDev

### Configuring your Web server

To host your Web site, you can use any Web server that works with your chosen ASP, JSP, or ColdFusion application server.

If you installed UltraDev on a Windows 95, 98 or NT Workstation computer, you can install free Web server software from Microsoft called Personal Web Server (PWS) and run it on your local computer. For installation instructions, see “Installing Microsoft Personal Web Server” on page 243. If you installed UltraDev on a Windows 2000 computer, you can install the Microsoft enterprise-strength Web server, Internet Information Server (IIS) 5.0, included in the Windows 2000 package.

Once your Web server software is installed (either locally or remotely on a server), you will need the following:

- An application server to run your Web application
- A database to be used by your Web application
- A database driver to allow your Web application to communicate with your database

The exact requirements vary depending on whether you use UltraDev to create ASP, ColdFusion, or JSP applications. For specific requirements, see “Requirements for ASP developers” on page 233, “Requirements for ColdFusion developers” on page 237, or “Requirements for JSP developers” on page 240.

## Defining a local site

Dreamweaver UltraDev gives you the ability to manage your files and to transfer files between your local disk and your Web server at the click of a button. To benefit from these features, you must do the following:

- Create a folder on your local disk to store the files you'll create for your application. You may want to create subfolders to store image files and other assets.
- Define a local site. The local site is the folder you create on your local disk to store your files. If you don't define a local site, UltraDev will not work properly. This section describes how to define a local site.
- Define a remote site. The remote site is the folder on the Web server destined to hold your site files. For more information, see "Defining a remote site" on page 23.

You must define a local site for each new site you create. The local site is the folder you use to store the site files on your local disk. If you don't define a local site, Dreamweaver UltraDev will not work properly. Defining a local site is a one-time-only requirement.

### To define a local site:

- 1 Choose Site > New Site.
- 2 In the Site Definition dialog box, select Local Info from the Category list.
- 3 Enter a name in the Site Name box.
- 4 In the Local Root Folder box, specify the folder on your local disk where the application files will be stored by entering a path or clicking the folder icon to browse to and select the folder.
- 5 If you want, complete the other options in the Local Info category (they are not required to make the site work).

For more information on these options, see "Planning and Setting Up Your Site," in Dreamweaver Help (Help > Using Dreamweaver) or in the *Using Dreamweaver* guide.

- 6 Leave the Site Definition dialog box open for now.

You must define a remote site next.

## Defining a remote site

The remote site is your site's published root folder on the Web server. The folder is "published" in the sense that you can request pages from it using the HTTP protocol. (The URL in your browser starts with "http://...".)

**Note:** Another common term for Web server is HTTP server.

Some Web servers can run on your local computer. For example, you can run Microsoft Personal Web Server on your local Windows computer.

In most cases, the Web server will run on the system (including your local computer) that also runs your application server. For more information, see “Specifying an application server” on page 26.

**To define a remote site:**

- 1 If the Site Definition dialog box is not open, open it by choosing Site > Define Sites, selecting your site, and clicking Edit.

- 2 In the Category list on the left, click Remote Info.

The Remote Info dialog box appears.

- 3 In the Access pop-up menu, choose one of the following options: Local/Network or FTP.

For more information, see “Site Management and Collaboration,” in Dreamweaver Help (Help > Using Dreamweaver) or in the *Using Dreamweaver* guide.

You can also send your files to a SourceSafe application by choosing SourceSafe Database. (SourceSafe is used by developers for file version control.) If you choose this option, you need to define a separate application server. For instructions, see “Specifying an application server” on page 26.

- 4 If you chose Local/Network, click the folder icon and specify the root folder of your remote site.

The root folder is where documents published by your Web server are stored.

- 5 If you chose FTP, complete the other options in the dialog box:

- Enter the host name of the FTP host.
- Enter the name of the host directory. The host directory is where documents published to the Web are stored.
- Enter the login name and password used to connect to the FTP server.
- Select the other options as appropriate.

For more information on the Remote Info options, see “Site Management and Collaboration,” in Dreamweaver Help (Help > Using Dreamweaver) or in the *Using Dreamweaver* guide.

- 6 Leave the Site Definition dialog box open for now.

You must specify a server technology next.



## Specifying a server technology

When configuring UltraDev for Web application development, you must not only organize your site files, you must specify a server technology, an application server (see “Specifying an application server” on page 26), and a URL prefix (see “Specifying a URL prefix” on page 27). This section describes how to specify a server technology.

You must specify a server technology for each new site you create. Specifying a server technology tells UltraDev what kind of server-side scripts to insert into your pages. If you have a ColdFusion server, UltraDev will insert the necessary ColdFusion tags and scripts in the page. If you have a Web server that implements Sun JavaServer Pages specification, UltraDev will insert Java code. If you have a server that implements Microsoft Active Server Pages specification, you can have UltraDev insert either VBScripts or JavaScripts.

You specify a server technology for a site as a whole, not for individual pages. This ensures that all the pages in your application are compatible.

**Note:** You must define a site before you can specify a server technology. See “Defining a local site” on page 23.

### To specify a server technology:

- 1 If the Site Definition dialog box is not open, open it by choosing Site > Define Sites, selecting your site, and clicking Edit.
- 2 In the Site Definition dialog box, select Application Server from the Category list.

The Application Server dialog box appears.

- 3 Complete the following options:

- In the Server Model pop-up menu, choose ASP 2.0, JSP 1.0, or ColdFusion 4.0.
- If you chose ASP as your server model, set the Scripting Language to VBScript or JavaScript. (If you chose JSP or ColdFusion, this option is set for you.)
- For the Page Extension, accept the default file extension or choose another extension from the pop-up menu. The file extension will be added to every page you create for the site.

**Note:** The default .asp, .jsp, or .cfm extension will not affect your non-dynamic pages. However, changing the extension to .htm or .html will disable any dynamic content you create from then on. You will have to manually change the extension of the dynamic pages to .asp, .jsp, or .cfm, as appropriate.

- 4 Leave the Site Definition dialog box open for now.

You may have to specify an application server next.

## Specifying an application server

By default, UltraDev assumes the application server runs on the same system as your Web server. If you defined a remote site in the Remote Info category, and if the application server runs on the same system as the remote site (including your local computer), then accept the default settings in the Application Server category and skip to “Specifying a URL prefix” on page 27.

If you did not define a remote site in the Remote Info category, the Application Server category defaults to the local root folder you defined in the Local Info category. You can leave this setting alone only if you meet the following two conditions:

- Your Web server and application server both run on your local computer. For example, you’re a ColdFusion developer running Personal Web Server and ColdFusion Server on your local Windows 98 system.
- Your local root folder is a subfolder of your home directory. For example, if you’re using Personal Web Server or IIS, your local root folder must be a subfolder of the c:\Inetpub\wwwroot\ folder, or the folder itself.

If your local root folder is not a subfolder of your home directory, you must define the local root folder as a virtual directory in your Web server.

For more information on home and virtual directories, see “About the URL prefix” on page 28.

The application server can also run on a different system than your remote site. For example, if the Access option you selected in the Remote Info category is SourceSafe, then you must specify a different server in the Application Server category.

**To specify an application server different from the server defined in the Remote Info category:**

- 1 If the Site Definition dialog box is not open, open it by choosing Site > Define Sites, selecting your site, and clicking Edit.
- 2 If the Application Server category is not displayed, display it by clicking Application Server in the Category list on the left.
- 3 In the Access pop-up menu, specify how UltraDev should send dynamic pages to the application server, then specify where the dynamic pages should be sent.

The destination folder must be on a system with an application server capable of processing your dynamic pages.

- 4 Leave the Site Definition dialog box open for now.

You must specify a URL prefix next.

## Specifying a URL prefix

You must specify a URL prefix so UltraDev can use your application server at design time. Among other things, UltraDev uses the application server to generate the dynamic content displayed in the Live Data window and in your browser when you use the Preview in Browser command. For more information on the Live Data window, see “Working in the Live Data window” on page 82. For more information on the Preview in Browser command, see “Using Preview in Browser” on page 86.

UltraDev also uses the application server to establish connections to a database at design time. UltraDev uses the design-time connection to provide you with useful information about the database, such as the names of the tables in your database and the names of the columns in your tables.

To learn more about URL prefixes, see “About the URL prefix” on page 28.

### To specify the URL prefix:

- 1 If the Site Definition dialog box is not open, open it by choosing Site > Define Sites, selecting your site, and clicking Edit.
- 2 If the Application Server category is not displayed, display it by clicking Application Server in the Category list on the left.
- 3 In the URL Prefix box, enter the URL that users type in their browsers to open your Web application, leaving out any file name.

For example, suppose your application’s URL is as follows:

`http://www.macromedia.com/mycoolapp/start.jsp`

Enter the following URL prefix:

`http://www.macromedia.com/mycoolapp/`

If UltraDev runs on the same system as your Web server, you can use the term “localhost” as a stand-in for your domain name. For example, suppose your application’s URL is as follows:

`http://buttercup_pc/mycoolapp/start.jsp`

You could then enter the following URL prefix:

`http://localhost/mycoolapp/`

For more information, see “About the URL prefix” on page 28.

- 4 Click OK, then click Done.

## About the URL prefix

A URL prefix is made up of the domain name and any of your home directory's subdirectories or virtual directories.

This section uses the terminology used in Microsoft Personal Web Server (PWS) and Internet Information Server (IIS). Though the terminology may vary from server to server, the same concepts apply to most Web servers.

**The home directory** is the folder on the server mapped to your site's domain name. Suppose the folder you want to use to process pages for the Live Data window is `c:\sites\company\`, and this folder is your home directory (that is, this folder is mapped to your site's domain name—for example, `www.mystartup.com`). In that case, the URL prefix is as follows:

`http://www.mystartup.com/`

If the folder you want to use to process your Live Data pages is a subfolder of your home directory, simply add the subfolder to the URL. For example, suppose your home directory is `c:\sites\company\`, your site's domain name is `www.mystartup.com`, and the folder you want to use to process Live Data pages is `c:\sites\company\inventory`. Here's the URL prefix:

`http://www.mystartup.com/inventory/`

If the folder you want to use to process Live Data pages is not your home directory or any of its subdirectories, you must create a virtual directory.

**A virtual directory** is a folder not physically contained in the home directory of the server even though it appears to be in the URL. To create a virtual directory, you specify an alias to stand in for the folder's path in the URL. For example, suppose your home directory is `c:\sites\company`, your processing folder is `d:\apps\inventory`, and you define an alias for this folder called "warehouse." Here's the URL prefix:

`http://www.mystartup.com/warehouse/`

**Localhost** is a term you can use to refer to the home directory in your URLs when the client (usually a browser, but in this case UltraDev) runs on the same system as your Web server. For example, suppose UltraDev is running on the same system as the Web server, your home directory is `c:\sites\company`, and you defined a virtual directory called "warehouse" to refer to the folder you want to use to process Live Data pages. Here's the URL prefix:

`http://localhost/warehouse/`

To determine your domain name and home directory in PWS and IIS 5.0, click the Main icon in Personal Web Manager and note the home page specified in the Publishing area.

## What's new in UltraDev 4

The new features in Dreamweaver UltraDev 4 simplify the tasks of building dynamic pages, hand-coding server scripts, writing custom server behaviors, and creating database connections. UltraDev 4 also incorporates all the new features of Dreamweaver 4 (see “What's New in Dreamweaver 4” in the introduction to the *Using Dreamweaver* guide or Dreamweaver Help).

Here are the major new features in UltraDev 4.

### Live objects

In UltraDev 4, you can use live objects to speed up development. Live objects let you create advanced page components in one operation. You can use live objects to create a master/detail page set, a record insert form, or a record update form. You can also use live objects to create recordset navigation bars and record counters.

### Site access server behaviors

UltraDev 4 gives you the tools to build pages that restrict access to your site. You can build pages that let first-time users register on your site and pages that require returning users to log in.

You can also give users different access privileges to your site. For example, if you set the authorization level for a page to Member, only registered users with Member access privileges will be able to view it.

### Improved hand-coding

New Code view and syntax coloring make it easier than ever to hand-code ASP or JSP server scripts or CFML tags.

At design time, UltraDev also recognizes application files such as `global.asa` and `application.cfm`, as well as server-side includes. These changes make the Live Data window even more powerful than before.

### Simplified extensibility model

Simplified architecture makes it easier to create extensions. UltraDev also comes with a new Server Behavior Builder—a tool for creating server behaviors and for customizing the code inserted in pages by existing server behaviors.

## Remote database connectivity

UltraDev 4 introduces remote database connectivity, a feature that lets you connect to your database at design time with little or no effort. You no longer need to define a design-time connection or, if you're a Macintosh user or JSP developer, install a JDBC driver on your local system. UltraDev uses your run-time connection at design time.

## UltraDev resources

The Dreamweaver UltraDev package contains a variety of resources to help you learn the program quickly and become proficient in creating Web sites, Web pages, and Web applications. These resources include online help pages, Guided Tour movies, lessons, tutorials, and two printed user guides. In addition, the Dreamweaver UltraDev Support Center (Help > UltraDev Support Center) is updated regularly.

Dreamweaver UltraDev has all the functionality of Dreamweaver 4 for building Web pages and managing Web sites. To learn more, see *Using Dreamweaver* or Dreamweaver Help (Help > Using Dreamweaver). To learn how to build Web applications, consult this guide or use UltraDev Help (Help > Using UltraDev).

The following sections describe the instructional resources at your disposal.

### Guided Tour movies

The Guided Tour movies show you how dynamic pages are created in UltraDev.

To start the UltraDev tour, choose Help > Guided Tour. A list of guided tour movies appears. Click the first movie to start. When the movie finishes, click the Home button to return to the list of movies, then click another movie.

Guided Tour movies require the Flash plugin, which is included on the Dreamweaver UltraDev CD. If you purchased your copy of UltraDev electronically, you can download the latest Flash plugin from the Macromedia Web site at <http://www.macromedia.com/software/flashplayer/downloads/>.

### Help systems

The Dreamweaver UltraDev package comes with two help systems: Dreamweaver Help and UltraDev Help. These HTML-based help systems provide comprehensive information about all Dreamweaver and UltraDev features.

To view the help systems, use Netscape Navigator 4.0 and later or Microsoft Internet Explorer 4.0 and later. Because the help systems make extensive use of JavaScript, make sure JavaScript is enabled in your browser. If you plan to use the search feature, make sure Java is enabled as well.

Each help system includes the following components:

**The table of contents** lets you see all the information organized by subject. Click top-level entries to view subtopics.

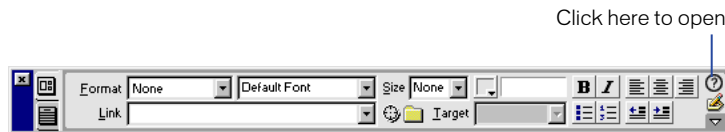
**The index**, like a traditional printed index, helps you find important terms and go to related topics.

**Search** allows you to find any character string in all topic text. To search for a phrase, simply type the phrase in the text field. To search for files that contain two keywords (for example, *layers* and *styles*), separate the search terms with a Plus (+) sign.

The search feature requires a 4.0 browser with Java enabled.

**Note:** After clicking Search, a Java security window may appear, asking for permission to read files on your hard disk. You must grant this permission for the search to work. The applet does not write anything to your hard disk, nor does it read any files outside the HTML help pages.

**Context-sensitive help** provides a Help button in each dialog box, or a question mark icon in inspectors, windows, and panels, that opens a relevant help topic.



**The navigation bar** provides buttons you can click to move through topics. The Previous and Next buttons move to the previous or next topic in a section (following the topic order listed in the table of contents).

## Tutorials

The Dreamweaver UltraDev package comes with one tutorial to get you started with Dreamweaver, and a different tutorial to introduce you to building Web applications with UltraDev.

The Dreamweaver tutorial is the best place to start if you don't have much experience with Dreamweaver. By working through the tutorial, you'll learn how to edit a sample Web site with some of the most useful and powerful features in Dreamweaver. The tutorial is in Dreamweaver Help (Help > Using Dreamweaver) and in the *Using Dreamweaver* book.

If you're already familiar with Dreamweaver, start with the UltraDev tutorial. The UltraDev tutorial teaches you how to build a small Web application that draws content from a database. The UltraDev tutorial is in UltraDev Help (Help > Using UltraDev) and in the *Using Dreamweaver UltraDev* book.

## Lessons

Dreamweaver UltraDev comes with a series of interactive lessons for Dreamweaver, and another series of lessons for UltraDev. Each lesson shows you how to use a different feature of the product even as you use that feature in your projects.

To take a lesson, choose Help > Lessons. A list of lessons appears. Click the lesson that interests you.

The lessons require the Flash plugin, which is included on the Dreamweaver UltraDev CD. If you purchased your copy of Dreamweaver electronically, you can download the latest Flash plugin from the Macromedia Web site at <http://www.macromedia.com/software/flashplayer/downloads/>.

## User guides (printed books)

The *Using Dreamweaver* and *Using Dreamweaver UltraDev* books included in the boxed version of UltraDev provide a printed alternative to the two help systems. Certain reference topics about program options are not included in the printed books; the books direct you to the help pages for information on those topics.

## Extensibility documentation

The *Extending Dreamweaver & UltraDev* book and help pages provide information on the Dreamweaver Document Object Model and the APIs (application programming interfaces) that allow JavaScript and C developers to create objects, commands, property inspectors, behaviors, and translators.

## Support centers

To help you get the most out of Dreamweaver UltraDev, you can consult two Web-based support centers:

- For information on the site-creation and page-design side of Dreamweaver UltraDev, visit the Dreamweaver Support Center at <http://www.macromedia.com/support/dreamweaver/>.
- For information on the application-building side of the product, visit the UltraDev Support Center at <http://www.macromedia.com/support/ultradev/>.

Both support centers are updated regularly with the latest information, plus advice from expert users, information on advanced topics, examples, and tips.



## UltraDev discussion group

Discuss technical issues and share helpful hints with other UltraDev users by visiting the UltraDev discussion group. You'll find information on accessing the discussion group on the Macromedia Web site at <http://www.macromedia.com/software/ultradev/discussiongroup/>.

## Learning Dreamweaver UltraDev

The Dreamweaver UltraDev package includes information for readers of all expertise levels. To get the most out of the documentation, start by reading the parts that are most relevant to your level of experience.

### **For users new to Dreamweaver:**

See “Where to start” in the introduction to *Using Dreamweaver*.

### **For experienced Web designers new to Web application development:**

- 1 Take the UltraDev Guided Tour (Help > Guided Tour).
- 2 Work through “Dreamweaver UltraDev Tutorial,” to learn the basics of using UltraDev.
- 3 Read “UltraDev Basics” on page 75, to make sure you understand basic concepts and terms.
- 4 Skim the other chapters in the *Using Dreamweaver UltraDev* guide.

### **For experienced Web application developers:**

- 1 Take the UltraDev Guided Tour (Help > Guided Tour of UltraDev).
- 2 Work through “Dreamweaver UltraDev Tutorial,” to learn the basics of using UltraDev.
- 3 Skim the other chapters of the *Using Dreamweaver UltraDev* guide.

## Web application resources

The following are some useful resources available on the Web:

**The HTML 4.01 specification** (<http://www.w3.org/TR/REC-html40/>) is the official specification for HTML from the World Wide Web Consortium.

**Microsoft ASP Overview pages** (<http://msdn.microsoft.com/workshop/server/asp/ASPOver.asp>) provide information about Active Server Pages (ASP).

**Sun JSP page** (<http://java.sun.com/products/jsp/>) provides information about JavaServer Pages (JSP).

**Allaire ColdFusion product page** (<http://www.allaire.com/Products/ColdFusion/productinformation/>) provides information about ColdFusion.

**The XML.com site** (<http://www.xml.com>) provides information about XML.

**IBM WebSphere page** (<http://www.ibm.com/software/webservers/appserv/>) provides information about IBM WebSphere application server.

**Chili!Soft product page** (<http://www.Chilisoft.com/products/>) provides information about Chili!Soft ASP.

## Accessibility and Dreamweaver

Macromedia supports the creation of great Web experiences that are accessible to everyone, including those with disabilities. We encourage the implementation of international standards to guide developers of accessible sites, including the guidelines offered by the World Wide Web Consortium (W3C). Many government policies on Web accessibility, including those of the United States, reference W3C guidelines. W3C guidelines on Web content encourage developers to adopt design and coding practices for accessibility, many of which are robustly supported by Macromedia products. For more information on W3C guidelines, please consult the Web Content Authoring Guidelines (<http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html>).

For the latest information on product features and resources that support accessible design, please see Macromedia's accessibility page (<http://www.macromedia.com/accessibility/>).

# CHAPTER 1

## Dreamweaver UltraDev Tutorial

---

This tutorial shows you how to build a simple Web application using Dreamweaver UltraDev. You'll create a Web-based employee directory for a fictitious adventure travel company called Compass. The directory will give Compass employees the ability to search and get more information about their coworkers.

Along the way, you'll learn how to build the following pages:

- Two sets of search/results pages—a simple one requiring no knowledge of SQL (Structured Query Language, a common database language), and a more advanced set requiring some knowledge of SQL
- A detail page to display the information about particular employees, including their photographs
- An insert page allowing the site administrator to insert new employee records in the database with a Web browser

This tutorial also covers the following one-time-only tasks:

- Defining a local site so you can develop the Web application on your local system
- Defining a remote site so you can deploy the application on a Web server
- Configuring UltraDev to work with your chosen application server
- Creating a database connection so you can interact with the tutorial database

To complete this tutorial, you need a Web server and the following software installed and configured on the server:

- An application server that supports your chosen server technology (ASP, JSP or ColdFusion)

**Note:** Microsoft Personal Web Server and Internet Information Server (IIS) double as ASP application servers. You don't need extra software to run ASP applications on these servers.

- A database driver that supports Microsoft Access 97 databases.

The easiest way to meet these requirements is to obtain a trial account with a Macromedia recommended Internet service provider (ISP). For more information, see the Macromedia Web site at <http://www.macromedia.com/software/ultradev/isp/>.

If you want to configure your own system, see “Tutorial quick start for Windows users” on page 36 or “Tutorial quick start for Macintosh users” on page 41.

If you don't want to use the system configurations described in the quick start sections, complete the procedures in the following sections:

- “Define a local site” on page 47
- “Define a remote site” on page 48
- “Configure UltraDev to work with your application server” on page 49
- “Create a database connection” on page 50

## Tutorial quick start for Windows users

If you choose not to sign up for the trial account, this section describes a quick way for Windows users to start working on the tutorial. The section guides you through the following steps:

- Configuring your computer
- Configuring UltraDev
- Creating a database connection

## Configuring your computer

Here is the simplest system configuration for Windows users.

### To configure your computer:

- 1 If you're a Windows 95, 98, or NT Workstation user, install Microsoft Personal Web Server (PWS).

For detailed instructions, see “Installing Microsoft Personal Web Server” on page 243.

- 2 If you're a Windows NT Server or Windows 2000 user, make sure Internet Information Server (IIS) is installed and running on your system.

IIS is the full-featured version of PWS. It should already be installed on your system. If not, install it or ask your system administrator to install it for you.

- 3 If you want to do the ColdFusion tutorial, install ColdFusion Server on your system.

For instructions, see “Installing Allaire ColdFusion Server” on page 245.

- 4 If you want to do the JSP tutorial, install the following components on your system:

- Java 2 SDK, Standard Edition, for Windows

Sun JDBC-ODBC Bridge driver is installed automatically when you install the SDK. You can download the SDK from the Sun Web site at <http://java.sun.com/j2se/>.

- An application server that implements Sun JavaServer Pages 1.0 specification

For more information, see “JSP application server” on page 241.

- 5 Create a subfolder in the c:\Inetpub\wwwroot folder and call it **MyTutorialSite**.

Here's the correct folder structure:

c:\Inetpub\wwwroot\MyTutorialSite

## Configuring UltraDev

Here is how to configure UltraDev to work with the system configuration outlined in the previous section

### To configure UltraDev:

- 1 Launch UltraDev and choose Site > Define Sites.

The Define Sites dialog box appears.

- 2 Select the predefined ASP, ColdFusion, or JSP tutorial site and click Edit.

UltraDev defined the tutorial sites during installation.

- 3 Click Remote Info and complete the dialog box as follows:

Access: Local/Network

Remote Folder: c:\Inetpub\wwwroot\MyTutorialSite

- 4 If you want to do the ASP tutorial, click Application Server and complete the dialog box as follows:

Server Model: ASP 2.0

Scripting Language: VBScript or JavaScript

Page Extension: .asp

Access: Local/Network

Remote Folder: c:\Inetpub\wwwroot\MyTutorialSite

URL Prefix: http://localhost/MyTutorialSite

- 5 If you want to do the ColdFusion tutorial, click Application Server and complete the dialog box as follows:

Server Model: ColdFusion 4.0

Scripting Language: CFML

Page Extension: .cfm

Access: Local/Network

Remote Folder: c:\Inetpub\wwwroot\MyTutorialSite

URL Prefix: http://localhost/MyTutorialSite

- 6 If you want to do the JSP tutorial, click Application Server and complete the dialog box as follows:

Server Model: JSP 1.0

Scripting Language: Java

Page Extension: .jsp

Access: Local/Network

Remote Folder: c:\inetpub\wwwroot\MyTutorialSite

URL Prefix: http://localhost/MyTutorialSite

- 7 Click OK.
- 8 Click Done.
- 9 Select all the files under Local Folder, including the images folder, and click the blue up-arrow on the toolbar to upload them to the remote site.

## Creating a database connection

A database connection allows your Web application to find and use a database. You create different database connections for the ASP, ColdFusion, and JSP tutorial.

### To create a database connection for the ASP tutorial:

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select Data Source Name (DSN) from the pop-up menu.  
The Data Source Name (DSN) dialog box appears.
- 3 Enter the following name for the new connection: **connCompass**.  
A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.
- 4 Select CompassTravel from the list of DSNs.  
UltraDev created the CompassTravel DSN when it installed.
- 5 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double check the DSN. If the connection still fails, check your URL prefix for the application server.
- 6 Click OK.  
Your new connection should now appear in the Connections dialog box.

**7** Click Done to close the Connections dialog box.

You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.

**To create a database connection for the ColdFusion tutorial:**

**1** In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

**2** Click the New button and select Data Source Name from the pop-up menu.

UltraDev prompts you for your ColdFusion user name and password.

**3** Enter the user name and password you use to log in to the ColdFusion Administrator.

UltraDev retrieves the ColdFusion DSNs and displays the Data Source Name dialog box.

**4** Enter the following name for the new connection: **connCompass**.

A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.

**5** Select CompassTravel from the list of DSNs.

UltraDev created the CompassTravel DSN when it installed.

**6** Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server.

**7** Click OK.

Your new connection should now appear in the Connections dialog box.

**8** Click Done to close the Connections dialog box.

You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.



**To create a database connection for the JSP tutorial:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select “ODBC Database (Sun JDBC-ODBC Driver)” from the pop-up menu.  
The ODBC Database (Sun JDBC-ODBC Driver) dialog box appears.
- 3 Enter a name for the new connection.
- 4 Replace the [odbc dsn] placeholder in the URL box with **CompassTravel**.

The URL box should look like this:

```
jdbc:odbc:CompassTravel
```

- 5 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see “Specify a URL prefix” on page 50).

- 6 Click OK.

Your new connection should now appear in the Connections dialog box.

- 7 Click Done to close the Connections dialog box.

You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.

## Tutorial quick start for Macintosh users

If you choose not to sign up for the trial account, this section describes a quick way for Macintosh users to start working on the tutorial. The section guides you through the following steps:

- Configuring your server
- Configuring UltraDev on the Macintosh
- Creating a database connection

Because popular Web servers and application servers do not yet support the Macintosh, you need another computer to run the server software. This section assumes you have access to a Windows NT Server or Windows 2000 computer running Internet Information Server (IIS), a common commercial Web server.

## Configuring your server

This section assumes you have access to a Windows NT Server or Windows 2000 computer.

### To configure the server:

- 1 If not already done, install Internet Information Server (IIS) on the server.

IIS should already be installed on the system. If not, install it or ask your system administrator to install it for you.

- 2 If you want to do the ColdFusion tutorial, install ColdFusion Server on the server.

For instructions, see “Installing Allaire ColdFusion Server” on page 245.

- 3 If you want to do the JSP tutorial, install the following components on the Windows server:

- Java 2 SDK, Standard Edition, for Windows

Sun JDBC-ODBC Bridge driver is installed automatically when you install the SDK. You can download the SDK from the Sun Web site at <http://java.sun.com/j2se/>.

- An application server that implements Sun JavaServer Pages 1.0 specification

For more information, see “JSP application server” on page 241.

- 4 On the Windows server, create a subfolder in the c:\Inetpub\wwwroot folder and call it **MyTutorialSite**.

Here’s the correct folder structure:

c:\Inetpub\wwwroot\MyTutorialSite

- 5 Copy the tutorial database file to the server.

The Microsoft Access database file, `compasstravel.mdb`, is located on your Macintosh hard disk in the Tutorial folder in the Dreamweaver UltraDev application folder.

- 6 On the server, set up a DSN called “CompassTravel” pointing to the tutorial database you copied to the server.

For instructions, see “Setting Up a DSN in Windows” on page 249.

## Configuring UltraDev on the Macintosh

Here is how to configure UltraDev on the Macintosh to work with the system configuration outlined in the previous section.

### To configure UltraDev on the Macintosh:

- 1 Launch UltraDev and choose Site > Define Sites.

The Define Sites dialog box appears.

- 2 Select the predefined ASP, ColdFusion, or JSP tutorial site and click Edit.

UltraDev defined the tutorial sites during installation.

- 3 Click Remote Info and complete the dialog box as follows:

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyTutorialSite/*

Login: *MyUserName*

Password: *MyPassword*

- 4 If you want to do the ASP tutorial, click Application Server and complete the dialog box as follows:

Server Model: ASP 2.0

Scripting Language: VBScript or JavaScript

Page Extension: .asp

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyTutorialSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyTutorialSite*

- 5 If you want to do the ColdFusion tutorial, click Application Server and complete the dialog box as follows:

Server Model: ColdFusion 4.0

Scripting Language: CFML

Page Extension: .cfm

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyTutorialSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyTutorialSite*

- 6 If you want to do the JSP tutorial, click Application Server and complete the dialog box as follows:

Server Model: JSP 1.0

Scripting Language: Java

Page Extension: .jsp

Access: FTP

FTP Host: *MyFTPHost*

Host Directory: *MyTutorialSite/*

Login: *MyUserName*

Password: *MyPassword*

URL Prefix: *http://MyDomainName/MyTutorialSite*

- 7 Click OK.

- 8 Click Done.

- 9 Select all the files under Local Folder, including the images folder, and click the blue up-arrow on the toolbar to upload them to the remote site.

## Creating a database connection

A database connection allows your Web application to find and use a database. You create different database connections for the ASP, ColdFusion, and JSP tutorial.

**To create a database connection for the ASP tutorial:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select Data Source Name (DSN) from the pop-up menu.  
The Data Source Name (DSN) dialog box appears.
- 3 Enter the following name for the new connection: **connCompass**.  
A common practice is to add the prefix *conn* to connection names to distinguish them from other objects in your code.
- 4 In the Data Source Name (DSN) box, enter **CompassTravel**.  
This is the DSN you defined on the server.
- 5 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server.
- 6 Click OK.  
Your new connection should now appear in the Connections dialog box.
- 7 Click Done to close the Connections dialog box.

You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.

**To create a database connection for the ColdFusion tutorial:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
  - 2 Click the New button and select Data Source Name from the pop-up menu.  
UltraDev prompts you for your ColdFusion user name and password.
  - 3 Enter the user name and password you use to log in to the ColdFusion Administrator.  
UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name dialog box.
  - 4 Enter the following name for the new connection: **connCompass**.  
A common practice is to add the prefix *conn* to connection names to distinguish them from other objects in your code.
  - 5 Select CompassTravel from the list of DSNs  
This is the DSN you defined on the server.
  - 6 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server.
  - 7 Click OK.  
Your new connection should now appear in the Connections dialog box.
  - 8 Click Done to close the Connections dialog box.
- You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.

**To create a database connection for the JSP tutorial:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click the New button and select “ODBC Database (Sun JDBC-ODBC Driver)” from the pop-up menu.  
The ODBC Database (Sun JDBC-ODBC Driver) dialog box appears.
- 3 Enter a name for the new connection.
- 4 Replace the [odbc dsn] placeholder in the URL box with **CompassTravel**.  
The URL box should look like this:  
`jdbc:odbc:CompassTravel`

**5** Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check your URL prefix for the application server (see “Specify a URL prefix” on page 50).

**6** Click OK.

Your new connection should now appear in the Connections dialog box.

**7** Click Done to close the Connections dialog box.

You can now start the UltraDev tutorial. Skip to “Create a simple search/results page set” on page 56.

## Define a local site

A local site tells UltraDev where all the documents and files of a particular Web site are stored on your local disk. You must define a local site to develop a Web application using UltraDev.

When you installed UltraDev, it automatically created three local sites for you: one for the ASP tutorial, one for the JSP tutorial, and one for the ColdFusion tutorial. Choose the local site that’s appropriate for your server technology (ASP, JSP, or ColdFusion). For more information on these technologies, see “About dynamic pages” on page 76.

To choose a local site, launch UltraDev, open the Site window (Site > Site Files), and select the site from the pop-up menu on the toolbar. For example, if you have an ASP server, select the ASP tutorial site.



If you need to restart the tutorial from scratch, clean copies of the tutorial files are available in the Tutorial folder in the Dreamweaver UltraDev application folder. Open the subfolder appropriate to your server technology (ASP, JSP, or ColdFusion) and copy the clean copies from the Compass Intranet Backup folder.

**Note:** The complete path to the Tutorial folder varies, depending on where you installed Dreamweaver UltraDev.

## Define a remote site

You can use UltraDev to deploy your application on a Web server by defining a remote site. A remote site tells UltraDev where all the documents and files of your Web site are stored on your Web server.

To define a remote site, you must complete the following tasks:

- Set up a published folder on your Web server
- Define the remote site folder in UltraDev

### Set up a published folder on your Web server

Make sure your Web server supports either ASP, JSP, or ColdFusion, then set up a published folder for the tutorial on the server. For setup instructions, see the server documentation, or consult your system administrator.

If you use the Microsoft Personal Web Server or Internet Information Server (IIS), the easiest way to set up a published folder is to add a subfolder to the `c:\inetpub\wwwroot\` folder. For example, to set up the tutorial folder, you would create the following subfolder:

```
c:\inetpub\wwwroot\MyTutorial
```

Next, you'll define the MyTutorial folder as the remote site folder of your tutorial application.

### Define the remote site folder in UltraDev

In UltraDev, you define the tutorial's remote site folder by specifying the published folder you set up on the Web server, then uploading all the tutorial files to it.

- 1 Choose Site > Define Sites.

A dialog box appears listing currently defined sites.

- 2 Select your tutorial site from the list and click Edit.

- 3 In the Category list on the left, click Remote Info.

- 4 Choose one of the following Access options: Local/Network or FTP.

- 5 If you chose Local/Network, click the folder icon and select the folder you set up as a published folder on your Web server. Example:

```
c:\inetpub\wwwroot\MyTutorial
```



- 6 If you chose FTP, enter the host name of the FTP host, and enter the name of the host directory on the remote site. The host directory is where documents visible to the public are stored. Next, enter the login name and password used to connect to the FTP server, then select the appropriate firewall options.
- 7 Click OK and click Done.
- 8 In the Site window (Site > Site Files), verify that you've specified the correct folder on the remote site. If you're using FTP, click the Connect button to view the remote site.
- 9 Select all the files under Local Folder, including the Images folder, and click the blue up-arrow on the toolbar to upload them to the remote site.

## Configure UltraDev to work with your application server

You must specify the kind of application server you're using (ASP, JSP, or ColdFusion) so UltraDev knows what server-side scripts to insert in your dynamic pages. You must also specify a URL prefix so UltraDev can borrow the services of the application server at design time. Among other things, UltraDev uses the application server at design time to power the Live Data window and to establish connections to databases.

The Live Data window is a fully functional, visual design and editing environment that displays your page's dynamic content at design time.

### Specify a server technology

For this tutorial, you don't need to specify a server technology. During the installation, UltraDev automatically specified it for you.

Server technologies such as ASP, JSP, and ColdFusion give a Web server the ability to modify Web pages at run time. This is where the term *dynamic pages* comes from: a dynamic page is essentially one that changes at run time.

Specifying a server technology tells UltraDev what server-side scripts to insert in your pages. A server-side script is a set of instructions the server executes at run time. In UltraDev, these scripts are called server behaviors.

## Specify a URL prefix

You must specify a URL prefix so UltraDev can borrow the services of your application server at design time.

- 1 Choose Site > Define Sites.

A dialog box appears listing currently defined sites.

- 2 Select your tutorial site and click Edit.

- 3 In the Category list on the left, click Application Server.

- 4 Verify the URL prefix in the URL Prefix box.

UltraDev tries to determine your URL prefix based on the settings for your remote site. If the UltraDev guess is wrong, correct the URL prefix.

The URL prefix is the URL users type into their browsers to open your Web application, minus the file name at the end. For example, if your tutorial's URL is `http://www.macromedia.com/MyTutorial/Search.htm`, enter the following URL prefix:

`http://www.macromedia.com/MyTutorial/`

If UltraDev runs on the same system as your Web server, you can use the term “localhost” as a stand-in for your domain name. For example, if your application's local URL is `http://buttercup_pc/MyTutorial/Search.htm`, you can enter the following URL prefix:

`http://localhost/MyTutorial/`

For more information, see “About the URL prefix” on page 28.

- 5 Click OK, then click Done.

## Create a database connection

A database connection is a set of parameters you define to establish a link to a database. Without it, your application won't know where to find the database or how to talk to it. In this part of the tutorial, you'll create a connection to the tutorial's database file, `compasstravel.mdb`.

### Driver requirements

The tutorial's database is a Microsoft Access 97 file. You don't need Microsoft Access to run the tutorial, but you do need a driver capable of reading Microsoft Access 97 files. The driver allows your Web application to read the contents of the database file. (To learn more about database drivers, see “Interfacing with the database” on page 227.)

**If you are an ASP or ColdFusion developer**, an ODBC driver capable of reading Access 97 files must be installed on your ASP or ColdFusion server.

If your Web server runs on a remote or local Windows system, the required driver is probably installed. To find out if it is, in Windows choose Start > Settings > Control Panel, then look for the ODBC Data Sources icon. (Depending on your system, the icon could also be called ODBC or 32bit ODBC.) If the icon is not there, download and install the Microsoft Data Access Components (MDAC) 2.5, which will install the missing Access driver.

If the ODBC Data Sources icon is present, double-click it. The ODBC Data Sources Administrator dialog box appears. Click the Drivers tab for a list of drivers installed on your system. Look for the Microsoft Access Driver, version 3.5 or later.

If the Microsoft Access Driver (\*.mdb) doesn't appear in the list, download and install the MDAC, which will install the missing Access driver.

You can download the MDAC from the Microsoft Web site at <http://www.microsoft.com/data/download.htm>.

**If you're a JSP developer**, a JDBC driver capable of reading Access 97 files must be installed on your JSP server. For more information on JDBC drivers and their vendors, see the searchable database of JDBC drivers on the Sun Web site at <http://industry.java.sun.com/products/jdbc/drivers>.

## Create a database connection: ASP users

Before you can start building the Compass employee directory, you must create a connection to the tutorial's database file, `compasstravel.mdb`.

If you installed UltraDev on a Windows system, a DSN to the tutorial database was set up on your system during installation. You'll use this DSN if you also run your Web server on the same system—that is, if you use Personal Web Server in Windows, or if you installed UltraDev on Windows NT Server or Windows 2000.

If your Web server runs on a remote Windows system, you can define a data source name (DSN) on the remote server, then use that DSN in UltraDev to create an easy database connection. A DSN is a kind of shortcut to a database. For instructions on defining one, see "Setting Up a DSN in Windows" on page 249.

If a DSN has not been set up on the system running your Web server, then you must use a connection string to create the connection.

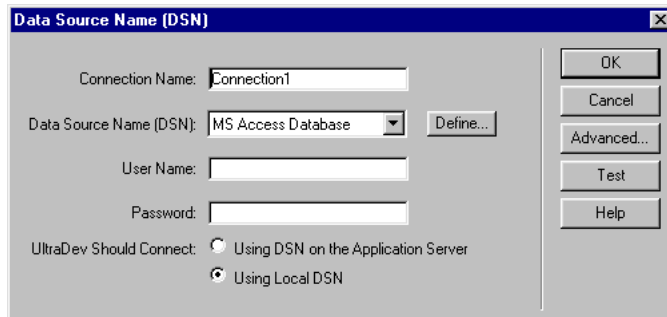
To create a database connection if a DSN has been set up on your Web server:

- 1 In the UltraDev Document window, choose Modify > Connections.

The Connections dialog box appears.

- 2 Click New and choose Data Source Name (DSN) from the pop-up menu.

The Data Source Name (DSN) dialog box appears.



- 3 In the Connection Name box, enter **connCompass**.

A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.

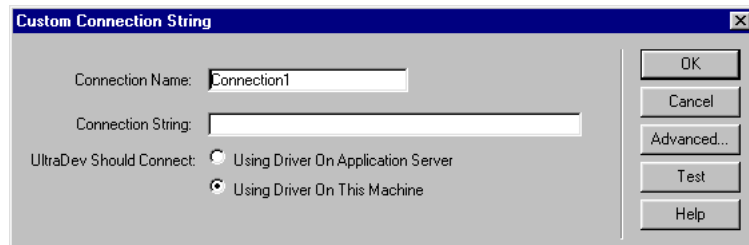
- 4 If your application server runs on your local computer, select CompassTravel from the list of DSNs.
- 5 If your application server runs on a remote computer, enter CompassTravel in the Data Source Name (DSN) box.
- 6 Click OK to finish defining the connection.

Your new connection, connCompass, appears in the Connections dialog box.

- 7 Click Done to close the Connections dialog box.

To create a database connection if a DSN has not been set up on your Web server:

- 1 In the UltraDev Document window, choose **Modify > Connections**.  
The Connections dialog box appears.
- 2 Click **New** and choose **Custom Connection String** from the pop-up menu.  
The Custom Connection String dialog box appears.



- 3 In the Connection Name box, enter **connCompass**.  
A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.
- 4 In the Connection String box, enter a connection string to the Compass database file, *compasstravel.mdb*, located on the Web server.  
If you're unfamiliar with connection strings, see "Writing a connection string" on page 97. Also see your server documentation or consult your system administrator.
- 5 Specify how UltraDev should connect to the database at design time:
  - If the driver specified in the connection string is located on a remote system, select the **Using Driver on Application Server** option.
  - If the driver specified in the connection string is located on the local system (that is, your Web server runs on the same Windows system that runs UltraDev), select the **Using Driver on This Machine** option.
- 6 Click **OK** to close the Custom Connection String dialog box.  
Your new connection, *connCompass*, appears in the Connections dialog box.
- 7 Click **Done** to close the Connections dialog box.

## Create a database connection: ColdFusion users

Before you can start building the Compass employee directory, you must create a connection to the tutorial's database file, `compasstravel.mdb`.

- 1 If ColdFusion Server does not run on the same system that runs your copy of UltraDev, use the ColdFusion Administrator to define a data source name (DSN) on the remote system.

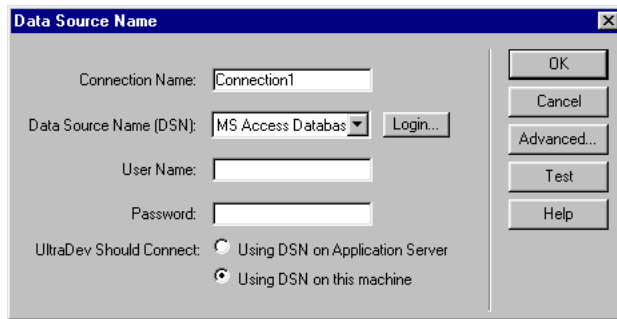
For instructions, see the ColdFusion documentation or consult your system administrator.

- 2 In the UltraDev Document window, choose **Modify > Connections**.

The Connections dialog box appears.

- 3 Click **New** and choose **Data Source Name** from the pop-up menu.

UltraDev prompts you for your ColdFusion user name and password. Enter the same user name and password you use to log in to the ColdFusion Administrator. After you enter them, UltraDev connects, retrieves the ColdFusion DSNs, and displays the Data Source Name dialog box.



- 4 In the Connection Name box, enter **connCompass**.

A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.

- 5 Select **CompassTravel** from the list of DSNs.
- 6 Click **OK** to finish defining the connection.

Your new connection, **connCompass**, appears in the Connections dialog box.

- 7 Click **Done** to close the Connections dialog box.

## Create a database connection: JSP users

Before you can start building the Compass employee directory, you must create a JDBC connection to the tutorial's database file, `compasstravel.mdb`.

- 1 In the UltraDev Document window, choose **Modify > Connections**.

The Connections dialog box appears.

- 2 Click **New** and choose your driver from the pop-up menu. If your driver does not appear, choose **Custom JDBC Connection**.

A connection dialog box appears.

- 3 In the Connection Name box, enter **connCompass**.

A common practice is to add the prefix *conn* to connection names to distinguish them from other object names in your code.

- 4 Enter the JDBC driver's connection parameters.

For the parameters specific to your driver, see your driver documentation or consult your system administrator. For general information, see "About JDBC connection parameters" on page 102.

For example, suppose the `compasstravel.mdb` database file resides on a JSP-enabled Windows NT Server system and you set up a DSN called `CompassTravel` on the server. If you're using Sun JDBC-ODBC driver, you would enter the following connection parameters:

Driver: `sun.jdbc.odbc.JdbcOdbcDriver`

URL: `jdbc:odbc:CompassTravel`

User Name:

Password:

- 5 Specify how UltraDev should connect to the database at design time:
  - If the driver specified in the Driver box is located on a remote system, select the **Using Driver on Application Server** option.
  - If the driver specified in the Driver box is located on the local system (that is, your Web server runs on the same Windows system that runs UltraDev), select the **Using Driver on This Machine** option.
- 6 Click **OK** to close the connection dialog box.

Your new connection, `connCompass`, appears in the Connections dialog box.
- 7 Click **Done** to close the Connections dialog box.

## Create a simple search/results page set

You're now ready to build the employee directory for the Compass intranet site. In this part of the tutorial, you'll create a simple search/results page set that lets Compass employees look up their coworkers online using a single search parameter. Building this simple page set requires no knowledge of SQL (Structured Query Language).

The search and results pages have been laid out for you. Your job is to make them work together. Later, you'll extend the application with a detail page to display more information about each employee (including the employee's photo) and a page to insert new employee records in the database.

All the information about Compass employees is stored in the Microsoft Access database file, `compasstravel.mdb`. By now you should have a connection to this database. If not, see "Create a database connection" on page 50. The tutorial will not work without a connection to this database.

### Prepare the search page

You need a search page to obtain search parameters from the user. A simple search page uses an HTML form to get a single search parameter and submit it to the results page on the server. This results page, not the search page, conducts the actual search and displays the results.

In the HTML form, specify the results page that will use the search parameter collected in the form.

- 1 Make sure the Site window is open.

Choose Window > Site Files to display the Site window.

- 2 Make sure your tutorial site is selected, then double-click the `Search.htm` file under Local Folder.

The simple search page for the Compass employee directory opens. This search page lets users conduct a search by department.



- 3 In the Document window, select the form by clicking the Department list/menu form object, then clicking the rightmost `<form>` tag on the tag selector.



The form is selected.

- 4 Make sure the Property inspector is open.  
Choose Window > Properties to open the Property inspector.
- 5 In the Action box, click the folder icon and choose the file called Results.
- 6 In the Method pop-up menu on the Property inspector, choose GET.  
Choosing GET ensures that the search parameters are passed to the server in the URL string used to open the results page. You'll learn more about using the URL to pass parameters later in this tutorial.
- 7 Rename the list/menu form object by clicking it then entering **mnuDept** in the List/Menu box on the Property inspector (Window > Properties).
- 8 Save your work (File > Save).

That's all for the search page. You're ready to build the results page, which involves defining a recordset to hold the search results and displaying the search results on the page.

### Define a filtered recordset for the results page

A recordset is a subset of data extracted from one or more tables in a database. It acts as a source of data for your dynamic pages.

You will define a recordset that contains only the records returned by the search. For a simple search/results page set, you can create this kind of recordset by defining a filter.

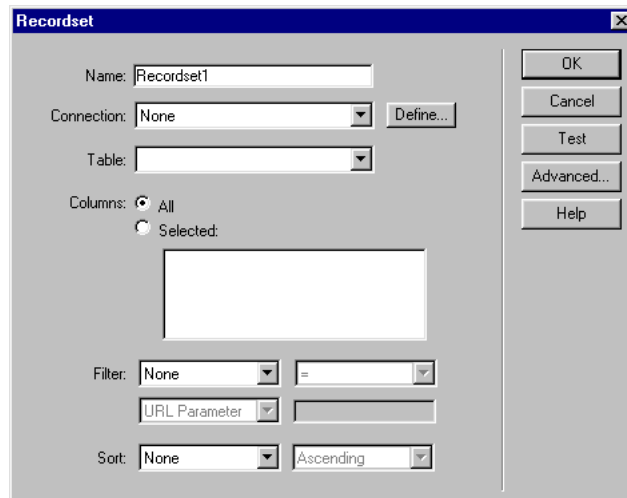
You'll begin by defining the general outlines of the recordset.

- 1 In the Site window (Window > Site Files), double-click the Results file under Local Folder.

The simple results page for the Compass employee directory opens in UltraDev.

- 2 In the Data Bindings panel (Window > Data Bindings), click the Plus (+) button and select Recordset (Query) from the pop-up menu.

The simple Recordset dialog box appears.

The image shows a 'Recordset' dialog box with a blue title bar and a close button. It contains several input fields and buttons. The 'Name' field is set to 'Recordset1'. The 'Connection' dropdown is set to 'None', with a 'Define...' button next to it. The 'Table' dropdown is empty. Under 'Columns', the 'All' radio button is selected. There is a large empty rectangular box below the column options. The 'Filter' section has a dropdown set to 'None', followed by an equals sign and another dropdown. Below that is a checkbox labeled 'URL Parameter' which is checked. The 'Sort' section has a dropdown set to 'None' and another dropdown set to 'Ascending'. On the right side, there are five buttons: 'OK', 'Cancel', 'Test', 'Advanced...', and 'Help'.

If the advanced Recordset dialog box appears instead, click Simple.

- 3 In the Name box, enter **Results**.

This is the name of your recordset. Do not confuse it with the file name of your page. In this case, they happen to be the same.

- 4 Select the connCompass connection from the Connection pop-up menu.

If it doesn't appear in the menu, click the Define button to create it. For instructions, see "Create a database connection" on page 50.

- 5 In the Table pop-up menu, select EMPLOYEES.

- 6 In the Columns area, click the Selected option to choose selected columns in the EMPLOYEES table.

- 7 Control-click (Windows) or Command-click (Macintosh) the following columns in the list to include them in the recordset: EMPLOYEEID, FIRSTNAME, LASTNAME, DEPARTMENT, and EXTENSION.

The results page displays information from four of these columns. You include the EMPLOYEEID column because it contains information identifying individual records—information that will become useful later when you build a detail page.

- 8 Click Test to test the recordset.

A recordset appears containing data extracted from the database table. Click OK to close it.

By default, the recordset contains all the records in the database table. Leave the Recordset dialog box open for now. You'll use it next to create a filter to remove all the records that don't meet the search criteria.

#### To create the recordset filter:

- 1 From the first pop-up menu in the Filter area, choose DEPARTMENT.

You know the search parameter submitted by the search page is a department name. Therefore, you set up the filter so that it compares this name against each name in the DEPARTMENT column in the database table. If the filter finds a match, the record is included in the search results.

- 2 From the pop-up menu beside the first menu, select the equal sign (it should be the default).

This choice states that you want only those records with DEPARTMENT column values that match exactly the department name submitted by the search page.

- 3 From the third pop-up menu, select URL Parameter (it should be the default).

Earlier on your search page, you specified the GET method for the HTML form, ensuring that the search parameter is passed to the server in the URL string. The server reads the string and stores the search parameter in memory as a URL parameter.

- 4 In the fourth box, enter **mnuDept**, the name of the form object used to collect the search parameter on the search page.

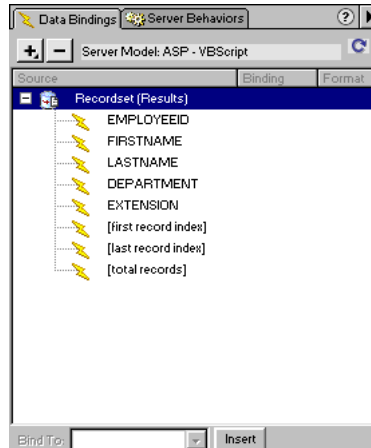
Here's how your Filter section should look:



Filter:	DEPARTMENT	=
	URL Parameter	mnuDept

**5** Click OK.

UltraDev adds the recordset to your list of available data sources in the Data Bindings panel. To view the fields you defined for the recordset, expand the recordset branch.



**6** Save your work (File > Save).

The next step is to display the results on the page.

## Add dynamic content to the results page

After defining a filtered recordset, you can use its columns as sources of dynamic content for your page. For the results page, you'll use the following columns (fields) as data sources: FIRSTNAME, LASTNAME, DEPARTMENT, and EXTENSION.

- 1 Make sure the Data Bindings panel is open (Window > Data Bindings) and lists the Results recordset you just defined. Expand the recordset's branch to see the data sources you need—namely, FIRSTNAME, LASTNAME, DEPARTMENT, and EXTENSION.

If these columns don't appear in the list, click the Plus (+) button to define a new recordset. For instructions, see “Define a filtered recordset for the results page” on page 57.

You'll begin by adding the LASTNAME data source to the page.

- 2 On the page, double-click the word “LAST” to select it.

- 3 In the Data Bindings panel, select LASTNAME and click Insert, or drag LASTNAME onto the text you selected on the page.

A data placeholder replaces the text selection on the page and the necessary server-side scripts are added to the page's HTML source code. When the server runs the page, the placeholder will be replaced with live data from the recordset.

- 4 Repeat steps 2 and 3 to replace the word "First" with the FIRSTNAME data source, the word "dept" with the DEPARTMENT data source, and the word "Ext" with the EXTENSION data source.

- 5 Save your work (File > Save).

If you make a mistake, open the Server Behaviors panel (Window > Server Behaviors), select the dynamic content (one of the Dynamic Text items), and click the Minus (-) button to delete it.

## View your work in the Live Data window

To display the page in the Live Data window, you must provide it with a URL parameter because the page is expecting one from a search page. For testing purposes, you can simulate a search query by choosing Live Data Settings from the View menu, clicking the URL Request Plus (+) button, and entering the following values:

- Name: **mnuDept**
- Value: **Trip Staff**

Click OK to close the Live Data Settings dialog box, then choose View > Live Data. The Live Data window shows the first record in the recordset. By default, a dynamic page displays only a single record. Naturally, you'd like to show all the records found, not just one.

## Add a repeated region to the results page

A search often returns more than one record. In this part of the tutorial, you'll give your page the ability to display more than one record by adding a repeated region to the page.

You create a repeated region by applying the Repeat Region server behavior to a page element—in this case, a table row. When the results page runs on your server, the Repeat Region server behavior will repeat the table row to accommodate all the records in the Results recordset. Each row will display the content of one employee record.

- 1 Select the table row with the dynamic content by clicking anywhere in the row and clicking the rightmost `<tr>` tag in the tag selector.

The table row is outlined.

- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Repeat Region.
- 3 In the Repeat Region dialog box, make sure the Results recordset is selected.
- 4 In the Show area, enter 5.

The page will only display five records at a time even if more than five records are found. The user will need to click the Next icon to see the other records. (You'll activate the Next icon later in the tutorial.)

- 5 Click OK.

- 6 View the page in the Live Data window.

If you're already working in the Live Data window, click the Refresh button (the circular arrow) on the toolbar. If you're working in the Document window, choose View > Live Data.

Where only one of the search results was displayed, now five records are displayed.

- 7 Save your work (File > Save).

If you make a mistake, open the Server Behaviors panel (Window > Server Behaviors) and double-click the repeat region in the list to edit it.

## Activate the recordset navigation links

The results page should allow users to move forward and backward through the results if the search returns more than five results. The tutorial page includes "Previous" and "Next" images for this purpose. Your job is to activate these images so that when a user clicks one, the page displays more results (if any more exist).

- 1 Select the "Previous" image on the page.

- 2 In the Server Behaviors panel, click the Plus (+) button and select Move to Record > Move to Previous Record.

The Move to Previous Record dialog box appears.

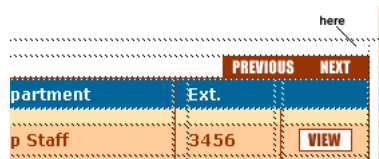
- 3 In the Recordset pop-up menu, make sure the Results recordset is selected, then click OK.
- 4 Repeat the procedure for the “Next” image, only this time select Move to Record > Move to Next Record from the pop-up menu.
- 5 Save your work (File > Save).

If you make a mistake, select the server behavior in the Server Behaviors panel and click the Minus (-) button to delete it.

## Add a record counter to the results page

If your searches are likely to produce lots of results, you can add a record counter to your page to help users keep track of where they are in the recordset. You can create a record counter in a single operation using an UltraDev live object.

- 1 Place the insertion point in the row above the “Previous” and “Next” images.



- 2 Choose Insert > Live Objects > Recordset Navigation Status.
- 3 Make sure your Results recordset is selected.
- 4 Click OK.
- 5 Save your work (File > Save).

UltraDev builds and adds the record counter to your page. Except for the dynamic content, the record counter is fully customizable.

To view the completed results page in the Live Data window, click the Refresh button on the Live Data window's toolbar. If you're working in the Document window, choose View > Live Data.

**Note:** The “Previous” and “Next” links don't work in the Live Data window. To test them, you must upload the page to your published directory on the server and open the page in a browser.

## Create an advanced search/results page set

In this part of the tutorial, you'll create an advanced search/results page set that lets Compass employees look up their coworkers by using more than one search parameter. Building an advanced search/results page set in UltraDev requires some knowledge of SQL (Structured Query Language). The tutorial provides the SQL statement required to make the pages work.

### Prepare the advanced search page

The advanced search page uses an HTML form to get multiple search parameters and submit them to the results page on the server. This results page, not the search page, conducts the actual search and displays the results.

In the HTML form, specify the results page that will use the search parameters collected in the form.

- 1 Make sure the Site window is open.

Choose Window > Site Files to display the Site window.

- 2 Double-click the SearchAdv.htm file under Local Folder.

The advanced search page for the Compass employee directory opens in UltraDev. This search page lets users conduct a search using the employee's last name, department, or both.

- 3 In the Document window, select the form by clicking the Department list/menu form object, then clicking the rightmost `<form>` tag in the tag selector.

- 4 Make sure the Property inspector is open.

Choose Window > Properties to open the Property inspector.

- 5 In the Action box, click the folder icon and choose the ResultsAdv file.

- 6 In the Method pop-up menu, choose GET.

Choosing GET ensures that the search parameters are sent to the server in the URL string used to open the results page.

- 7 Rename the form objects as follows:

- Click the text field and enter **txtLastName** in the Text Field box on the Property inspector.
- Click the list/menu object and enter **mnuDept** in the List/Menu box on the Property inspector.

- 8 Save your work (File > Save).

That's all for the search page. Next you'll build a results page that can handle multiple search parameters.



## Define a recordset for the advanced results page

You'll add a recordset to the results page that contains only the records returned by the search. For an advanced search/results page set, you create this kind of recordset by writing a SQL statement that uses the search parameters passed by the search page. The search parameters are stuffed in variables in the SQL statement, which is then used to generate the recordset.

- 1 In the Site window, double-click the ResultsAdv file under Local Folder.

The advanced results page for the Compass employee directory opens in UltraDev.

- 2 In the Data Bindings panel (Window > Data Bindings), click the Plus (+) button and select Recordset (Query) from the pop-up menu.

The simple Recordset dialog box appears. Click the Advanced button to switch to the advanced Recordset dialog box.

- 3 In the Name box, enter **Results**.

- 4 In the Connection pop-up menu, select connCompass.

If it doesn't appear in the list, click the Define button to create it. For instructions, see "Create a database connection" on page 50.

- 5 In the Database Items area at the bottom of the dialog box, expand the Table branch, then expand the EMPLOYEES branch.

Next, you'll build the Select clause.

- 6 Select EMPLOYEEID in the Database Items area and click the Select button.

- 7 Select FIRSTNAME and click the Select button.

- 8 Select LASTNAME and click the Select button.

- 9 Select DEPARTMENT and click the Select button.

- 10 Select EXTENSION and click the Select button.

- 11 In the SQL text area, enter the following line below FROM EMPLOYEES:

```
WHERE LASTNAME LIKE 'varLastName' AND DEPARTMENT LIKE 'varDept'
```

The full statement in the SQL text area should look as follows:

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, DEPARTMENT, EXTENSION ↵  
FROM EMPLOYEES WHERE LASTNAME LIKE 'varLastName' AND ↵  
DEPARTMENT LIKE 'varDept'
```

For help on understanding this SQL syntax, see "SQL Primer" on page 251.

- 12 “Stuff” the two SQL variables, `varLastName` and `varDept`, with the values of the search parameters passed by the search page by clicking the Plus (+) button in the Variables area and entering the following information.

- For the ASP tutorial:

Name	Default Value	Run-time Value
<code>varLastName</code>	<code>%</code>	<code>Request("txtLastName")</code>
<code>varDept</code>	<code>%</code>	<code>Request("mnuDept")</code>

- For the ColdFusion tutorial:

Name	Default Value	Run-time Value
<code>varLastName</code>	<code>%</code>	<code>#txtLastName#</code>
<code>varDept</code>	<code>%</code>	<code>#mnuDept#</code>

- For the JSP tutorial:

Name	Default Value	Run-time Value
<code>varLastName</code>	<code>%</code>	<code>request.getParameter("txtLastName")</code>
<code>varDept</code>	<code>%</code>	<code>request.getParameter("mnuDept")</code>

The default value is the value the variable should take if no run-time value is returned. The run-time value is usually a server object holding a value sent by the search page.

- 13 If you're satisfied with the recordset, click OK.

UltraDev adds the recordset to your list of available data sources in the Data Bindings panel.

When the SQL query runs on the server, each record in the database table is checked. If the specified field in a record meets the Where condition, the record is included in the recordset. The query in effect builds a recordset containing only the search results.

## Display the results on the page

After defining the recordset, you can use its columns as sources of dynamic content for your page. The procedure for adding the data to the page is identical to the procedure for the simple result page. For instructions, see “Add dynamic content to the results page” on page 60.

To view your work in the Live Data window, you must provide the page with some test parameters to simulate the URL parameters submitted by your search page. Provide those test parameters by choosing View > Live Data Settings and entering the following values:

Name	Value
txtLastName	Nicholas
mnuDept	Trip Staff

Click OK to close the dialog box, then choose View > Live Data.

Give your page the ability to display more than one record by creating a repeated region on the page and activating the “Next” and “Previous” recordset navigation images. The steps are the same as for the simple results page. See “Add a repeated region to the results page” on page 62 and “Activate the recordset navigation links” on page 62.

Finally, add a record counter to help users keep track of where they are in the recordset. See “Add a record counter to the results page” on page 63.

## Create a detail page

By now you should have created at least one results page that can list employees. In this part of the tutorial, you’ll create a detail page that displays more information about each employee listed on the results page. The detail page will even display a photograph of the employee.

Here’s how this “master/detail” set of pages should work: A user viewing the list of employees on the results (master) page should be able to see more details about any listed employee by clicking a link specific to that employee. The link opens a detail page showing more details about the selected employee.

By making the detail page dynamic, you don’t have to create a separate detail page for each employee. You create one detail page with content that changes at run time depending on the link the user clicks on the results page.

The first step in setting up this master/detail page set is to add a link on the results page to open the detail page.

## Add a link on the results page

Clicking an employee link on your results page should open a detail page displaying more information about the employee. However, using a standard link to open the detail page won't work: to retrieve the correct record, the detail page needs to know what employee the user selected on the results page. In other words, the results page must pass information to the detail page.

Use the following steps to add a link that passes information from the results page to the detail page. Perform this procedure for both versions of your results pages (the files called Results and ResultsAdv).

- 1 Make sure your results page is open in UltraDev.
- 2 On the results page, click the “View” image to select it.  
If you're working in the Live Data window and it displays multiple records, select the first “View” image in the column.
- 3 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Go to Detail Page from the pop-up menu.  
The Go to Detail Page dialog box appears.
- 4 In the Detail Page box, click Browse and select the file called Detail (the predesigned detail page), then click OK.
- 5 In the Pass URL Parameter box, enter a parameter called `id` and make sure the parameter is set to the value of the EMPLOYEEID column in the Results recordset.

You're telling the page to pass a parameter called `id` to the detail page. The parameter identifies the employee selected by the user. The server will set the parameter's value to the value specified in the Recordset and Column pop-up menus.

- 6 Click OK and save your work (File > Save).

When the user clicks the linked “View” image on the results page, not only will the detail page open, but information identifying the record the user chose will be passed to the detail page so it can display the correct employee details.

You're finished with the results page. Now you need a detail page that can display detailed information about employees selected on the results page.

## Define a recordset for the detail page

You'll define a recordset for the detail page to hold the detailed employee information. Begin by opening the predesigned detail page.

- 1 In the Site window, double-click the Detail file under Local Folder.

The Compass detail page opens in UltraDev.

- 2 In the Data Bindings panel (Window > Data Bindings), click the Plus (+) button and select Recordset (Query) from the pop-up menu.

The simple Recordset dialog box appears. If the advanced Recordset dialog box appears instead, click Simple.

- 3 In the Name box, enter **Details**.

- 4 In the Connection pop-up menu, choose connCompass.

- 5 In the Table pop-up menu, choose EMPLOYEES.

- 6 In the Columns area, make sure the All option is selected.

For this page, you want to retrieve data in all the columns in the table.

- 7 Click Test to test the recordset.

A recordset appears containing data extracted from the database table. Click OK to close it.

- 8 Click OK and save your work (File > Save).

## Add dynamic content to the detail page

After defining the recordset, you can use its columns as sources of dynamic content for the detail page.

- 1 Make sure the Data Bindings panel is open (Window > Data Bindings) and lists the Details recordset you just defined. Expand the recordset's branch to see the data sources you need.
- 2 On the detail page, double-click the text string LastName (in the blue table row) to select it.
- 3 In the Data Bindings panel, select the LASTNAME field and drag it onto the selected string on the page.

4 Repeat steps 2 and 3 (select text on page, then drag the corresponding data source to it) for the other text strings on the page, as follows:

- Drag FIRSTNAME to FirstName
- Drag PHONE to number
- Drag STARTDATE to date
- Drag DEPARTMENT to dept
- Drag EXTENSION to ext
- Drag EMAIL to email
- Drag NOTES to notes

Next, you'll bind the source attribute of the image on the page to display employees' photographs. The source attribute consists of a text string such as `jones_lyn.jpg` stored in the database.

5 Open the Property inspector (Window > Properties).

6 Click the image on the page, then click the small folder icon beside the Src box on the Property inspector.

The Select Image Source dialog box appears.

7 Select the Data Sources option.

A list of data sources appears.

8 Select PHOTO from the list.

9 In the URL box, type **images/** at the start of the line of code.

The Compass database contains the file names of the images, not the folder where they are stored on your site. The code in the URL box will retrieve the correct file name from the database and insert it in the image's source attribute. Typing **images/** in the URL box adds the correct path to the attribute.

10 Click OK to close the dialog box.

11 Save your work (File > Save).

Choose View > Live Data to display the dynamic content. The Live Data window displays the details of the first employee in the recordset.

If you make a mistake, open the Server Behaviors panel (Window > Server Behaviors), select the dynamic content, and click the Minus (-) button.

## Enable the page to find and display an employee

When the detail page opens in a browser, it should display detail information about an employee the user selected on the results page. (You worked on results pages earlier in this tutorial.) To make the page work this way, you use the Move to Specific Record server behavior, which finds and displays the employee the user selected on the results page.

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Move to Record > Move to Specific Record.

The Move to Specific Record dialog box appears.

- 2 In the Move to Record In pop-up menu, make sure the Details recordset is selected.
- 3 In the Where Column pop-up menu, make sure the EMPLOYEEID column is selected.

The previous page you worked on passed the ID number (id=EMPLOYEEID) of an employee record to the detail page. By specifying the EMPLOYEEID column, you tell the detail page to look in the EMPLOYEEID column of the current recordset to find an ID number matching the one sent by the results page. When the behavior's server-side script finds a match, it displays the corresponding employee record.

- 4 Click OK.

If you make a mistake, open the Server Behaviors panel (Window > Server Behaviors), and double-click the server behavior to edit it.

## Create an insert page

The final page you'll create for the Compass employee directory is a page that lets the site administrator insert new employee records in the database with a Web browser.

An insert page consists of two building blocks:

- An HTML form that lets users enter data
- An Insert Record server behavior that takes the data entered in the form and updates the database

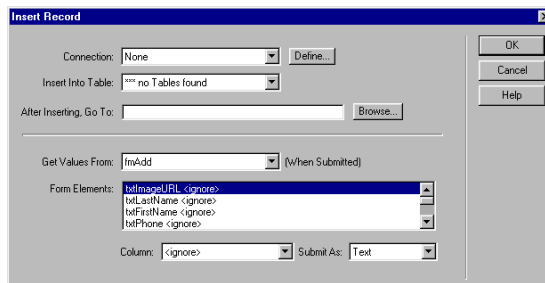
The HTML form has already been created for you. You'll add the server behavior to update the database table.

- 1 In the Site window, double-click the file called Insert under Local Folder.

The insert page for the Compass employee directory opens in UltraDev.

- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Insert Record from the pop-up menu.

The Insert Record dialog box appears.



- 3 In the Connection pop-up menu, choose connCompass.
- 4 In the Insert Into Table pop-up menu, choose EMPLOYEES.
- 5 In the “After Inserting, Go To” box, click Browse and select the InsertOK file.

This page will open after the record is successfully inserted into the table.

- 6 In the Get Values From pop-up menu, make sure fmAdd is selected.  
fmAdd is the name of HTML form on the page.



- 7 Specify what each object on your form will update in the database table by selecting the first form element in the Form Elements list (txtImageURL) and selecting the corresponding item in the Column pop-up menu (PHOTO).

Repeat this step for the other form elements, as follows:

- txtLastName updates the LASTNAME column
- txtFirstName updates the FIRSTNAME column
- txtPhone updates the PHONE column
- txtDate updates the STARTDATE column
- txtDept updates the DEPARTMENT column
- txtExt updates the EXTENSION column
- txtEmail updates the EMAIL column
- txtNotes updates the NOTES column

- 8 Click OK.

With the insert page completed, the employee directory for the Compass intranet site is done. In the Site window, select all the pages, then click the blue up-arrow on the toolbar to upload them to your published directory on the server. Launch your browser and open the Search.htm page.



## CHAPTER 2

### UltraDev Basics

.....

To get the most out of Macromedia Dreamweaver UltraDev, you should be familiar with the following topics:

- How dynamic pages work in general
- The workflow involved in creating a dynamic page in UltraDev
- The UltraDev working environment

This chapter looks briefly at each of these topics. First, here's some basic terminology:

**A Web application** is a collection of static and dynamic pages that interact with each other and with various resources on a Web server, including databases.

**A dynamic page** is a Web page modified at run time by the Web server before being sent to a browser.

**A server technology** is a technology such as ASP, JSP, or ColdFusion that gives the Web server the ability to modify a Web page at run time.

**A server behavior** is the set of instructions executed on the server at run time. Server behaviors are inserted in the Web page at design time.

If you're new to the world of databases or database connections, see "Beginner's Guide to Databases" on page 225.

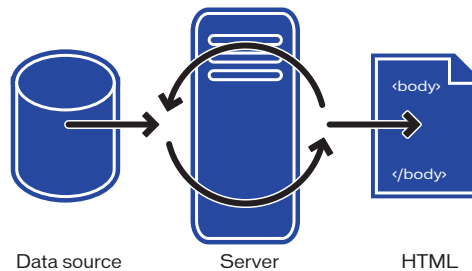
## About dynamic pages

A Web application is a collection of static and dynamic pages. Dynamic pages resemble static pages in all aspects except one: where some of their scripts are run. Both kinds of pages are plain text (ASCII) files, contain HTML, and sit on a server waiting to be served up to a Web browser. Both can contain scripts written in languages like VBScript or JavaScript. However, certain scripts in a dynamic page can run on a server while those in a static page cannot.

**Note:** Strictly speaking, a “static” page may not be static at all. For example, a rollover image or a Flash movie can make a static page come alive. However, this guide refers to a page as static if it doesn’t contain scripts that are executed on a server.

Scripts that run on a server, or server-side scripts, give you the ability to work with resources such as databases on the server. For example, before a page is served up to the browser, a server-side script in the page could instruct the server to extract data from a database and insert it into the page’s HTML. In UltraDev, server-side scripts are called server behaviors.

Here’s the route data takes to make it into the HTML of your pages:



In effect, the server creates a part of your page during run time and adds it to the parts you designed earlier in UltraDev. The resulting page is then sent to the browser.

UltraDev supports the following server technologies:

- Microsoft Active Server Pages (ASP). To learn more about ASP, visit the Microsoft Web site at <http://msdn.microsoft.com/workshop/server/toc.htm>.
- Sun JavaServer Pages (JSP). To learn more about JSP, visit the Sun Web site at <http://java.sun.com/products/jsp/>.
- Allaire ColdFusion. To learn more about ColdFusion, visit the Allaire Web site at <http://www.allaire.com/Products/ColdFusion/productinformation/>.

A Web application can have many kinds of dynamic pages. The most common are search pages, results pages, detail pages, and record-editing pages (which allow users to insert, update or delete records in a database). Each type of page has different requirements in terms of HTML code, dynamic content, and server behaviors. These requirements are covered in the following chapters:

- “Building Pages That Search Databases” on page 165
- “Building Pages That Edit Database Records” on page 181
- “Building Pages That Restrict Access to Your Site” on page 199

## About the UltraDev workflow

All dynamic pages start as blank or static pages. You can first build a static page, then transform it into a dynamic one. For example, you could create a page with a logo, some introductory text, a site map, and a table. Next, you could modify the table to display information from a database.

The workflow for creating a dynamic page consists of four distinct phases:

- Laying out the page
- Defining an UltraDev data source such as a recordset
- Adding dynamic content to the page from the UltraDev data source you defined
- Adding server behaviors to give the page more functionality

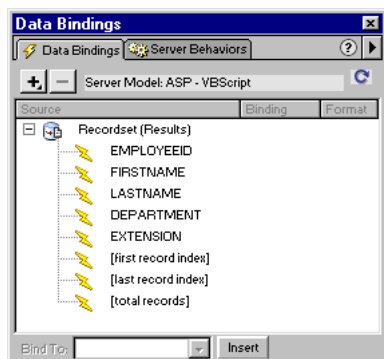
This section describes the workflow in general terms. (The *Using Dreamweaver* book describes how to lay out a page in detail.) To see an animated overview of the UltraDev workflow, choose Help > Guided Tour.

### Defining an UltraDev data source

The first step in the process is to define an UltraDev data source. An UltraDev data source is a store of information from which you can pick and choose data to include in your Web page.

If you decide to use a database with your application, you must define an UltraDev data source to store data from the database. This data source is called a recordset in ASP and ColdFusion, and a resultset in JSP. (This guide uses the term *recordset* to refer to all three.)

Any recordset you define is added to your list of data sources in the Data Bindings panel:



You use this panel to add dynamic content to your page.

For detailed procedures, see the following chapters:

- “Defining UltraDev Data Sources” on page 109
- “Creating a Recordset” on page 125

## Adding dynamic content

After adding a recordset or other data sources such as server variables to the Data Bindings panel, you can add dynamic content to your page. Dreamweaver UltraDev lets you add dynamic content without worrying about the underlying server-side scripts inserted into the page. You simply need to specify where you want to put the content, and what you want that content to be.

First, you specify where you want to put the dynamic content. In UltraDev, you can put dynamic content anywhere in the page’s HTML:

- You can place it at the insertion point.
- You can replace a text string with it.
- You can insert it in an HTML attribute. For example, dynamic content can define the `src` attribute of an image or the `value` attribute of a form field.

Second, you specify what you want the dynamic content to be. You can choose from any data source listed in your Data Bindings panel. For example, you could choose a field in a recordset, a value submitted by a requesting page, or a value of a server object. After making your selection, UltraDev inserts a server-side script into the page that instructs the server to transfer the data from the selected data source to the page’s HTML code.

For detailed procedures, see “Adding Dynamic Content” on page 133.

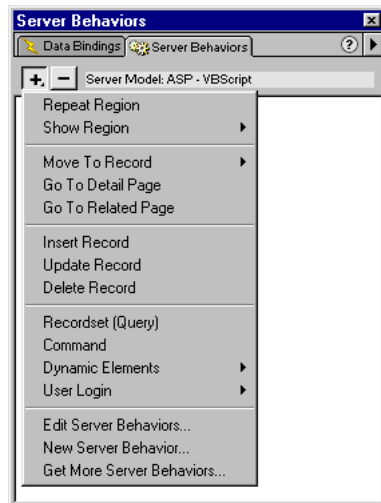
## Adding server behaviors

The next step in the process is supplying the page with the “intelligence” it needs to be more functional. In many cases, you supply this intelligence by adding server behaviors to the page. A server behavior consists of VBScript, JavaScript, Java, or ColdFusion code that runs on a server instead of in a browser.

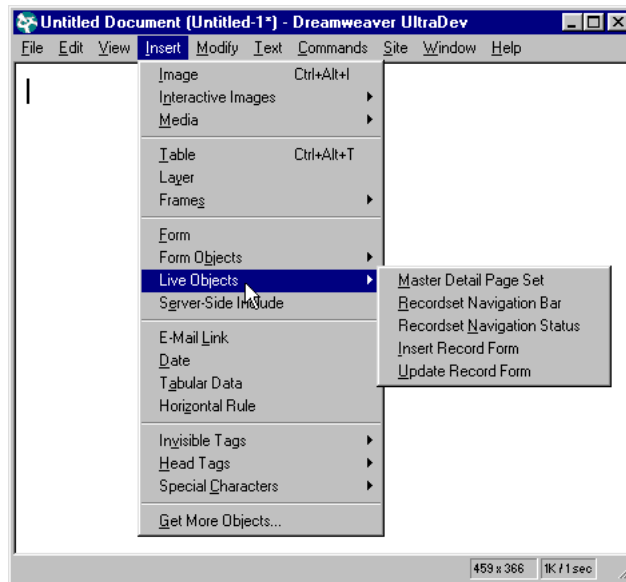
UltraDev comes with a number of predefined server behaviors to power your application. For example, after creating a page to display a database record, you can use a server behavior to display the next or the previous record in the database. You can also write your own server behaviors or install server behaviors written by other people.

UltraDev also provides you with live objects that insert multiple server behaviors into your Web pages at one time to create advanced page components, such as a record navigation bar or a master/detail page set.

You add server behaviors to your pages using the Server Behaviors panel.



You can also add advanced components to your page in one operation by using live objects.



For detailed procedures, see the following chapters:

- “Displaying Database Records” on page 147
- “Building Pages That Search Databases” on page 165
- “Building Pages That Edit Database Records” on page 181
- “Building Pages That Restrict Access to Your Site” on page 199



## About the UltraDev working environment

UltraDev provides you with several working environments:

- You can work in the Document window's Design view.
- You can work in the Live Data window's Design view, which displays dynamic content.
- You can preview the page in a browser to test how the pages in your application interact.
- You can work directly on the source code by using the Code view in the Document or Live Data windows, the Code inspector, the Quick Tag Editor, or your favorite text editor.

### Working in the Document window's Design view

The Document window's Design view is the traditional editing environment in Dreamweaver. When you open a document in UltraDev, the Document window opens by default. The window offers three views: a Design view, a Code view, and a split-screen Design/Code view. For more information on the Code view, see "Using the Code view" on page 87.

The Document window's Design view (View > Design) gives you a rough idea of what the page will look like in the browser *before* dynamic content is added to the page. This is an ideal editing environment for pages with no dynamic content. However, since dynamic content may fundamentally alter the way a page looks and works, this environment is not ideal for dynamic pages.

You can still use the Document window to work on dynamic pages. Dynamic content in the Document window's Design view is represented by text placeholders, as in this example:

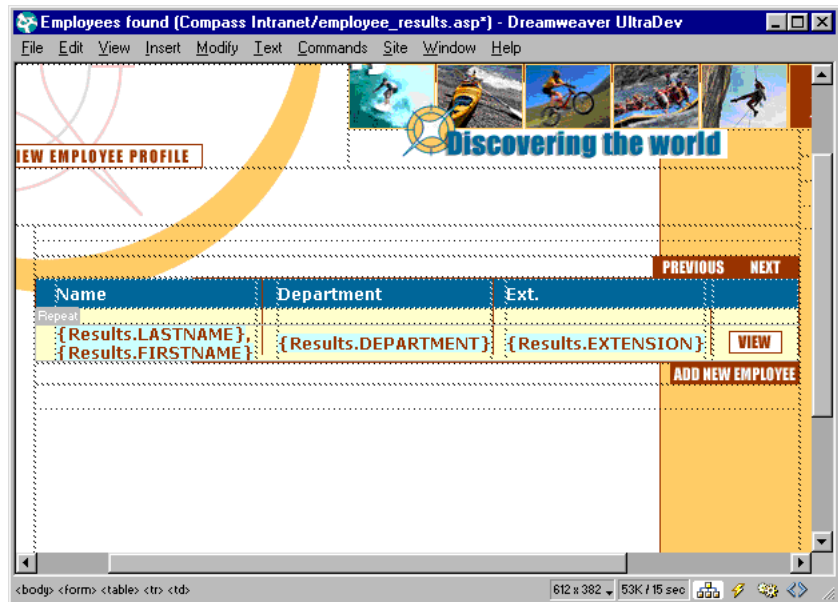
```
{rsMembers.LastName}
```

This placeholder tells you that the data from the `LastName` column in the `rsMembers` recordset will be inserted at this location on the page when the page runs on the server.

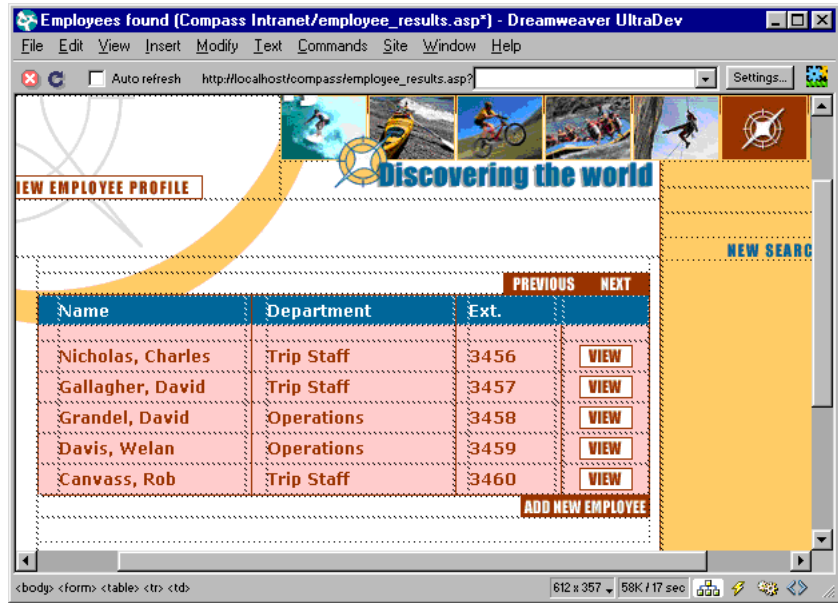
Repeated regions are surrounded by a thin, tabbed gray outline while most of the other server behaviors applied to the page have no effect on the page, nor are they visible in the Document window.

## Working in the Live Data window

You can use the UltraDev Live Data window to work on your pages in a live data environment. Unlike the Document window, which uses placeholders to represent dynamic content on the page, the Live Data window displays the actual dynamic content. For example, here is a dynamic page in the Document window:



Here is the same page displayed in the Live Data window:



**Note:** Links don't work in the Live Data window. To test your links, use the UltraDev Preview in Browser feature. (See "Using Preview in Browser" on page 86.)

While dynamic content is displayed, you can do the following:

- Adjust the page's layout using the Dreamweaver page-design tools
- Add, edit, or delete dynamic content
- Add, edit, or delete server behaviors

To achieve this effect, UltraDev runs the dynamic page on your server before displaying it in the Live Data window. Whenever you switch to the Live Data window, a temporary copy of the open document is sent to your application server for processing. The resulting page is returned and displayed in the Live Data window, and the temporary copy on the server is deleted.

You can toggle between the Document window and the Live Data window by choosing Live Data from the View menu. If a page expects data from the user—for example, the ID number of a record selected in a master page—you can provide the page with that data yourself in the Live Data Settings dialog box.

You must upload all the necessary files, including server-side includes and dependent files such as image files and JSP class files, to the application server. UltraDev does not automatically copy dependent files to the server when you switch to the Live Data window.

**Note:** The UltraDev Live Data window supports code in server-side includes (Insert > Server-Side Include) and application files like `global.asa` (ASP) and `application.cfm` (ColdFusion). Make sure to upload these files to the server before switching to the Live Data window.

**To copy dependent files to the application server:**

- 1 In the Site window (Site > Site Files), click the Application Server icon on the toolbar (the second icon from the left).

The application server's root folder appears under Remote Site.

- 2 Under Local Folder, select the dependent files.
- 3 Click the blue up arrow on the toolbar to copy the files to the application server, or drag the files to the appropriate folder under Remote Site.

You need to do this only once for your site unless you add more dependent files, in which case you must upload them to the Web server too.

**To work on a page in the Live Data window:**

- 1 Make sure the Document window displays a dynamic page.

In the Document window, placeholders are used for all dynamic content.

- 2 Choose View > Live Data to switch to the Live Data window.

UltraDev must run a temporary copy of the page on a server before displaying the page and its dynamic content. The process may take a few seconds. To cancel the process, click the Stop button (the red button with the white X).

If successful, the Live Data window appears with dynamic content displayed on the page.

**Note:** If the page displays an error message, make sure the URL prefix in the Site Definition dialog box is correct. See "Specifying a URL prefix" on page 27.

- 3 If desired, choose View > Visual Aids > Invisible Elements to remove the highlighting applied to dynamic content.

The highlighting may affect how dynamic content is displayed and thus give you an inaccurate picture of the page.

- 4 If desired, select the Auto Refresh option on the toolbar.

The page will refresh whenever you make a change affecting dynamic content. If you have a slow database connection, you may want to leave this option off when working in the Live Data window.

- 5 Make the necessary changes to the page.
- 6 If your page expects values from an HTML form using the GET method, enter the values in the text box on the toolbar and click the Refresh button (the circle-arrow icon).

**Note:** A text box for entering values appears only if you specified the GET method in the Live Data Settings dialog box (View > Live Data Settings).

Enter the test data in the following format:

*name=value;*

In this format, *name* is the variable name expected by your page and *value* is the value held by that variable.

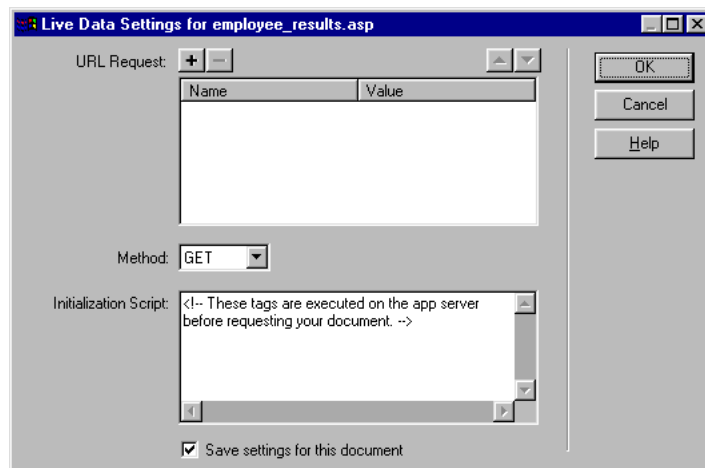
You can also define name/value pairs in the Live Data Settings dialog box (View > Live Data Settings) and save them with the page.

- 7 Click the Refresh button if your page needs refreshing.

**To provide the page with data expected from users:**

- 1 In the Document window, choose Live Data Settings from the View menu.

The Live Data Settings dialog box appears.



- 2 In the URL Request area, click the Plus (+) button to enter a variable your page expects. Specify a name and a test value for each variable.
- 3 In the Method pop-up menu, select the HTML form method your page expects: POST or GET.

- 4 In the Initialization Script text area, include any source code you want to insert at the top of the page before it runs. The code usually consists of one or more tags that initialize session variables.
- 5 To save your settings for the current page, click Save Settings For This Document.

**Note:** To save the settings, Design Notes (File > Design Notes) must be enabled.

- 6 Click OK.

## Using Preview in Browser

Use the Preview in Browser feature to test links in your application. The Preview in Browser command (File > Preview in Browser) lets you preview documents in a browser at any time. By default, UltraDev takes the document from the local file system and creates a temporary copy to display in your browser. However, because dynamic pages must be run on a server, UltraDev must run the temporary copy on a server before displaying it in the browser. (UltraDev then deletes the temporary file from the server.)

### To configure Preview in Browser for dynamic pages:

- 1 Choose Edit > Preferences, then select Preview in Browser.
- 2 Select the Preview Using Application Server option.

UltraDev uses the same application server used to generate pages for the Live Data window. See “Specifying a server technology” on page 25.

- 3 Click OK.
- 4 Upload any related pages, server-side includes, and dependent files to the server.

Preview in Browser only uploads a temporary copy of the page to the server. It does not upload related pages such as a results or a detail page, dependent files such as image files, or server-side includes. To upload a file, choose Site > Site Files to open the Site window, click on the Application Server icon, select the file under Local Folder, and click the blue up arrow on the toolbar to upload the file to your remote site.

### To open a page using Preview in Browser:

Open the page in either the Document window or the Live Data window, then choose File > Preview in Browser, or press F12.

## Working with the source code

You can write or edit code for your pages using the UltraDev Code view, the UltraDev Code inspector, or your favorite text editor. To edit single HTML tags without leaving the Document or Live Data window, you can use the Quick Tag Editor.

Text in the Code view and the Code inspector is color-coded. To change the color scheme, see “Editing HTML in Dreamweaver,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver). To change the color of script keywords and strings, in Preferences (Edit > Preferences), select Code Colors, and change the script colors.

You can also change the color of specific HTML tags, including the `script` tag. The color you set for the `script` tag is also used for the `<%` and `%>` delimiters and for all CFML tags. To change a tag’s color, select the tag in the list in the Tag Specific area, then pick a new color.

## Using the Code view

You can use the Code view (View > Code) to write or edit HTML and scripts in the Document and Live Data window. You can also display the Code view (View > Code and Design) in part of the Document and Live Data window. For more information, see “Editing HTML in Dreamweaver,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

**Note:** Changes made to static content in the Code view are reflected in the Design view only when you click in the Design view. In the Live Data window, changes made to dynamic content are reflected immediately in the Design view if the Auto Refresh option on the toolbar is selected. If the Auto Refresh option is not selected, you must click the Refresh button to view the changes in the Design view.

## Using the Code inspector

You can use the Code inspector to write or edit source code for your pages. To open the Code inspector, choose Window > Code Inspector. For more information on the inspector, see “Editing HTML in Dreamweaver,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

**Note:** Changes made to static content in the Code inspector are reflected in the Design view only when you click in the Design view. Changes made to dynamic content are reflected in the Live Data window only when you click the Refresh button in the window.

## Using the Quick Tag Editor

You can use the Quick Tag Editor to edit single HTML tags within either the Document or Live Data window by selecting text, an object, or a tag, then pressing Control+T (Windows) or Command+T (Macintosh).

For more information on the Quick Tag Editor, see “Editing HTML in Dreamweaver,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

## Using another text editor

If you want, you can use your favorite text editor to hand-code large amounts of HTML, JavaScript, VBScript, ColdFusion, or Java. Within UltraDev, you can open any external text editor, including Notepad (Windows), SimpleText (Macintosh), BBEdit, and HomeSite. To set up your external editor to work with UltraDev, see “Editing HTML in Dreamweaver,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

To open your external editor, press Control+E (Windows) or Command+E (Macintosh).



## CHAPTER 3

### Connecting to a Database

.....

If you plan to use a database with your Web application, you need to create at least one database connection. Without one, Dreamweaver UltraDev won't know where to find the database or how to connect to it. UltraDev lets you create any number of connections for your sites. You can also edit or delete any existing connection.

To learn more about databases and database connections, see "Beginner's Guide to Databases" on page 225.

### Creating a database connection for an ASP application

An ASP application can communicate with any OLE DB driver (or "provider"). The OLE DB provider in turn communicates with your database.

Because OLE DB providers are not yet common, ASP applications often use a special OLE DB provider that can communicate with any ODBC driver. The ODBC driver is then responsible for communicating with the database.

Creating a direct OLE DB connection can improve the speed of your connection. By using a database-specific OLE DB provider, you eliminate the ODBC middleman. Your OLE DB provider communicates directly with the database.

You can use a data source name (DSN) or a connection string to create an ODBC connection between your Web application and your database. If you want to create an OLE DB connection, you must use a connection string.

If you want to write a connection string to a file-based database on a remote server, you must know the full path to that database on the server. This information is not always available, especially if you're working with a commercial Internet service provider. You can use the `MapPath` method of the ASP session object to find a file's physical path.

## Creating a DSN connection

You can use a data source name (DSN) to create an ODBC connection between your Web application and your database. A DSN is a name containing all the parameters needed to connect to a specific database using an ODBC driver.

You define the connection parameters when you define the DSN. The parameters can include the server name, the path to the database or the database name, the ODBC driver to use, and the user name and password, if any. Once the DSN is defined, you can use it to invoke the underlying parameters.

For example, suppose you have a SQL Server database called `MedCenter` located on a server called `Socrates`. To gain access to the database, you must enter the user name `mwe1by` and the password `clooney7`. After using these parameters to define a DSN called `patients`, you can enter the single word `patients` in UltraDev to create the same connection.

**Note:** Since you can only specify an ODBC driver in a DSN, you must use a connection string if you want to use an OLE DB driver (or "provider"). For more information, see "Creating an OLE DB connection" on page 94.

The DSN can be defined on your local system if you're a Windows user, or on a remote system. If you want to use a local DSN, then your application server and database driver must be located on the local Windows system.

### To create a DSN connection if your application server runs locally (Windows users only):

- 1 Define a DSN on your local Windows system.

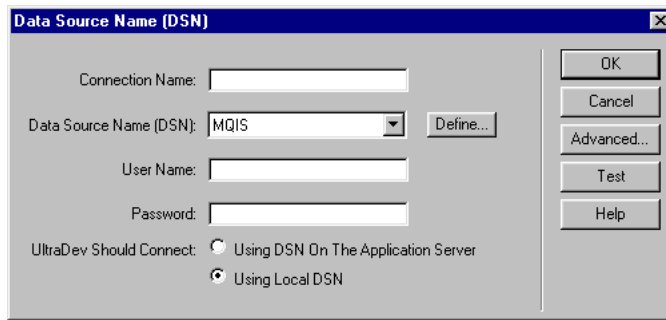
For instructions, see "Setting Up a DSN in Windows" on page 249.

- 2 In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

- 3 Click New and select Data Source Name (DSN) from the pop-up menu.

The Data Source Name (DSN) dialog box appears with the Using Local DSN option selected by default.



- 4 Enter a name for the new connection.

- 5 Select the DSN from the pop-up menu.

If you want to use a local DSN but haven't defined one yet, click Define to open the Windows ODBC Data Source Administrator. For instructions, see "Setting Up a DSN in Windows" on page 249.

- 6 If required, complete the User Name and Password boxes.

- 7 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.

For more information, see "Restricting the amount of information" on page 107.

**Note:** You cannot create a schema or catalog in Microsoft Access.

- 8 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check the URL prefix for the application server (see "Specifying a URL prefix" on page 27).

- 9 Click OK.

Your new connection should now appear in the Connections dialog box.

- 10 Click Done to close the Connections dialog box.

**To create a DSN connection if your application server runs on a remote server:**

- 1** Define a DSN on the system hosting your application server.  
For instructions, see “Setting Up a DSN in Windows” on page 249.
- 2** In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 3** Click New and select Data Source Name (DSN) from the pop-up menu.  
The Data Source Name (DSN) dialog box appears.
- 4** Enter a name for the new connection.
- 5** Enter the DSN.  
  
If you want, you can click the DSN button to connect to the server and retrieve the DSNs.  
  
**Note:** UltraDev can only retrieve server DSNs created with the Windows ODBC Data Source Administrator.
- 6** If required, complete the User Name and Password boxes.
- 7** If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.  
  
For more information, see “Restricting the amount of information” on page 107.  
  
**Note:** You cannot create a schema or catalog in Microsoft Access.
- 8** Click Test.  
  
UltraDev attempts to connect to the database. If the connection fails, double-check the DSN. If the connection still fails, check the URL prefix for the application server (see “Specifying a URL prefix” on page 27).
- 9** Click OK.  
  
Your new connection should now appear in the Connections dialog box.
- 10** Click Done to close the Connections dialog box.

## Creating a connection without a DSN

You can use a connection string to create an ODBC or OLE DB connection between your Web application and your database.

A connection string combines all the information your Web application needs on the server to connect to a database. UltraDev inserts this string in your page’s server-side scripts to be processed later by your application server.

Here's an example of a connection string:

```
Driver={Microsoft Access Driver (*.mdb)};  
DBQ=C:\Inetpub\wwwroot\Academy\curriculum.mdb
```

Here's a second example:

```
Driver={SQL Server};Server=Socrates;Database=MedCenter;  
UID=mweib;PWD=clooney7
```

To learn more about connection strings, see “Writing a connection string” on page 97.

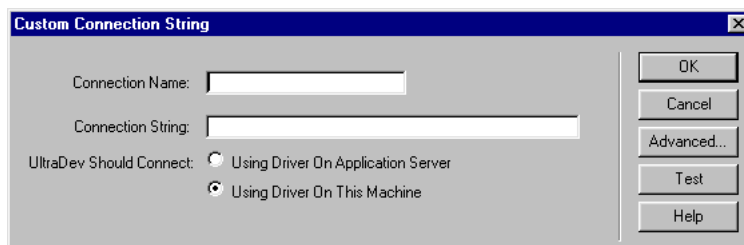
**To create a DSN-less connection:**

- 1 In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

- 2 Click New and select Custom Connection String from the pop-up menu.

The Custom Connection String dialog box appears.



- 3 Enter a name for the new connection.

- 4 Enter a connection string to the database.

If you do not specify an OLE DB provider in the connection string—that is, if you don't include a *Provider* parameter—ASP will automatically use the OLE DB provider for ODBC drivers. In that case, you must specify an appropriate ODBC driver for your database.

If your site is hosted by a commercial Internet service provider (ISP) and you don't know the full path to your database, use the MapPath method of the ASP session object in your connection string. For more information, see “Creating a DSN-less connection with MapPath” on page 96.

- 5 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.

For more information, see “Restricting the amount of information” on page 107.

**Note:** You cannot create a schema or catalog in Microsoft Access.

**6** Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the connection string. If the connection still fails, check the URL prefix for the application server (see “Specifying a URL prefix” on page 27).

**7** Click OK.

Your new connection should now appear in the Connections dialog box.

**8** Click Done to close the Connections dialog box.

## Creating an OLE DB connection

Creating a direct OLE DB connection can improve the speed of your connection by eliminating the ODBC layer between your Web application and the database. If you don't specify an OLE DB provider for your database, ASP uses the default OLE DB provider for ODBC drivers to communicate with an ODBC driver, which in turn communicates with the database. By using a database-specific OLE DB provider, you eliminate the ODBC middleman.

You can obtain OLE DB providers for Microsoft Access and SQL Server in the Microsoft Data Access Components (MDAC) 2.5 package, which you can download from the Microsoft Web site at <http://www.microsoft.com/data/download.htm>.

The Oracle Provider for OLE DB is available with Oracle8i Release 2 for Windows. You can also download the provider from the Oracle Web site at [http://technet.oracle.com/tech/nt/ole\\_db/](http://technet.oracle.com/tech/nt/ole_db/) (registration is required).

In UltraDev, you create an OLE DB connection by including a `Provider` parameter in a connection string. For example, here are parameters for common OLE DB providers for Access, SQL Server, and Oracle databases, respectively:

```
Provider=Microsoft.Jet.OLEDB.4.0;...  
Provider=SQLOLEDB;...  
Provider=OraOLEDB;...
```

For the parameter value of your OLE DB provider, see your provider vendor's documentation, or consult your system administrator.

To learn more about connection strings, see “Writing a connection string” on page 97.

**To create an OLE DB connection:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click New and select Custom Connection String from the pop-up menu.  
The Custom Connection String dialog box appears.
- 3 Enter a name for the new connection.
- 4 Enter a connection string to the database.
- 5 Specify a `Provider` parameter for the connection string.

For example, if you have a SQL Server database and the Microsoft OLE DB driver for SQL Server databases is installed on your server, then you would include the following `Provider` parameter in your connection string:

```
Provider=SQLOLEDB;...
```

In this string, `SQLOLEDB` is the name of the Microsoft OLE DB driver for SQL Server databases.

- 6 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.

For more information, see “Restricting the amount of information” on page 107.

**Note:** You cannot create a schema or catalog in Microsoft Access.

- 7 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the connection string. If the connection still fails, check the URL prefix for the application server (see “Specifying a URL prefix” on page 27).
- 8 Click OK.  
Your new connection should now appear in the Connections dialog box.
- 9 Click Done to close the Connections dialog box.

## Creating a DSN-less connection with MapPath

To write a connection string to a file-based database on a remote server, you must know the full path to that database on the server. This information is not always available, especially if you're working with a commercial Internet service provider (ISP).

To find a file's physical path, use the MapPath method of the ASP session object. The MapPath method translates the logical file path information used by a client browser into a physical path on the server.

**To create a DSN-less connection with the MapPath method:**

- 1 In UltraDev, choose Connections from the Modify menu.

The Connections dialog box appears.

- 2 Click New and select Custom Connection String from the pop-up menu.

The Custom Connection String dialog box appears.

- 3 Enter a name for the new connection.

- 4 Enter a connection string in which the Server.MapPath method supplies the value of the DBQ parameter.

The Server.MapPath method takes one parameter: the file's virtual or relative path. For example, suppose the UltraDev tutorial database is located in a directory called Data in your ISP host directory called MySite. The MapPath method can be expressed as follows in the connection string:

```
...DBQ="Server.MapPath("/MySite/Data/compasstravel.mdb")"
```

For more information on the Server.MapPath method, consult the ASP documentation from Microsoft.

- 5 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the connection string. If the connection still fails, check the URL prefix for the application server (see "Specifying a URL prefix" on page 27).

- 6 Click OK.

Your new connection should now appear in the Connections dialog box.

- 7 Click Done to close the Connections dialog box.



## Writing a connection string

A connection string combines all the information your Web application needs on the server to connect to a database. UltraDev inserts this string in your page's server-side scripts to be processed later by your application server.

A connection string for Microsoft Access and SQL Server databases consists of a combination of the following parameters separated by semicolons:

**Provider** specifies the OLE DB provider for your database. If you don't include this parameter, then the default OLE DB provider for ODBC is used and you must specify an appropriate ODBC driver for your database.

**Driver** specifies the ODBC driver to use if you don't specify an OLE DB provider for your database.

**Server** specifies the server hosting the SQL Server database if your Web application runs on a different server.

**Database** is the name of a SQL Server database.

**DBQ** is the path to a file-based database such as one created in Microsoft Access. The path is the one on the server hosting the database file.

**UID** specifies the user name.

**PWD** specifies the user password.

**DSN** is the data source name, if you use one. Depending on how you define the DSN on your server, you can omit the connection string's other parameters. For example, `DSN=Results` can be a valid connection string if, when you create the DSN, you define the other parameters needed to connect to the database. (For more information, see "Setting Up a DSN in Windows" on page 249.)

Connection strings for other kinds of databases may not use the parameters listed above, or will have different names or uses for the parameters. For more information, see your database vendor's documentation, or consult your system administrator.

Here's an example of a connection string that will create an ODBC connection to an Access database called `trees.mdb`:

```
Driver={Microsoft Access Driver (*.mdb)};  
DBQ=C:\Inetpub\wwwroot\Research\trees.mdb
```

Here's an example of a connection string that will create an OLE DB connection to a SQL Server database system called `Mothra` located on a server called `Gojira`:

```
Provider=SQLOLEDB;Server=Gojira;Database=Mothra;UID=jsmith;  
PWD=orlando8
```

## Creating a database connection for a ColdFusion application

A ColdFusion application can communicate with any ODBC driver or OLE DB provider. ColdFusion applications can also connect to a database using native drivers. The driver or provider in turn communicates with the database.

ColdFusion applications rely on data source names (DSNs) to establish a connection to a database. A DSN is a name representing all the parameters needed to connect to a specific database.

To create a database connection, you must first set up a DSN for the database, then use it in UltraDev.

### Set up a ColdFusion DSN

You can set up a DSN in the ColdFusion Administrator on the server; for instructions, see the ColdFusion documentation or consult your system administrator. You can also set up a DSN in the ODBC Data Source Administrator in Windows; for more information, see “Setting Up a DSN in Windows” on page 249.

The DSN defines the connection parameters. The parameters can include the server name, the path to the database or the database name, the ODBC driver to use, and the user name and password, if any. Once the DSN is defined, you can use it to invoke the underlying parameters.

For example, suppose you have a SQL Server database called Precinct located on a server called Kojak. To gain access to the database, you must enter the user name **columbo** and the password **savalas7**. Using these parameters, you can define a DSN called `ourcops` in ColdFusion Administrator. Then you can create the connection by entering the single word `ourcops` in UltraDev instead of all the other parameters.

### Creating a regular ColdFusion connection in UltraDev

You can create regular ColdFusion connections in UltraDev.

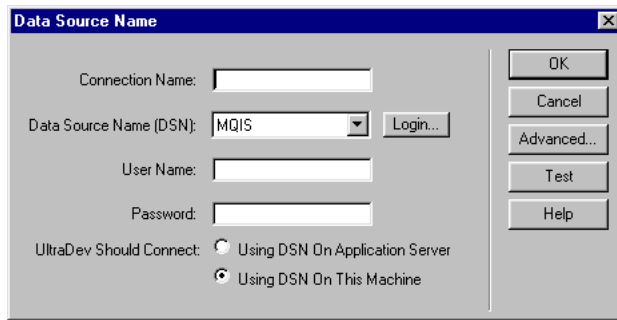
The UltraDev implementation of ColdFusion connectivity does not support stored procedures in databases other than SQL Server 7.0. If you're a Windows user and want to access a stored procedure in a database other than SQL Server 7.0, select the Using DSN On This Machine option and use ODBC to connect to the database at design time.

If you're a Macintosh user and want to use a stored procedure in a database other than SQL Server 7.0, create an advanced ColdFusion connection using JDBC to connect to the database at design time. For procedures, see “Creating an advanced ColdFusion connection (Macintosh users)” on page 100.

**To create a regular ColdFusion connection:**

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click New and choose Data Source Name from the pop-up menu.
- 3 If this is the first connection you create for the site, UltraDev prompts you for your ColdFusion user name and password.

Enter the user name and password you use to log in to the ColdFusion Administrator. After you enter them, UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name dialog box.



If this is not the first connection you create for the site, UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name dialog box. (UltraDev remembers your ColdFusion user name and password.)

- 4 Enter a name for the new connection.
- 5 Select the DSN from the list.
- 6 If required, complete the User Name and Password boxes.
- 7 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.

For more information, see “Restricting the amount of information” on page 107.

**Note:** You cannot create a schema or catalog in Microsoft Access.

- 8 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the DSN, password, and user name. If the connection still fails, check the URL prefix for the application server (see “Specifying a URL prefix” on page 27).

- 9 Click OK.

Your new connection should now appear in the Connections dialog box.

- 10 Click Done to close the Connections dialog box.

### Creating an advanced ColdFusion connection (Macintosh users)

ColdFusion connectivity does not support stored procedures in databases other than SQL Server 7.0. If you're a Macintosh user and want to use a stored procedure in a database other than SQL Server 7.0, you must create an advanced ColdFusion connection.

An advanced connection allows UltraDev (rather than your ColdFusion application) to connect to the database at design time and display information about the stored procedure while you build your pages.

#### To create an advanced ColdFusion database connection in UltraDev:

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click New and choose Data Source Name - Advanced.
- 3 If this is the first connection you create for the site, UltraDev prompts you for your ColdFusion user name and password.  
  
Enter the user name and password you use to log in to the ColdFusion Administrator. After you enter them, UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name - Advanced dialog box.  
  
If this is not the first connection you create for the site, UltraDev connects to the server, retrieves the ColdFusion DSNs, and displays the Data Source Name - Advanced dialog box. (UltraDev remembers your ColdFusion user name and password.)
- 4 Enter a name for the new connection.
- 5 Select the appropriate DSN from the list.

- 6 If required, complete the User Name and Password boxes in the Data Source Name - Advanced dialog box.

User name and password information for each ColdFusion data source is usually kept in the ColdFusion Administrator.

- 7 Select the Using JDBC Driver on This Machine option.

- 8 Define the JDBC connection parameters.

For more information, see “About JDBC connection parameters” on page 102.

- 9 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.

For more information, see “Restricting the amount of information” on page 107.

**Note:** You cannot create a schema or catalog in Microsoft Access.

- 10 Click Test.

UltraDev attempts to connect to the database. If the connection fails, double-check the JDBC connection parameters.

- 11 Click OK.

Your new connection should now appear in the Connections dialog box.

- 12 Click Done to close the Connections dialog box.

## Creating a database connection for a JSP application

A JSP application can communicate with any JDBC driver. The JDBC driver in turn communicates with your database. You can also use an ODBC driver if you have a JDBC-ODBC bridge driver. A JDBC-ODBC bridge driver is software that turns your JDBC-speaking application into an ODBC speaker.

Some common JDBC drivers include the Oracle Thin JDBC driver, Oracle Java Driver, the JDBC Driver for DB2, and the Sun JDBC-ODBC Bridge driver. For more information on JDBC drivers and their vendors, see the searchable database of JDBC drivers on the Sun Web site at <http://industry.java.sun.com/products/jdbc/drivers>.

## Creating a JDBC database connection

Make sure a JDBC driver appropriate for your database is properly installed on the system hosting your database, then proceed as follows.

### To create a JDBC database connection:

- 1 In UltraDev, choose Connections from the Modify menu.  
The Connections dialog box appears.
- 2 Click New and choose your driver from the pop-up menu.  
If your driver is not listed, choose Custom JDBC Connection.  
A connection dialog box appears.
- 3 Enter a name for the connection.
- 4 Enter the driver's connection parameters.  
See the examples in the next section. For driver-specific requirements, see the driver vendor's documentation or consult your system administrator.
- 5 If you want, restrict the number of database items UltraDev retrieves at design time by clicking Advanced and entering a schema or catalog name.  
For more information, see "Restricting the amount of information" on page 107.  
**Note:** You cannot create a schema or catalog in Microsoft Access.
- 6 Click Test.  
UltraDev attempts to connect to the database. If the connection fails, double-check the JDBC connection parameters. If the connection still fails, check the URL prefix for the application server (see "Specifying a URL prefix" on page 27).
- 7 Click OK.  
Your new connection should now appear in the Connections dialog box.
- 8 Click Done to close the Connections dialog box.

## About JDBC connection parameters

JDBC connections usually consist of four parameters: the driver, user name, password, and URL (which specifies the location of the database). Generally, the values of the driver parameter and the URL parameter depend on the driver.

Three common JDBC drivers are the I-net JDBC driver, the Oracle Thin JDBC driver, and the Sun JDBC-ODBC Bridge. Use the following connection parameters for these drivers. For the connection parameters of other drivers, consult the driver vendor's documentation.

**The I-net JDBC driver** supports Microsoft SQL Server databases. If you use this driver to connect to your SQL Server database, enter the following parameter values in UltraDev:

Driver: `com.inet.tds.TdsDriver`

URL: `jdbc:inetdae:server_name:db_port?database=database_name`

Username: `my_username`

Password: `my_password`

The value `server_name` is the IP address or the name assigned to the database server by the system administrator.

For example, if your SQL Server database is called Students and your server has an IP address of 192.176.63.42 and a database port number of 1343, you would enter the following values in UltraDev:

Driver: `com.inet.tds.TdsDriver`

URL: `jdbc:inetdae:192.176.63.42:1343?database=Students`

Username: Anna

Password: lacrosse3

**The Oracle Thin JDBC driver** supports Oracle databases. If you use this driver to connect to your Oracle database, enter the following parameter values in UltraDev:

Driver = `oracle.jdbc.driver.OracleDriver`

URL = `jdbc:oracle:thin:@server_name:db_port:SID`

Username = `my_username`

Password = `my_password`

The value `server_name` is the IP address or the name assigned to the database server by the system administrator. The value `SID` is the database system identifier. If you have more than one Oracle database running on the same system, you use the SID to tell them apart.

For example, if your server is called Aristotle, the database port is 1343, and you defined a database SID called `patients` on that server, you would enter the following parameter values in UltraDev:

Driver: `oracle.jdbc.driver.OracleDriver`

URL: `jdbc:oracle:thin:@Aristotle:1343:patients`

Username: dana

Password: r1ngette

**Sun Jdbc-Odbc Bridge** driver can communicate with ODBC drivers like Microsoft Access Driver. If you use this driver to connect to your database through the intermediary of an ODBC driver, enter the following parameter values in UltraDev:

Driver: `sun.jdbc.odbc.JdbcOdbcDriver`

URL: `jdbc:odbc:my_DSN`

Username: `my_username`

Password: `my_password`

For example, if your DSN is called `CompassTravel` and you don't need a user name or password to access the Microsoft Access database, you would enter the following parameter values in UltraDev:

Driver: `sun.jdbc.odbc.JdbcOdbcDriver`

URL: `jdbc:odbc:CompassTravel`

Username:

Password:

## Editing or deleting database connections

When you create a database connection, UltraDev stores the connection information in a file in the `Connections` subfolder in the site's local root folder. UltraDev does not actually create a database connection for your Web application until you define a recordset for a page in the application (see "Defining a recordset" on page 125). At that point, UltraDev writes code in the file to establish the connection, and inserts an include directive in your page. An include directive is an instruction to the server to include a specified file in the current document at run time. In this case, the server will insert the connection code in your document.

**Note:** If you open a page containing a connection created in UltraDev 1, UltraDev 4 automatically creates an include file for the old connection. The new file is not actually used until you edit the existing recordset or server behavior that uses the old connection, or create a new recordset or server behavior using the new connection.

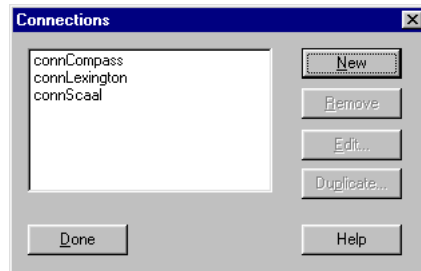
For more information on server-side includes, see "Reusing Content with Templates and Libraries," in *Using Dreamweaver* or in Dreamweaver Help (Help > Using Dreamweaver).



**To update a connection:**

- 1 Choose Connections from the Modify menu.

The Connections dialog box appears:



- 2 Select the connection, then click Edit or Remove.

If you click Edit, make the changes and click OK.

- 3 Click Done to close the Connections dialog box.

UltraDev automatically updates the include file, which in turn automatically updates all the pages in the site that use the connection.

If you deleted the connection, you must update every recordset that uses the old connection by double-clicking the name of the recordset in the Data Bindings panel and choosing a new connection.

## Creating a connection for UltraDev use

By default, the copy of UltraDev running on your development computer (as opposed to the copy of your Web application running on your server) uses remote database connectivity to communicate with databases. UltraDev sends an HTTP request to the Web server much as a browser sends an HTTP request to any Web server. Scripts uploaded to the server by UltraDev read the incoming HTTP request and handle the details of communicating with the database. (The connection scripts on the server use the database connection you created for your Web application.)

Because remote database connectivity eliminates the need to install and configure database drivers on your local system, it's especially well suited for the Macintosh.

Though remote database connectivity greatly simplifies the job of connecting to a database at design time, UltraDev can sometimes connect to the database without it. In certain conditions, at design time UltraDev can use the direct database connection used by your Web application at run time. For more information, see "Using a direct connection at design time" on page 106.

## Setting up remote database connectivity

If you properly configured your system, you are already set up for remote database connectivity. You don't need to do anything else. For more information, see “Configuring your system” on page 22.

After you configure your system, UltraDev takes care of the rest. The program creates a subfolder in your remote site's root folder called `_mmDBScripts` and uploads database connection scripts to it. The connection scripts communicate with the database by using the database connection parameters you specified for your Web application.

UltraDev uploads the connection scripts to your remote site without your intervention using the method you specified for your application server (for example, FTP).

To delete the scripts from the server, choose Site > Remote Connection Scripts.

## Using a direct connection at design time

Instead of using remote database connectivity, UltraDev can use the direct database connection used by your Web application when deployed on the server. Your Web application does not rely on HTTP connectivity; it relies instead on a direct database connection using OLE DB, ODBC, or JDBC connectivity.

UltraDev can use this direct connection only if the connection parameters that work for your Web application on the server also work for UltraDev running on your development computer. If you must change a single parameter to make the connection work for UltraDev, you must use HTTP connectivity.

### To instruct UltraDev to use a direct connection at design time:

- 1 Choose Connections from the Modify menu, select your connection, and click Edit.

The dialog box for your connection appears.

- 2 Verify that the connection's parameters will also work for UltraDev running on your development computer.

For example, will the path (in some cases, URL) to the database work if your starting point is the UltraDev computer, not the Web server? Does the UltraDev computer have the specified driver? If a DSN is specified, can UltraDev use the DSN's parameters? A DSN's parameters typically include the database driver, the location of the database file, and the password and user name, if any. (For more information on DSNs, see “Setting Up a DSN in Windows” on page 249.)

If any one of the parameters will not work for UltraDev running on your development computer, you cannot use your Web application's connection. You must use the default HTTP connection instead.

- 3 If each of the listed parameters works for UltraDev running on your development computer, specify that you want to connect using a DSN or driver on your local machine by clicking the option.
- 4 Click OK.

## Restricting the amount of information

Advanced users of large database systems like Oracle should restrict the number of database items retrieved by UltraDev at design time. An Oracle database may contain items that UltraDev cannot process at design time. In Oracle, create a schema, then use it in UltraDev to filter out unnecessary items at design time.

Other users may benefit from restricting the amount of information UltraDev retrieves at design time. Some databases contain dozens or even hundreds of tables, and you might not want UltraDev to list them all for you at design time. In the Recordset dialog box in UltraDev, for example, clicking the Tables pop-up menu lists all the tables in the specified database. If that database contains dozens of tables, the list will be long and hard to use.

UltraDev also connects and gets all the tables in the database whenever you modify a recordset. If the database has a large number of tables, UltraDev might take a long time to retrieve them all on certain systems.

If your database contains a schema or catalog, you can use it to restrict the number of database items UltraDev gets at design time.

You must first create a schema or catalog in your database application before you can apply it in UltraDev. Consult your database documentation or your system administrator.

**Note:** You cannot create a schema or catalog in Microsoft Access.

### To apply a schema or catalog in UltraDev:

- 1 Choose Modify > Connections.
- 2 Select the connection you want to restrict and click Edit.  
The dialog box appropriate for the connection appears.
- 3 Click Advanced and enter the name of your schema or catalog.
- 4 Click OK.



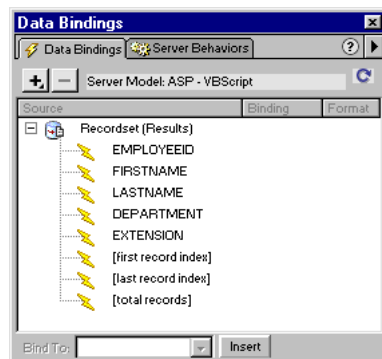
## CHAPTER 4

### Defining UltraDev Data Sources

The first step in building a dynamic page in Dreamweaver UltraDev is defining one or more UltraDev data sources for it. An UltraDev data source is a store of information from which you can select data to include in your Web page.

Your data sources can include not only fields in a recordset, but values submitted by an HTML form, values contained in a server object, values of JavaBean properties, and other data.

Any data source you define is added to your list of data sources in the Data Bindings panel, which you use to add dynamic content to your page.



You can store—or cache—your data sources in a Design Note so you can work on your site even if you don't have access to your database or server. Caching your data sources may also speed up development.

For more information on the data sources discussed in this chapter, consult your server technology's documentation or visit the following Web sites:

- For ASP documentation, visit the Microsoft server technology Web site at <http://msdn.microsoft.com/workshop/server/toc.htm>.
- For ColdFusion documentation, visit the Allaire ColdFusion Web site at <http://www.allaire.com/Documents/cf4docs.cfm>.
- For JSP documentation, visit the Sun JSP Web site at <http://java.sun.com/products/jsp/docs.html>.

## Defining a recordset as a data source

If you decide to use a database with your application, you must define a recordset to temporarily store data from the database.

**Note:** A recordset is called a resultset in JSP. This guide uses *recordset* as a generic term for resultset.

**To define a recordset as a data source:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and select Recordset (Query) from the pop-up menu.
- 3 Define the recordset you selected.

For detailed procedures, see “Defining a recordset” on page 125.

The newly defined recordset appears in the panel's data sources list.

To display the value of a recordset field in your Web page, drag the field from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining browser-submitted data sources for ASP pages

You can define a data source for your ASP page to store or display information that the user's browser submits to your server. In ASP, information submitted by the browser is placed in a request object on the server.

### About the ASP request object

The request object in ASP has five collections: Request.QueryString, Request.Form, Request.ServerVariables, Request.Cookie, and Request.ClientCertificates.

The **QueryString** collection is used to retrieve information appended to the sending page's URL, such as when the page has an HTML form using the GET method. The query string consists of one or more name/value pairs (example, last=Smith, first=John) appended to the URL with a question mark (?). If the query string has more than one name/value pair, they are combined with ampersands (&).

For example, suppose you have a page called survey.asp that contains an HTML form with text fields called "last" and "first", and the form uses the GET method. If John Smith completes the form and clicks the Submit button, the following URL is sent to the server:

```
http://www.somesite.com/survey.asp?last=Smith&first=John
```

On the server, the values of "last" and "first" are stored in the following variables:

```
Request.QueryString("last")  
Request.QueryString("first")
```

The following code snippet in your HTML source code would display the word Smith on a Web page:

```
<% = Request.QueryString("last") %>
```

The **Form** collection is used to retrieve form information included in the body of the HTTP request by an HTML form using the POST method.

For example, suppose you have a page that contains an HTML form with text fields called "last" and "first", and the form uses the POST method. If Jane Doe completes the form and clicks the Submit button, the information she entered is included in the body of the HTTP request sent to the browser.

On the server, the values of "last" and "first" are stored in the following variables:

```
Request.Form("last")  
Request.Form("first")
```

The following code snippet in your HTML source code would display the word Doe on a Web page:

```
<% = Request.Form("last") %>
```

The **ServerVariables** collection is used to retrieve the values of predetermined environment variables. The collection has a long list of variables, including CONTENT\_LENGTH (the length of content submitted in the HTTP request, which you can use to see if the form is empty), and HTTP\_USER\_AGENT (the user's browser).

For example, Request.ServerVariables("HTTP\_USER\_AGENT") contains information about the submitting browser, such as Mozilla/4.07 [en] (WinNT; I), which denotes a Netscape Navigator 4.07 browser.

For a complete list of server environment variables, see the online documentation installed with Microsoft Personal Web Server (PWS) or Internet Information Server (IIS).

The **Cookies** collection is used to retrieve the values of the cookies sent in an HTTP request. For example, suppose your page reads a cookie called “acme” on the user’s system. On the server, the values of the cookie are stored in the variable `Request.Cookies("acme")`.

The **ClientCertificate** collection is used to retrieve the certification fields from the HTTP request sent by the browser. The certification fields are specified in the X.509 standard.

## Defining an ASP request variable as a data source

You can display the value of a request variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

To define a request variable as data source for an ASP page:

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Request Variable from the pop-up menu.
- 3 Choose one of the request collections from the Type pop-up menu.

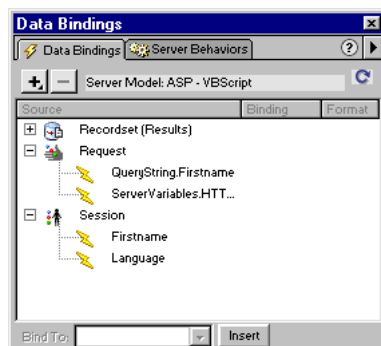
For example, if you want to access the information in the `Request.ServerVariables` collection, choose `ServerVariables`. If you want to access the information in the `Request.Form` collection, choose `Form`.

- 4 Specify the variable in the collection that you want to access.

For example, if you want to access the information in the `Request.ServerVariables("HTTP_USER_AGENT")` variable, enter the argument `HTTP_USER_AGENT`. If you want to access the information in the `Request.Form("lastname")` variable, enter the argument `lastname`.

- 5 Click OK.

The newly defined data source appears in the Data Bindings panel.





To display the value of the request variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining browser-submitted data sources for ColdFusion pages

You can define a data source for your ColdFusion page to store or display information that the user’s browser submits to your server. In ColdFusion, information submitted by the browser is placed in URL, form, and client variables on the server. You can also define ColdFusion cookie, CGI, server, and local variables as data sources.

### About ColdFusion URL, form, and client variables

In ColdFusion, most browser-submitted information is contained in three server variables: URL, form, and client.

**URL variables** are used to retrieve information appended to the sending page’s URL, such as when the page has an HTML form using the GET method. The query string consists of one or more name/value pairs (example, `last=Smith, first=John`) appended to the URL with a question mark (?). If the query string has more than one name/value pair, they are combined with ampersands (&).

For example, suppose you have a page called `survey.asp` that contains an HTML form with text fields called “last” and “first”, and the form uses the GET method. If John Smith completes the form and clicks the Submit button, the following URL is sent to the server:

```
http://www.somesite.com/survey.asp?last=Smith&first=John
```

On the ColdFusion server, the values of “last” and “first” are stored in the following variables:

```
URL.last  
URL.first
```

The following code snippet in your HTML source code would display the word Smith on a Web page:

```
<CFOUTPUT>  
    #URL.last#  
</CFOUTPUT>
```

**Form variables** are used to retrieve form information included in the body of the HTTP request by an HTML form using the POST method.

For example, suppose you have a page that contains an HTML form with text fields called “lastname” and “firstname”, and the form uses the POST method. If Jane Doe completes the form and clicks the Submit button, the information she entered is included in the body of the HTTP request sent to the browser.

On the ColdFusion server, the values of “lastname” and “firstname” are stored in the following variables:

```
Form.lastname  
Form.firstname
```

The following code snippet in your HTML source code would display the word Doe on a Web page:

```
<CFOUTPUT>  
    #Form.lastname#  
</CFOUTPUT>
```

**Client variables** are used to maintain the application’s state as the user moves from page to page in the application, as well as from session to session. Maintaining state means to preserve information from one page (or session) to the next so that the application “remembers” the user and the user’s previous choices and preferences.

ColdFusion has the following system-set client variables: CFID, CFToken, URLToken, HitCount, TimeCreated, and LastVisit. You can also create your own client variables in the source code.

For example, the following code snippet in your HTML source code would format and display the last date the user opened the application:

```
<CFOUTPUT>  
    Date last visited: #DateFormat(Client.LastVisit)#.  
</CFOUTPUT>
```

## Defining ColdFusion form or URL variables as data sources

You can display the value of a form or URL variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a form or URL variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose either URL or Form from the pop-up menu.

The choice depends on the method used by your HTML form. Choose URL if the method is GET; choose Form if the method is POST.

- 3 Enter the name of the variable.

For example, if you want to access the information in the `Form.lastname` ColdFusion variable, enter `lastname`.

- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the ColdFusion variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining ColdFusion client variables as data sources

You can display the value of a client variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a client variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Client Variable from the pop-up menu.
- 3 Enter the name of the variable.

For example, if you want to access the information in the `Client.LastVisit` ColdFusion variable, enter `LastVisit`.

- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the client variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining ColdFusion cookie variables as data sources

You can display the value of a cookie variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a cookie variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Cookie Variable from the pop-up menu.
- 3 Enter the name of the variable.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the cookie variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining ColdFusion CGI variables as data sources

You can display the value of a CGI variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a CGI variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose CGI Variable from the pop-up menu.
- 3 Enter the name of the variable.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the CGI variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining ColdFusion server variables as data sources

You can display the value of a server variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a server variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Server Variable from the pop-up menu.
- 3 Enter the name of the variable.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the server variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining ColdFusion local variables as data sources

You can display the value of a local variable in your Web page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a local variable as a data source for a ColdFusion page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Local Variable from the pop-up menu.
- 3 Enter the name of the variable.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

To display the value of the local variable in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining browser-submitted data sources for JSP

You can define a data source for your JSP page to store or display information that the user’s browser submits to your server. In JSP, information submitted by the browser is placed in a request object on the server.

You can display the value of a request variable in your JSP page by defining it as a data source in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define browser-submitted information as a data source for a JSP page:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose Request Variable from the pop-up menu.
- 3 Enter the name of the variable.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

## Defining session variables as data sources

You can use session variables to store and display information maintained for the duration of a user’s visit (or session). The server creates a different Session object for each user and maintains it for a set period of time or until the object is explicitly terminated.

Because session variables last the whole session and persist when the user moves from page to page in the application, they’re ideal for storing user preferences. You can also use a session variable to keep track of a user’s name and personalize subsequent pages requested by the same user.

You set the value of session variables in your source code. You can then display the values in your pages by defining them as data sources in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a predefined session variable as a data source for your page:**

- 1 Assign a value to a session variable in your source code.

Here is a simple example in ASP:

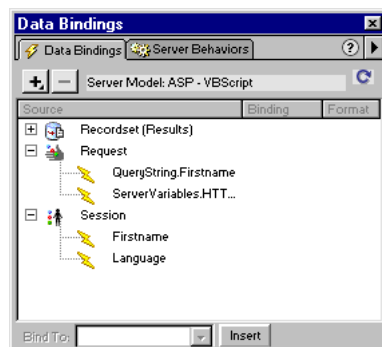
```
<%  
Session("Firstname") = "Elmer"  
%>
```

- 2 Open the Data Bindings panel by choosing Window > Data Bindings.
- 3 Click the Plus (+) button and choose Session Variable from the pop-up menu.
- 4 Enter the name of the variable you defined in the source code.

For example, "Firstname".

- 5 Click OK.

The newly defined data source appears in the Data Bindings panel.



## Defining application variables as data sources

In ASP and ColdFusion, you can use application variables to store and display information that is maintained for the lifetime of the application and persists from user to user. The application's lifetime lasts from the time the first user requests a page in the application to the time the Web server is stopped. (An application is defined as all the files in a virtual directory and its subdirectories.)

**Note:** There is no application object in JSP.

Because application variables last for the application's lifetime and persist from user to user, they're ideal for storing information that must exist for all users, such as a page counter.

You set the value of application variables in your source code. You can then display their values in your pages by defining them as data sources in UltraDev, then dragging the data source from the Data Bindings panel to the page.

**To define a predefined application variable as a data source for your page:**

- 1 From an ASP or ColdFusion page, open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and select Application Variable from the pop-up menu.
- 3 Enter the name of the variable you defined in the source code.
- 4 Click OK.

The newly defined data source appears in the Data Bindings panel.

## Defining a stored procedure server object as a data source

You can create a stored procedure server object and define it as a data source for your page. A stored procedure object consists of SQL statements that can perform one or more database operations. The stored procedure object typically returns recordsets, but it can also return other data such as output parameters. It can also add or delete records, or even create new tables in the database.

You create the object by selecting a pre-existing stored procedure in a database. A stored procedure consists of one or more SQL statements saved in a database (as opposed to saved in the source code of your dynamic page).

The stored procedure server object is called a command in ASP, a callable in JSP, and a stored procedure in ColdFusion.

You can also use a stored procedure to define a recordset data source (as opposed to a server object data source). See "Invoking a stored procedure" on page 131.

**To define a stored procedure server object as a data source for your page:**

- 1 Open any dynamic page in UltraDev.
- 2 In the Data Bindings panel, click the Plus (+) button and select one of the following items from the pop-up menu:
  - In ASP, select Command (Stored Procedure).
  - In JSP, select Callable (Stored Procedure).
  - In ColdFusion, select Stored Procedure.
- 3 Enter a name for the stored procedure, then select a connection from the Connections pop-up menu to specify the database containing the stored procedure.
- 4 If you're using ASP, select Stored Procedure in the Type pop-up menu.
- 5 Click the Return Recordset option and enter a name for the recordset to be returned.
- 6 Select a stored procedure that returns a recordset from the tree of database items at the bottom of the dialog box.
- 7 Enter any required parameters in the Variables table.

You don't need to enter any parameters for the `RETURN_VALUE` variable.
- 8 Click OK.

The stored procedure object is added to the list of data sources in the Data Bindings panel.

To display the value of a recordset field in your Web page, drag it from the Data Bindings panel to the page. For more information, see “Adding Dynamic Content” on page 133.

## Defining JavaBeans as data sources (JSP only)

JavaBeans are common architectural elements of multitier JSP applications. JavaBeans are typically used as part of a middle “business-logic” layer meant to buffer the presentation logic from the data-access logic. In these applications, the beans, not the JSP pages, contain the logic that directly accesses the database.

In UltraDev, JavaBeans are treated as data sources. They appear in the Data Bindings panel. You can double-click the bean in the panel to see its properties. You can drag the bean's properties to the page to create dynamic data references.

You can also define a JavaBeans collection as an Ultradev data source. A JavaBeans collection is simply a set of beans.

**Note:** In UltraDev, only repeated regions and dynamic bindings are supported for collections.



Copies of the bean class (or of the .zip or .jar file containing the bean class) must reside in the following locations:

- On the system running UltraDev, a copy of the bean class must reside in the UltraDev Configuration\classes folder or in the system's classpath. (UltraDev uses this copy of the class at design-time.)
- On the system running the JSP application server, the bean class must reside in the application server's classpath. (Your application server uses this copy of the class at run-time.) The application server's classpath varies from application server to application server, but generally the classpath is to a WEB-INF folder with a classes/bean subfolder.

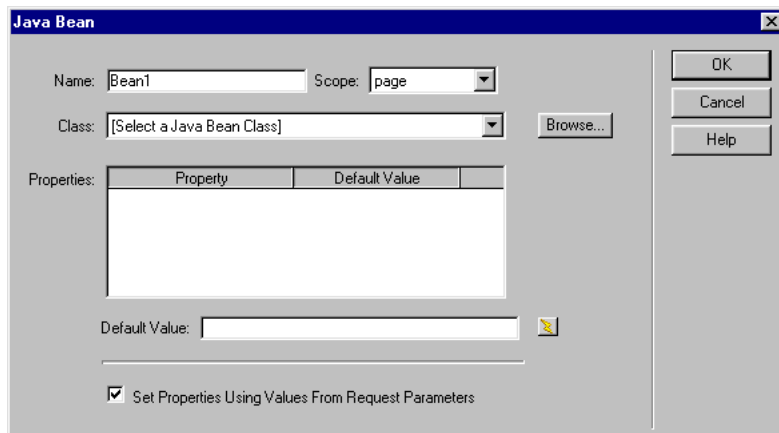
If UltraDev and the application server both run on the same system, and the application server uses the system classpath (not an internal classpath), a single copy of the bean class can reside on the computer in the system classpath. Both the application server and UltraDev will use this copy of the class. Otherwise, copies of the bean class must reside in two paths on the computer, as outlined above.

The folder structure must match the bean's package. For example, if your bean's package is called com.lenny.myBean, then the package must be stored in \com\lenny\ in the classpath or in the UltraDev Configuration\classes folder.

#### To define a bean as a data source:

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Click the Plus (+) button and choose JavaBean from the pop-up menu.

The JavaBean dialog box appears.



- 3 Enter the bean's name.

4 Choose the bean's scope.

5 Choose the bean's class.

To list the classes in a .zip or .jar file, click Browse and select the file.

The class is expressed in the following format:

```
packagename.classname
```

6 If you want to give one of the bean's properties a default value, select the property from the list and enter a value in the Default Value box below the list.

You can also set the property's default value to a dynamic value by clicking the lightning bolt icon beside the Default Value box.

7 Click OK.

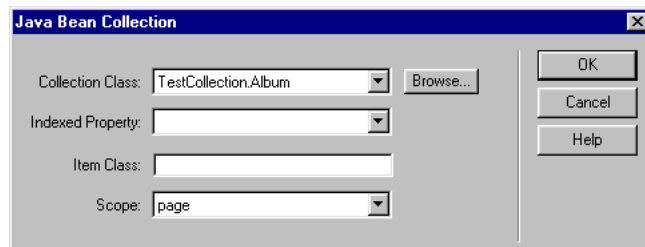
The newly defined data source appears in the Data Bindings panel.

**To define a JavaBeans collection as a data source:**

1 Open the Data Bindings panel by choosing Window > Data Bindings.

2 Click the Plus (+) button and choose JavaBean Collection from the pop-up menu.

The JavaBean Collection dialog box appears.



3 Choose the collection's class.

To list the classes in a .zip or .jar file, click Browse and select the file.

The class is expressed in the following format:

```
packagename.classname
```

4 Choose one of the collection's indexed properties.

UltraDev displays a default name in the Item Class box. If the name is wrong, enter the correct one.

5 Choose the bean's scope.

6 Click OK.

The newly defined data source appears in the Data Bindings panel.

## Caching data sources

You can store your data sources in a Design Note so you can keep working on your site even if you don't have access to your database or server. Caching may also speed up development.

To cache your data sources, click the arrow button in the top right corner of the Data Bindings panel and toggle Cache in the pop-up menu.

If you make changes to one of your data sources, you should refresh the cache by clicking the Refresh button (the circle-arrow icon) in the top right corner of the Data Bindings panel. (Expand the panel if you don't see the button.)

## Changing or deleting data sources

You can change or delete any UltraDev data source—that is, any data source listed in the Data Bindings panel.

Changing or deleting a data source in the Data Bindings panel does *not* change or delete any instance of that data source on the page. It merely changes or deletes it as a possible data source for the page.

**Note:** To edit an instance of the data source on the page, double-click the data source name in the Server Behaviors panel, make your changes in the dialog box that appears, and click OK. To delete an instance of a data source on the page, select the data source in the Server Behaviors panel and click the Minus (-) button. For more information, see “Changing dynamic content” on page 144 and “Deleting dynamic content” on page 145.

### To change a data source in the list of data sources available to your page:

- 1 In the Data Bindings panel, double-click the name of the data source you want to edit.
- 2 Make your changes in dialog box that appears.
- 3 If satisfied with your work, click OK.

### To delete a data source from the list of data sources available to your page:

- 1 In the Data Bindings panel, select the data source from the list.
- 2 Click the Minus (-) button.



## CHAPTER 5

### Creating a Recordset

---

If you decide to use a database with your application, you can't work with the database directly: you must work through the intermediary of a recordset. For example, when you bind page attributes to data, you bind them to the data in the recordset, not in the database.

A recordset is a subset of records extracted from a database by a database query. A query consists of search criteria that determine what's included in the recordset and what's not. A query can produce a recordset that includes only certain columns, only certain records, or a combination of both.

A recordset can also include all the records and columns of a database table. However, because your applications will rarely need to use every piece of data in a database, strive to make your recordsets as small as possible. A server temporarily holds the recordset in memory, then discards it when it's no longer needed. Therefore, smaller recordsets use less memory than larger ones, and the server's performance may improve as a result. The basic guideline when defining recordsets is to include only the data your application needs.

### Defining a recordset

A recordset is defined by a query, which is a statement composed of search criteria designed to find and extract information from a database. Dreamweaver UltraDev uses the Structured Query Language to build queries. You don't need to know SQL (pronounced "sequel") to define a simple recordset in UltraDev. However, if you do know SQL, you can use it to define your recordsets.

**Note:** After you define a recordset, the data it contains won't be immediately visible in the Document window or the Live Data window. Data only becomes visible after you add a column from the recordset to your page and switch to the Live Data window (View > Live Data).

## Defining a recordset without using SQL

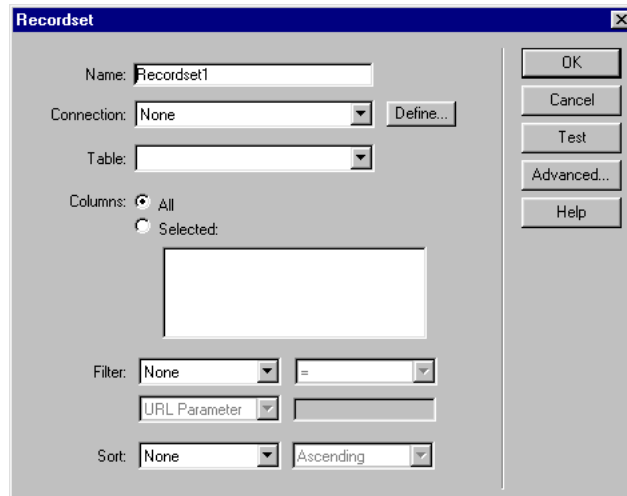
If you're unfamiliar with SQL, you can define recordsets using the UltraDev simple Recordset dialog box. Defining a recordset using this method can be as easy as selecting a connection and a database table from the pop-up menus.

If you want to work with SQL, use the advanced Recordset dialog box. See “Defining a recordset using SQL” on page 128.

### To define a recordset without using SQL:

- 1 Make sure the page that will use the recordset is open in the Document window or the Live Data window.
- 2 In the Data Bindings panel (Window > Data Bindings), click the Plus (+) button and choose Recordset (Query) from the pop-up menu.

The simple Recordset dialog box appears.



If the advanced Recordset dialog appears instead, switch to the simple Recordset dialog box by clicking the Simple button.

- 3 In the Name box, enter a name for the recordset.

A common practice is to add the prefix *rs* to recordset names to distinguish them from other object names in your code—for example, *rsPressReleases*.

**Note:** Do not use spaces or special characters in recordset names.

- 4 Select a connection from the Connection pop-up menu.

If no connection appears in the list, click Define to create one. For more information, see “Connecting to a Database” on page 89.

- 5 In the Table pop-up menu, select the database table that will provide data to your recordset or receive data from it.

The pop-up menu displays all the tables in the connected database.

- 6 To include only some of the table’s columns in the recordset, click Selected and choose the desired columns by Control-clicking (Windows) or Command-clicking (Macintosh) them in the list.

- 7 To include only some of the table’s records, complete the Filter section as follows:

- From the first pop-up menu, select a column in the database table to compare against a test value you define.
- From the second pop-up menu, select a conditional expression to compare the selected value in each record against the test value.
- From the third pop-up menu, select Entered Value.
- In the fourth box, enter the test value.

If the specified value in a record meets your filtering condition, the record will be included in the recordset.

- 8 If you want the records to be sorted, select a column to sort by, then specify whether the records should be sorted in ascending order (1, 2, 3... or A, B, C...) or descending order.

- 9 If you want, click Test to connect to the database and create an instance of the recordset.

A table appears displaying the data in your recordset. Each row contains a record and each column represents a field in that record. Click OK to close the recordset.

- 10 If satisfied with your work, click OK.

UltraDev adds the recordset to your list of available data sources in the Data Bindings panel. Expand the recordset branch to view the columns you defined for it. You can use any of these columns as a source of dynamic content for your page. For more information, see “Adding Dynamic Content” on page 133.

## Defining a recordset using SQL

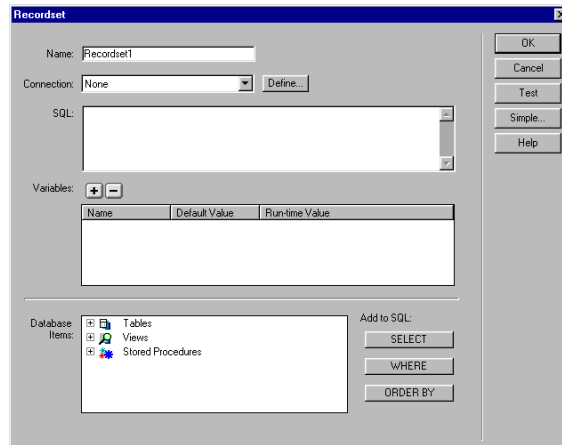
If you're familiar with SQL or if you want to learn it, you can define your recordsets using the UltraDev advanced Recordset dialog box. (For help on writing SQL statements, see, "SQL Primer" on page 251.)

### To define a recordset using SQL:

- 1 Make sure the page that will use the recordset is open in the Document window or the Live Data window.
- 2 In the Data Bindings panel (Window > Data Bindings), click the Plus (+) button and select Recordset (Query) from the pop-up menu.

If the simple Recordset dialog box appears, switch to the advanced Recordset dialog box by clicking the Advanced button.

The advanced Recordset dialog box appears.



- 3 In the Name box, enter a name for the recordset.

A common practice is to add the prefix *rs* to recordset names to distinguish them from other object names in your code—for example, *rsPressReleases*.

**Note:** Do not use spaces or special characters in recordset names.

- 4 Select a connection from the Connection pop-up menu.

If no connection appears in the list, click Define to create one. For more information, see "Connecting to a Database" on page 89.



**5** Enter the SQL statement in the SQL text area.

To reduce the amount of typing, you can use the tree of database items at the bottom of the dialog box. To use the tree, first make sure the SQL text area is blank. Next, expand the branches in the tree until you find the database object you need—a column in a table, for example. Select it and add it to your SQL statement by clicking one of the three buttons on the right side of the tree: Select, Where, and Order By. Each of these buttons adds a clause to the SQL statement.

**6** If you entered variables in the SQL statement, define their values in the Variables area by clicking the Plus (+) button and entering the variable's name, default value (the value the variable should take if no run-time value is returned), and run-time value (usually a server object holding a value sent by a browser, such as an ASP request object).

For example, suppose an HTML form on the requesting page has a field called "Name". The run-time value for this field in ASP could be `Request("Name")`, `Request.Form("Name")`, or `Request.QueryString("Name")`, depending on the form method used (GET or POST). The run-time value for ColdFusion would be `#Name#`. The run-time value for JSP would be `request.getParameter("Name")`.

**7** If you want, click Test to connect to the database and create an instance of the recordset.

If successful, a table appears displaying the data in your recordset. Each row contains a record and each column represents a field in that record. Click OK to clear the recordset.

**8** If satisfied with your work, click OK.

UltraDev adds the recordset to your list of available data sources in the Data Bindings panel. Expand the recordset branch to view the columns you defined for it. You can use any of these columns as a source of dynamic content for your page. For more information, see "Adding Dynamic Content" on page 133.

## Sample SQL statements

Here are two sample SQL statements and the steps for creating them in the advanced Recordset dialog box.

**Note:** To open the advanced Recordset dialog box, click the Plus (+) button in the Data Bindings panel (Window > Data Bindings) and select Recordset (Query) from the pop-up menu. If the simple Recordset dialog box appears, click Advanced.

**To create the following SwwQL statement:**

```
SELECT * FROM Employees
```

- 1 In the tree of database items at the bottom of the dialog box, expand the Tables branch and select the Employees table.
- 2 Click the Select button.
- 3 Click OK to add the recordset to the Data Bindings panel.

**To create the following SQL statement:**

```
SELECT emplNo, emplName  
FROM Employees  
WHERE emplJob = 'varJob'  
ORDER BY emplName
```

- 1 In the tree of database items, expand the Tables branch, then expand the Employees branch.
- 2 Build the SQL statement as follows:
  - Select emplNo and click the Select button.
  - Select emplName and click the Select button.
  - Select emplJob and click the Where button.
  - Select emplName and click the Order By button.
- 3 Place the insertion point after WHERE emplJob in the SQL text area and type ='varJob' (include the equal sign).
- 4 Define the variable 'varJob' by clicking the Plus (+) button in the Variables text area and entering the following values in the Name, Default Value, and Run-Time Value columns: varJob, CLERK, Request("job").
- 5 Click OK to add the recordset to the Data Bindings panel.

## Invoking a stored procedure

A recordset can be defined by a stored procedure, which consists of one or more SQL statements residing in a database (as opposed to residing in the source code of your dynamic pages). Stored procedures can return one or more recordsets, though UltraDev only supports stored procedures that return one recordset or no recordsets.

Stored procedures can also be placed in the server's memory and used as a data source for your page. For more information, see “Defining a stored procedure server object as a data source” on page 119.

### To invoke a stored procedure to define a recordset:

- 1 Open the page that needs the recordset.
- 2 In the Data Bindings panel, click the Plus (+) button and select Recordset (Query) from the pop-up menu.  
  
If the simple Recordset dialog box opens, click the Advanced button to open the advanced Recordset dialog box.
- 3 In the advanced Recordset dialog box, enter a name for your recordset and select the connection to the database containing the stored procedure.
- 4 In the tree of database items at the bottom of the dialog box, expand the Stored Procedures branch, select the stored procedure you want, and click the Procedure button.
- 5 If the stored procedure takes parameters, define their default and run-time values in the Variables area.
- 6 Click OK.

## Copying a recordset to another page

You can copy a recordset from one page to another in your site.

### To copy a recordset to another page:

- 1 Select the recordset in either the Data Bindings panel or the Server Behaviors panel.
- 2 Click the arrow button in the top right corner of the panel and choose Copy from the pop-up menu.
- 3 Open the other page.
- 4 Click the arrow button in the top right corner of the Data Bindings panel or the Server Behaviors panel, and choose Paste from the pop-up menu.

## Editing or deleting a recordset as a data source

You can edit or delete any recordset in the list of data sources available to your page—that is, any recordset listed in the Data Bindings panel.

For example, suppose you decide to display the cellular phone numbers of your sales staff on a results page. If your existing recordset doesn't have a column listing the cellular numbers, you must change its definition to include one.

Editing or deleting a recordset in the Data Bindings panel does *not* edit or delete any instance of that recordset on the page. It merely edits or deletes it as a possible data source for the page. To edit or delete an instance of the recordset on the page, see “Changing dynamic content” on page 144 and “Deleting dynamic content” on page 145.

### To edit a recordset in the Data Bindings panel:

- 1 In the Data Bindings panel, double-click the name of the recordset you want to edit.
- 2 Make your changes in the simple or advanced Recordset dialog box.  
For more information, see “Defining a recordset” on page 125.
- 3 Click the Test button to view the contents of the updated recordset, then click OK to close the test recordset.
- 4 If satisfied with your work, click OK.

### To delete a recordset in the Data Bindings panel:

- 1 In either the Data Bindings panel or the Server Behaviors panel, select the recordset you want to delete.
- 2 Click the Minus (-) button.

## CHAPTER 6

### Adding Dynamic Content

.....

After defining one or more data sources for your page, you can use the data sources to add dynamic content to the page. Data sources can include a column in a recordset, a value submitted by an HTML form, the value contained in a server object, and other data. For more information, see “Defining UltraDev Data Sources” on page 109.

In Dreamweaver UltraDev, you can place dynamic content almost anywhere in the page or its HTML source code:

- You can place it at the insertion point.
- You can replace a text string with it.
- You can insert it in an HTML attribute. For example, dynamic content can define the `src` attribute of an image, or the `value` attribute of a form field.

You add dynamic content by choosing one of your data sources in the Data Bindings panel. UltraDev inserts a server-side script in the page’s source code instructing the server to transfer the data from the data source to the page’s HTML source code.

There is often more than one way to make a given page element dynamic. For example, to make an image dynamic you can use the Data Bindings panel, the Property inspector, or the Image command in the Insert menu. This chapter describes the most efficient ways of making various page elements dynamic.

By default, an HTML page can display only one record at a time. To display the other records in the recordset, you can add a link to move through the records one at a time (see “Creating recordset navigation links” on page 147), or you can create a repeated region to display more than one record on a single page (see “Displaying multiple records” on page 151).

After adding dynamic content to a page, you can make changes to it. For more information, see “Changing dynamic content” on page 144 and “Deleting dynamic content” on page 145.

## Making text dynamic

You can replace existing text with dynamic text, or you can place dynamic text at a given insertion point on the page.

Dynamic text adopts any text formatting applied to the existing text or to the insertion point. For example, if a CSS style affects the selected text, the dynamic content replacing it is also affected by the style. You can add or change the text format of dynamic content by using any of the Dreamweaver text formatting tools.

You can also apply a data format to dynamic text. For example, if your data consists of dates, you can specify a particular date format such as 04/17/00 for U.S. visitors, or 17/04/00 for Canadian visitors.

## Adding dynamic text

You can replace regular text on your page with dynamic text, or you can add dynamic text at the insertion point on the page.

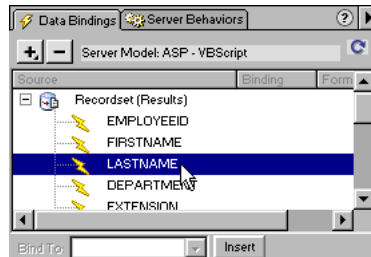
**To add dynamic text:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Make sure the Data Bindings panel lists the data source you want to use.

The data source should contain plain text (ASCII text). For example, the data source could contain full-fledged HTML, which is plain text. If no data sources appear in the list, or if the available data sources don't meet your needs, click the plus (+) button to define a new data source. See “Defining UltraDev Data Sources” on page 109.

- 3 In Design view, select text on the page, or click where you want to add dynamic text.

- 4 In the Data Bindings panel, select a data source from the list. If you select a recordset, specify the column you want in the recordset.



- 5 Click Insert, or drag the data source onto the page.

The dynamic content appears on the page if you're working in the Live Data window. In the Document window, a placeholder appears instead. (If you selected text on the page, the placeholder replaces the text selection.)

The placeholder for a recordset data source uses the syntax

`{RecordsetName.ColumnName}`, where `Recordset` is the name of the recordset and `ColumnName` is the name of the column you chose from the recordset.

## Modifying the appearance of placeholders

Sometimes, the length of the placeholders for dynamic text distorts the page's layout in the Document window. You can solve the problem by using empty curly braces as placeholders.

**To use empty curly braces as placeholders for dynamic text:**

- 1 Choose Edit > Preferences > Invisible Elements.
- 2 In the "Show Dynamic Text As" pop-up menu, choose {}.
- 3 Click OK.

## Applying a data format

If you want, specify a data format for the dynamic text. For example, if the price data in a record reads 10.989, you can display the price on your page as \$10.99 by selecting the Currency - 2 Decimal Places format in the pop-up menu. This format takes a number and displays it with two decimal places. If the number has more than two decimal places, the data format rounds the number to the closest decimal; if the number has no decimal places, the data format adds a decimal point and two zeros.

**To apply a data format to dynamic text:**

- 1 Select the dynamic content (Live Data window) or its placeholder (Document window) on the page.
- 2 In the Data Bindings panel (Window > Data Bindings), click the arrow button in the Format column.
- 3 Select a format from the pop-up menu.

Make sure the data format is appropriate for the data. For example, the Currency formats work only if the dynamic text consists of numbers. Also, you cannot apply more than one format to the same data.

To edit existing data formats or create one of your own, see “Editing and creating data formats” on page 211.

## Making images dynamic

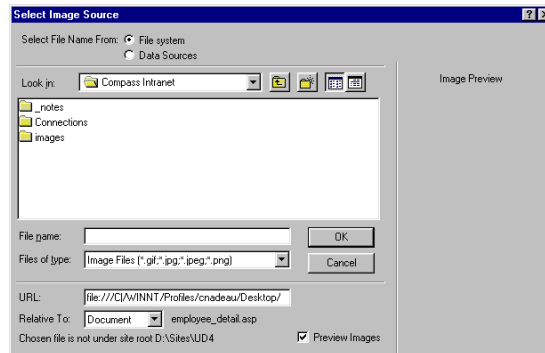
You can make images on your page dynamic. For example, suppose you design a page to display items for sale at a charity auction. Each page would include descriptive text and a photo of one item. The page’s general layout would remain the same for each item, but the photo (and descriptive text) could change.



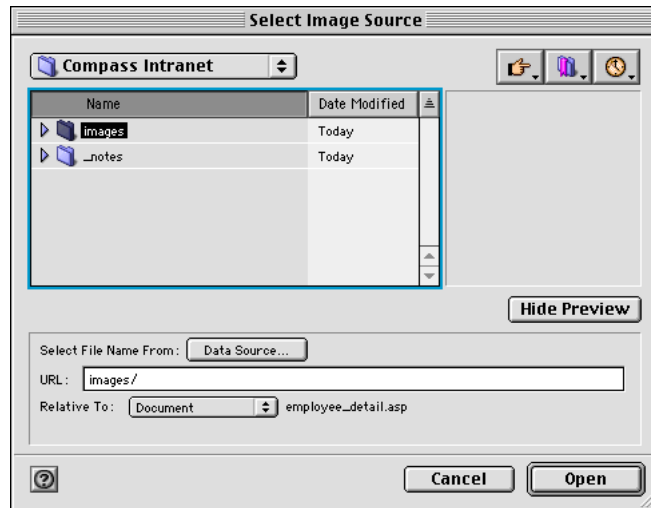
### To make an image dynamic:

- 1 With the page open in the window's Design view (View > Design), place the insertion point where you want the image to appear on the page, then select Insert > Image.

The Select Image Source dialog box appears.



On the Macintosh, the dialog box is different:



- 2 Click the Data Sources option (Windows) or the Data Source button (Macintosh).

A list of data sources appears.

- 3 Select a data source from the list.

The data source should be a recordset containing the paths to your image files. Depending on the file structure of your site, the paths can be absolute, document relative, or root relative. For more information, see About document locations and paths in “Linking and Navigating,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

**Note:** UltraDev does not currently support binary images stored in a database.

If no recordsets appear in the list, or if the available recordsets don't meet your needs, define a new recordset. For instructions, see “Defining a recordset” on page 125.

- 4 Click OK.

## Making form objects dynamic

You can create a dynamic HTML form to display records in your database. For example, you could design a form to display suppliers' contact information.

You can only display one record at a time in a form. To give users the ability to view other records, you can add links to move through the records one at a time. (See “Creating recordset navigation links” on page 147.)

The most common dynamic form objects are text fields, image fields, checkboxes, and radio buttons. You can also use a data source to populate the options in a list/menu object.

## Making text and image fields dynamic

You can make text and image fields on a form dynamic.

**To make text fields dynamic:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Make sure the Data Bindings panel lists the data source you want to use.

The data source should contain textual information. If no data sources appear in the list, or if the available data sources don't meet your needs, click the Plus (+) button to define a new data source. For instructions, see “Defining UltraDev Data Sources” on page 109.
- 3 In Design view, select a text field in your HTML form.
- 4 In the Data Bindings panel, select a data source from your list of data sources.
- 5 In the Bind To box, make sure the `value` attribute (`input.value`) is selected.
- 6 Click Bind.

### To make image fields dynamic:

- 1 Place the insertion point where you want the image field to appear on the page, then select Insert > Image.

The Select Image Source dialog box appears.

- 2 Click the Data Sources option (Windows) or the Data Source button (Macintosh).

A list of data sources appears.

- 3 Select a data source from the list.

The data source should be a recordset containing the paths to your image files. Depending on the file structure of your site, the paths can be absolute, document relative, or root relative. For more information, see About document locations and paths in “Linking and Navigating,” of the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

**Note:** UltraDev does not currently support binary images stored in a database.

If no recordsets appear in the list, or if the available recordsets don't meet your needs, define a new recordset. For instructions, see “Defining a recordset” on page 125.

- 4 Click OK.

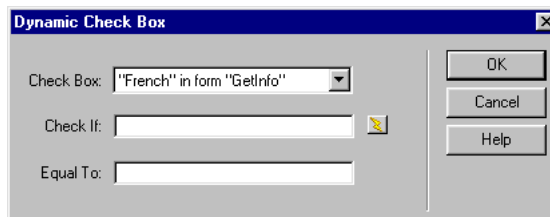
## Making checkboxes dynamic

You can make checkboxes on a form dynamic.

### To make checkboxes dynamic:

- 1 Select a checkbox in the HTML form on your page.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Dynamic Elements > Dynamic Check Box from the pop-up menu.

The Dynamic Check Box dialog box appears.



**3** If you want the checkbox to be selected when a field in a record equals a certain value, do the following:

- Click the lightning bolt icon beside the Check If box and select the field from the list of data sources.

Typically, the chosen field contains Boolean data such as *Yes* and *No*, or *true* and *false*.

- In the Equal To box, enter the value the field must have for the checkbox to appear selected.

For example, if you want the checkbox to appear selected when a specific field in a record equals *Yes*, enter *Yes* in the Equal To box.

**Note:** This value will also be returned to the server if the user clicks the form's Submit button.

**4** Click OK.

The checkbox will appear selected or unselected—depending on the data—when the form is viewed in a browser.

## Making radio buttons dynamic

You can make radio buttons on a form dynamic.

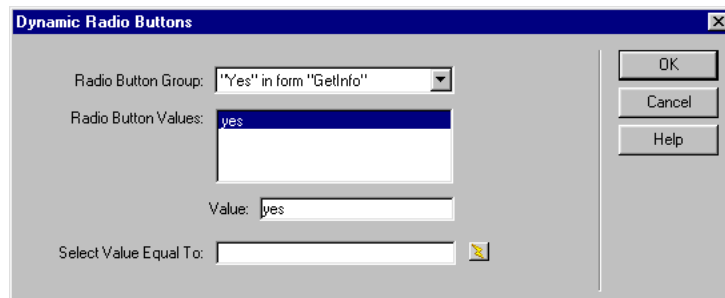
**To make radio buttons dynamic:**

- 1** Make sure the page has at least one group of radio buttons.

You create a group of radio buttons by giving all the radio buttons in a group the same name. For more information, see “Creating Forms,” in the *Using Dreamweaver* book or in Dreamweaver Help (Help > Using Dreamweaver).

- 2** In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Dynamic Elements > Dynamic Radio Button from the pop-up menu.

The Dynamic Radio Buttons dialog box appears.



- 3 In the Radio Button Group pop-up menu, select a group of radio buttons on your page.
- 4 You can specify the value of each radio button in the group by selecting a radio button in the Radio Button Values list, then entering a value for the radio button in the Value box.

**Note:** The value of the currently selected radio button will be returned to the server if the user clicks the form's Submit button.

- 5 If you want a particular radio button to be selected when the page opens in a browser or when a record is displayed in the form, enter a value equal to the radio button's value in the Select Value Equal To box.

You can enter a static value or you can specify a dynamic one by clicking the lightning bolt icon beside the box and selecting a dynamic value from the list of data sources. In either case, the value you specify should match one of the radio button values.

- 6 Click OK.

## Making list/menu objects dynamic

You can make a list/menu object on your form dynamic.

**To make a list/menu object dynamic:**

- 1 Select the list/menu object in the HTML form on your page.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select Dynamic Elements > Dynamic List/Menu from the pop-up menu.

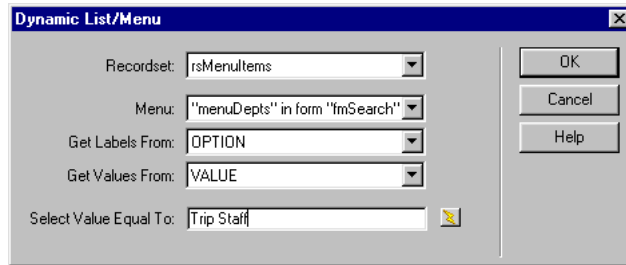
The Dynamic List/Menu dialog box appears.

- 3 In the Recordset pop-up menu, select the recordset containing the menu information.
- 4 In the Get Labels From pop-up menu, select the field containing the labels for the menu items.
- 5 In the Get Values From pop-up menu, select the field containing the values of the menu items.
- 6 If you want a particular menu item to be selected when the page opens in a browser or when a record is displayed in the form, enter a value equal to the menu item's value in the Select Value Equal To box.

You can enter a static value or you can specify a dynamic one by clicking the lightning bolt icon beside the box and selecting a dynamic value from the list of data sources. In either case, the value you specify should match one of the menu item values.

7 Click OK.

Here's an example of a completed Dynamic List/Menu dialog box:



## Making HTML attributes dynamic

You can dynamically change the appearance of your page by binding HTML attributes to data. For example, you could change the background image of a table by binding the table's `background` attribute to a field in a recordset.

You can bind HTML attributes with the Data Bindings panel or with the Property inspector.

**To make HTML attributes dynamic with the Data Bindings panel:**

- 1 Open the Data Bindings panel by choosing Window > Data Bindings.
- 2 Make sure the Data Bindings panel lists the data source you want to use.

The data source should contain data that's appropriate for the HTML attribute you want to bind. If no data sources appear in the list, or if the available data sources don't meet your needs, click the plus (+) button to define a new data source. For instructions, see "Defining UltraDev Data Sources" on page 109.
- 3 In Design view, select an HTML object.
- 4 In the Data Bindings panel, select a data source from your list.
- 5 In the Bind To box, select an HTML attribute from the pop-up menu.
- 6 Click Bind.

**To make HTML attributes dynamic with the Property inspector:**

- 1 In Design view, select an HTML object and open the Property inspector (Window > Properties).
- 2 If the attribute you want to bind has a folder icon next to it in the inspector's Standard view, click the folder icon to open a file selection dialog box, then click the Data Sources option to display a list of data sources. Skip to step 6.
- 3 If the attribute you want to bind does not have a folder icon next to it in the Standard view, click the List tab (the lower of the two tabs) on the left side of the inspector.

The Property inspector's List view appears.



- 4 If the attribute you want to bind is not listed in the List view, click the Plus (+) button, then enter the attribute's name or click the small arrow button and select the attribute from the pop-up menu.
- 5 To make the attribute's value dynamic, click the attribute then click the lightning-bolt icon or folder icon at the end of the attribute's row.

If you clicked the lightning-bolt icon, a list of data sources appears.

If you clicked the folder icon, a file selection dialog box appears. Click the Data Sources option to display a list of data sources.

- 6 Select a data source from the list of data sources.

The data source should hold data that's appropriate for the HTML attribute you want to bind. If no data sources appear in the list, or if the available data sources don't meet your needs, define a new data source. For instructions, see "Defining UltraDev Data Sources" on page 109.

- 7 Click OK.

## Making ActiveX, Flash, and other object parameters dynamic

You can make the parameters of Java applets and plug-ins dynamic, as well as the parameters of ActiveX, Flash, Shockwave, Director, and Generator objects.

Before starting, make sure the fields in your recordset hold data that's appropriate for the object parameters you want to bind.

### To make object parameters dynamic:

- 1 In Design view, select an object on the page and open the Property inspector (Window > Properties).
- 2 Click the Parameters button.  
The Parameters dialog box appears.
- 3 If your parameter does not appear in the list, click the Plus (+) button and enter a parameter name in the Parameter column.
- 4 Click the parameter's Value column, then click the lightning-bolt icon to specify a dynamic value.

A list of data sources appears.

- 5 Select a data source from the list.

The data source should hold data that's appropriate for the object parameter you want to bind. If no data sources appear in the list, or if the available data sources don't meet your needs, define a new data source. For instructions, see "Defining UltraDev Data Sources" on page 109.

- 6 Click OK.

## Changing dynamic content

You can change the dynamic content on your page by editing the server behavior that provides the content. For example, you can edit a recordset server behavior to provide more records to your page.

Dynamic content on a page is listed in the Server Behaviors panel. For example, if you add a recordset to your page, the Server Behaviors panel lists it as follows:

```
Recordset(myRecordset)
```

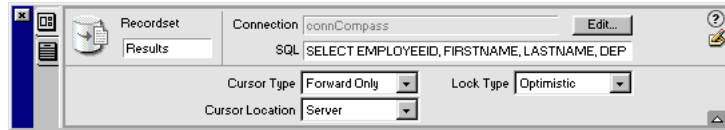
If you add another recordset to your page, the Server Behaviors panel lists both recordsets as follows:

```
Recordset(mySecondRecordset)  
Recordset(myRecordset)
```



To edit a server behavior providing dynamic content, double-click the server behavior in the Server Behaviors panel. The same dialog box you used to define the original data source appears. Make your changes in the dialog box and click OK.

You can also use the Property inspector to edit the recordsets on your page. Open the Property inspector (Window > Properties), then select the recordset in the Server Behaviors panel (Window > Server Behavior). Here's the Property inspector for a recordset:



If you edit a recordset in the Live Data window with the Auto Refresh option not selected, you must refresh the page to see your changes. To refresh the page, click the Refresh button or choose View > Refresh Live Data.

## Deleting dynamic content

After adding dynamic content to a page, you can delete it by selecting the dynamic content on the page and pressing Delete. You can also delete it by selecting the dynamic content in the Server Behaviors panel and clicking the Minus (-) button.

**Note:** This operation removes the server-side script in your page that retrieves the dynamic content from the database. It does not delete the data in the database.



## CHAPTER 7

### Displaying Database Records

---

If you add recordset data to your page, the page by default displays only one record (the first one in the recordset). To display the other records, you can add a link to move through the records one at a time, or create a repeated region to display more than one record on a single page. In a repeated region, you can add a link to each record to open a detail page that gives users more information. You can also enhance the page's usability by adding record counters and hidden regions.

UltraDev provides you with a number of live objects to add advanced page components such as record navigation links, record counters, and master/detail pages. Advanced users can also create these features from scratch using server behaviors.

### Creating recordset navigation links

You can add a set of recordset navigation links to your page to let users move from one record to the next, or from one set of records to the next. For example, after designing a page to display five records at a time, you might want to add links such as "Next Records" or "Previous Records" to let users see the next or previous five records.

Dreamweaver UltraDev lets you create four types of navigation links to move through a recordset: First, Previous, Next, Last. A single page can contain any number of these links, provided they all work on a single recordset. In other words, you can't add links to move through a second recordset on the same page.

Recordset navigation links require the following building blocks:

- A recordset to navigate
- Dynamic content on the page to display the record or records
- Text or images on the page to serve as a clickable navigation bar
- A “Move To Record” set of server behaviors to navigate the recordset

You can add the last two building blocks in a single operation using the Record Navigation Bar live object, or you can add them separately using the UltraDev design tools and the Server Behaviors panel.

## Creating a recordset to navigate

You can define the recordset yourself, or you can let the user define the recordset by running a search. To define a recordset yourself, see “Defining a recordset” on page 125. To let the user define the recordset by running a search, see “Building Pages That Search Databases” on page 165.

## Creating a page display

Bind some or all of the recordset’s columns to text, HTML attributes, or form objects on your page. For information, see “Adding Dynamic Content” on page 133.

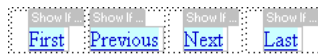
You can also display multiple records using a repeat region. For more information, see “Displaying multiple records” on page 151.

## Creating the recordset navigation bar in one operation

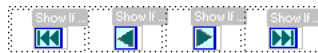
You can create a recordset navigation bar in a single operation using the Recordset Navigation Bar live object. The live object adds the following building blocks to the page:

- An HTML table with either text or image links
- A set of “Move to” server behaviors
- A set of “Show Region” server behaviors

The text version of the bar created by the live object looks like this:



The image version looks like this:



Before placing the navigation bar on the page, make sure the page contains a recordset to navigate (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).

After placing the navigation bar on the page, you can use the UltraDev design tools to customize the bar to your liking. You can also edit the “Move to” and “Show Region” server behaviors by double-clicking them in the Server Behaviors panel.

If you want to build the navigation bar block by block using the UltraDev design tools and the Server Behaviors panel, see “Creating the recordset navigation bar block by block” on page 150.

**To create the recordset navigation bar with the live object:**

- 1 In Design view, place the insertion point at the location on the page where you want the navigation bar to appear.
- 2 Choose Insert > Live Objects > Recordset Navigation Bar.

The Insert Recordset Navigation Bar dialog box appears.



- 3 Choose the recordset to navigate.
- 4 Select text or image links.

In the image version of the bar, UltraDev uses its own image files. If you want, you can replace them with your own image files after placing the bar on the page.

- 5 Click OK.

UltraDev builds a table that contains text or image links that move through the selected recordset when clicked. When the first record in the recordset is displayed, the “First” and “Previous” links or images are hidden. When the last record in the recordset is displayed, the “Next” and “Last” links or images are hidden.

The layout of the navigation bar is fully customizable using the Dreamweaver design tools.

## Creating the recordset navigation bar block by block

You can build the navigation bar block by block using the UltraDev page-design tools and the Server Behaviors panel. You use individual server behaviors to create navigation links to move to the first record, the last record, the next record (or set of records), and the previous record (or set of records) in a recordset.

You can also create a complete recordset navigation bar in a single operation using the Recordset Navigation Bar live object (see “Creating the recordset navigation bar in one operation” on page 148).

When creating a navigation bar from scratch, begin by creating its visual representation using the UltraDev page-design tools. You don’t have to create a link for the text string or image: UltraDev will create one for you.

A simple recordset navigation bar looks like this:



Before creating the navigation bar on the page, make sure the page contains a recordset to navigate (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).

Next, you apply server behaviors to create the navigation links.

### To create record navigation links with server behaviors:

- 1 In Design view, select the text string or image on the page to act as your record navigation link.
- 2 Open the Server Behaviors panel (Window > Server Behaviors) and click the Plus (+) button.
- 3 Choose Move to Record from the pop-up menu, then choose one of the listed server behaviors.

**Note:** If the recordset contains a large number of records, the Move to Last Record server behavior can take a long time to run when the user clicks the link.

- 4 In the Recordset pop-up menu, select the recordset containing the records.
- 5 Click OK.

## Showing and hiding regions

UltraDev has a set of server behaviors that lets you show or hide a region such as a record navigation link depending on whether or not the region is needed. For example, after adding “Previous records” and “Next records” links to a results page, you can specify that the “Previous records” link be shown on all pages of results except the first, and that the “Next records” link be shown on all pages except the last. You can even specify that the results list be shown only if the query returns a recordset that is not empty. If the query returns an empty recordset, the results list is hidden.

Before showing or hiding navigation links on the page, make sure the page contains a recordset to navigate (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).

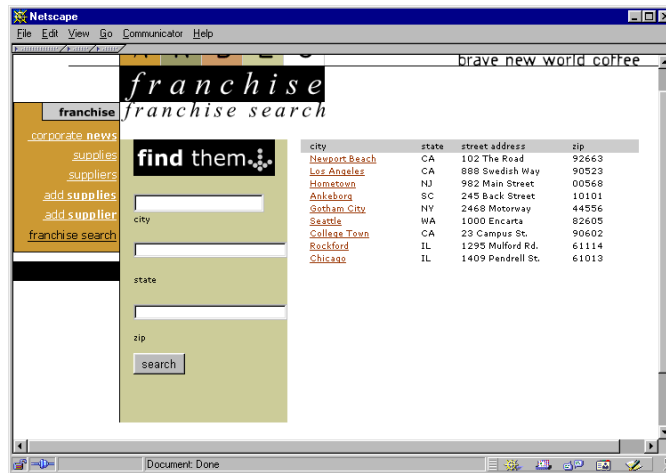
**To show a region only when it’s needed:**

- 1 In Design view, select the region on the page to show and hide.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button.
- 3 Choose Show Region from the pop-up menu, then choose one of the listed server behaviors.
- 4 Click OK.

## Displaying multiple records

To display more than one record on a single page, you apply a Repeat Region server behavior to a selection containing dynamic content. Any selection can be turned into a repeated region. The most common “regions” are a table, a table row, or a series of rows.

For example, you could design a table to display all the franchises of a company. Each row in the table would display a different franchise, and each column would display a different piece of information about the franchises:



You build this kind of table by applying the Repeat Region server behavior to a table row containing dynamic content. At design time, the repeated region consists of a single row. At run time, the row is repeated a number of times and a different record is inserted in each new row.

Before creating a repeated region on the page, make sure the page contains a recordset (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).

#### To create a repeated region:

- 1 In Design view, select a region that contains dynamic content.

The selection can be anything, including a table, a table row, or even a paragraph of text.

To select a region on the page precisely, you can use the tag selector on the status bar. For example, if the region is a table row, click inside the row on the page, then click the right-most `<tr>` icon on the status bar to select the table row.



- 2 Open the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button, and select Repeat Region.

The Repeat Region dialog box appears.



- 3 Specify the recordset containing the data to display in the repeated region.
- 4 Specify the number of records to display per page.

If you specify a limited number of records per page and it's possible the number of records requested will exceed it, add record navigation links to let users display the other records. See “Creating recordset navigation links” on page 147.

- 5 Click OK.

In the Document window, a thin, tabbed gray outline appears around the repeated region. In the Live Data window (View > Live Data), the gray outline disappears and the selection expands to display the number of records you specified.

## Building a record counter

You can use UltraDev to build a record counter such as “Displaying Records 1 - 8 of 31”. Record counters are especially useful in results pages that may display many records.

A record counter requires the following building blocks:

- A recordset to track
- Dynamic content on a page to display the records
- Recordset navigation links to move through the recordset
- A text string on the page to serve as record counter
- A set of dynamic recordset statistics to keep track of the records displayed

For information on the first three building blocks, see “Creating recordset navigation links” on page 147.

You can add the last two building blocks in a single operation using the Record Navigation Status live object, or you can add them separately using the UltraDev page-design tools and the Data Bindings panel.

## Building a record counter in one operation

You can create a record counter in a single operation using the Recordset Navigation Status live object. The live object adds the following building blocks to the page:

- A text string
- A set of dynamic recordset statistics to keep track of the records displayed

The default record counter created by the live object looks like this:

Records (Employees\_first) to (Employees\_last) of (Employees\_total)

In the Live Data window, the counters looks like this:

Records 1 to 1 of 22

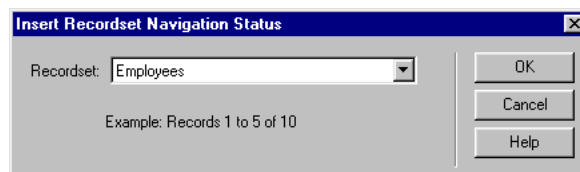
After placing the record counter on the page, you can use the UltraDev page-design tools to customize it to your liking.

You can also build the record counter block by block using the UltraDev page-design tools and the Data Bindings panel (see “Building a record counter block by block” on page 155).

### To create the record counter with the live object:

- 1 Make sure the page contains a recordset (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).
- 2 In Design view, place the insertion point at the location on the page where you want the record counter to appear.
- 3 Choose Insert > Live Objects > Recordset Navigation Status.

The Insert Recordset Navigation Status dialog box appears.



- 4 Choose the recordset to track.

## 5 Click OK.

UltraDev builds and adds the record counter to the page.

The record counter is fully customizable using the Dreamweaver page-design tools.

## Building a record counter block by block

You can build the record counter block by block using the UltraDev Data Bindings panel.

You can also create a complete recordset navigation bar in a single operation using the Recordset Navigation Status live object (see “Building a record counter in one operation” on page 154).

The following procedure describes how to build a typical counter with the Data Bindings panel. You can use the same technique to build different counters.

### To build a record counter with the Data Bindings panel:

- 1 Make sure the page contains a recordset (see “Creating a recordset to navigate” on page 148) and a page layout to display the records (see “Creating a page display” on page 148).

- 2 In Design view, type the counter’s text on the page as follows:

Showing records - of

The text can be anything you choose.

- 3 Place the insertion point at the end of the text string.

- 4 Open the Data Bindings panel (Window > Data Bindings), expand the branch of the recordset to be monitored, select [total records] from the list of data sources, and click Insert. You can also drag [total records] onto the page.

Here’s how the record counter should look in the Document window:

Showing records - of {myRecordset\_total}

- 5 Place the insertion point after the word *records*.

- 6 In the Data Bindings panel, select [first record index] from the list of data sources and click Insert. You can also drag [first record index] onto the page.

Here’s how the counter should look:

Showing records {myRecordset\_first} - of {myRecordset\_total}

- 7 Place the insertion point after the hyphen.

- 8 In the Data Bindings panel, select [last record index] from the list of data sources and click Insert. You can also drag [last record index] onto the page.

Here's how the counter should look:

```
Showing records {myRecordset_first} - {myRecordset_last} of {myRecordset_total}
```

If you view the page in the Live Data window (View > Live Data), the counter should read something like this:

```
Showing records 1 - 5 of 16
```

If the results page has a navigation link to move to the next records, clicking the link would display the next five records in the recordset and the counter would read as follows:

```
Showing records 6 - 10 of 16
```

**Note:** Links don't work in the Live Data window. To test them, you can use the UltraDev Preview in Browser feature. Make sure the Preview Using Live Data Server option is selected in Preferences (Edit > Preferences > Preview in Browser), then select File > Preview in Browser.

## Creating a master/detail page set

A master page is a page that lists records and corresponding links for each record. When the user clicks a link, a detail page opens displaying more information about the record. For example, here's a master page from the intranet site of a fictional company:

CLICK TO VIEW EMPLOYEE PROFILE

Discovering the world

employees 1-5 of 5

Name	Department	Ext.	
BATES, Chris	Operations	3476	VIEW
DAVIS, Welan	Operations	3459	VIEW
GRANDEL, David	Operations	3458	VIEW
RIELY, Dan	Operations	6799	VIEW
SMITH, Ken	Operations	3479	VIEW

ADD NEW EMPLOYEE

When a user clicks one of the linked View icons, a detail page opens:

The screenshot shows a web page titled "EMPLOYEE DIRECTORY DETAIL PAGE" with a logo "Discovering the world". The page displays details for "Bates, Chris".

Bates, Chris	
<b>PHOTO I.D.</b>	<b>ALT. TEL</b> 415-555-6783
	<b>START DATE</b> 10/5/98
	<b>DEPARTMENT</b> Operations
	<b>EXTENSION</b> 3476
	<b>E-MAIL</b> chrisb@compasstravel.com
<b>NOTES</b>	
<div></div>	

A results page is a common type of master page. However, unlike the master page described in this section, the list of records on a results page is determined not by you, the designer, but by the user. (The user determines the list by conducting a database search.) For more information on this type of master page, see “Building Pages That Search Databases” on page 165.

A detail page can be used to update or delete the record displayed. For more information on update and delete pages, see “Building a page to update records” on page 187 and “Building a page to delete a record” on page 193.

A master page consists of the following building blocks:

- A recordset
- A page layout to display multiple records
- A Go to Detail Page server behavior to open the detail page and pass the ID of the record the user clicked

A detail page consists of the following building blocks:

- A page layout to display a single record
- A recordset to hold the record’s details
- Either a recordset filter to retrieve a specific record from the database table, or a Move To Specific Record server behavior to move to a specific record in the recordset.

## Defining a recordset for the master page

Start building a master/detail page set by creating the master page and defining a recordset for the page. The recordset on the master page can be defined by you at design time (see “Defining a recordset” on page 125) or by the user at run time (see “Building Pages That Search Databases” on page 165).

Typically, the recordset on the master page extracts a few columns from a database table while the recordset on the detail page extracts more columns from the same table to provide the extra detail.

## Completing the master/detail page set in one operation

After adding a recordset to a blank master page, you can complete the master/detail page set in a single operation using the Master/Detail Page Set live object. The live object adds the following building blocks to the master page:

- A basic table with a repeated region to display multiple records
- A recordset navigation bar
- A record counter
- A Go to Detail Page server behavior to open the detail page and pass the ID of the record the user clicked

The live object also creates a detail page if you didn’t already create one and adds the following building blocks to it:

- A basic table to display a single record
- A filtered recordset to find and display the record the user clicked on the master page

After the live object places the building blocks on the pages, you can use the UltraDev page-design tools to customize the pages’ layout or the Server Behaviors panel to edit the server behaviors (see “Editing server behaviors on a page” on page 164).

### To complete the master/detail page set with a live object:

- 1 Create a blank master page and add a recordset to it.

Make sure the recordset contains not only all the columns you’ll need for the master page, but also all the columns you’ll need for the detail page.

For instructions, see “Defining a recordset for the master page” on page 158.

- 2 Open the master page in Design view, and choose Insert > Live Objects > Master Detail Page Set.

The Insert Master-Detail Page Set dialog box appears.

Insert Master-Detail Page Set

Recordset: Employees

Master Page Fields: + -

EMPLOYEEID  
FIRSTNAME  
LASTNAME  
PHONE  
STARTDATE  
TITLE

Link To Detail From: EMPLOYEEID

Pass Unique Key: EMPLOYEEID

Show: ☒ 10 Records at a Time  
☐ All Records

Detail Page Name: Browse...

Detail Page Fields: + -

EMPLOYEEID  
FIRSTNAME  
LASTNAME  
PHONE  
STARTDATE  
TITLE

OK  
Cancel  
Help

- 3 In the Recordset pop-up menu, make sure the recordset containing the records you want to display on the master page is chosen.
- 4 In the Master Page Fields area, select the recordset columns to display on the master page.  

By default, UltraDev selects all the columns in the recordset. If your recordset contains a unique key column such as `recordID`, select it and click the Minus (-) button so that it is not displayed on your page.
- 5 If you want to change the order in which the columns appear on the master page, select a column in the list and click the up or down arrow.

On the master page, the recordset columns will be arranged horizontally in a table. Clicking the up arrow moves the column to the left; clicking the down arrow moves the column to the right.

- 6 In the Link To Detail From pop-up menu, choose the column in the recordset that will display a value that also serves as a link to the detail page.

For example, if you want each product name on your master page to have a link to the detail page, choose the recordset column containing product names.

- 7 In the Pass Unique Key pop-up menu, choose the column in the recordset containing values identifying the records.

Usually, the column chosen is the record ID number. This value is passed to the detail page so that it can identify the record chosen by the user.

- 8 Specify the number of records to display on the master page.

- 9 In the Detail Page Name box, click Browse and locate the detail page file you created, or enter a name and let the live object create one for you.

- 10 In the Detail Page Fields area, select the columns to be displayed on the detail page.

By default, UltraDev selects all the columns in the master page's recordset. If the recordset contains a unique key column such as `recordID`, select it and click the Minus (-) button so that it is not displayed on the detail page.

- 11 If you want to change the order in which the columns appear on the detail page, select a column in the list and click the up or down arrow.

On the detail page, the recordset columns will be arranged vertically in a table. Clicking the up arrow moves the column up; clicking the down arrow moves the column down.

- 12 Click OK.

The live object creates a detail page (if you didn't already create one) and adds dynamic content and server behaviors to both the master and detail pages.

- 13 Customize the master and detail pages to suit your needs.

You can fully customize the layout of each page using the UltraDev page-design tools. You can also edit the server behaviors by double-clicking them in the Server Behaviors panel.



## Completing the master/detail page set block by block

This section describes how to build a set of master/detail pages without using the Master/Detail Page Set live object. For instructions on using the live object, see “Completing the master/detail page set in one operation” on page 158.

This section assumes you already created a blank master page and defined a recordset for the master page. To complete the master/detail page set, you need to accomplish the following tasks:

- You must display the records on the master page.
- You must pass the ID of the record the user selected to the detail page.
- You must define a recordset for the detail page to hold the detail data, then bind the recordset columns to the page
- You can either define a recordset filter that retrieves a specific record from the database table, or you can add a Move To Specific Record server behavior to move to a specific record in the recordset.

Using a recordset filter is more efficient than using the server behavior because the filtered recordset will only contain one record.

### To display the records on the master page:

- 1 Create a page layout to display multiple records and bind recordset columns to the page.

A common approach is to create a two-row HTML table on the master page and to drag a limited number of recordset columns from the Data Bindings panel (Window > Data Bindings) into the table's second row. (Use the first row to display the table's column headings.)

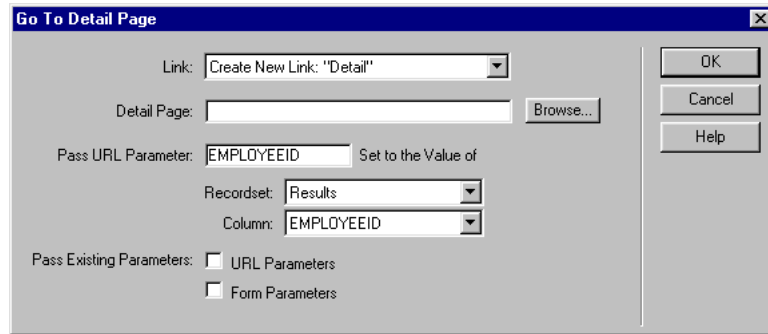
- 2 Create a repeated region to display more than one record at a time.

The repeated region is normally applied to the table row containing the dynamic content. For instructions, see “Displaying multiple records” on page 151.

To open the detail page and pass it a URL parameter that contains the ID of the record the user clicked:

- 1 In the repeated region on the master page, select the dynamic content to double as a link.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button, and choose Go to Detail Page from the pop-up menu.

The Go to Detail Page dialog box appears.



- 3 In the Detail Page box, click Browse and locate the detail page file you created.
- 4 Specify the information you want to pass to the detail page by selecting a recordset and a column from the Recordset and Column pop-up menus.

Typically the information is unique to the record, such as the record's unique key ID.

- 5 Click OK.

The master page passes the value to the detail page in a URL parameter, which is simply a variable appended to the end of the URL used to open the detail page. For example, if the URL parameter is called `id` and the detail page is called `customerdetail.asp`, then the URL will look something like the following when the user clicks on the link:

`http://www.mysite.com/customerdetail.asp?id=43`

The first part of the URL, `http://www.mysite.com/customerdetail.asp`, opens the detail page. The second part, `?id=43`, is the URL parameter. It tells the detail page what record to find and display. The term `id` is the name of the URL parameter and 43 is its value. In this example, the URL parameter contains the record's ID number, 43.

**To complete the detail page:**

- 1 Switch to the detail page.
- 2 In the Data Bindings panel, click the Plus (+) button and choose Recordset (Query) from the pop-up menu.  
The simple Recordset dialog box appears. If the advanced Recordset dialog box appears instead, click Simple to switch.
- 3 Give a name to the recordset, then choose a connection and database table that will provide data to your recordset.
- 4 In the Column area, select the table columns to include in the recordset.

The recordset can be identical or different from the recordset on the master page. Usually a detail page recordset has more columns to display more detail.

If the recordsets are different, the recordset on the detail page should contain at least one column in common with the master page. The common column is usually the record ID column, but it can also be the join field of related tables.

To include only some of the table's columns in the recordset, click Selected and choose the desired columns by Control-clicking (Windows) or Command-clicking (Macintosh) them in the list.

- 5 If you plan to define a recordset filter to find and display the record specified in the master page, leave the Recordset dialog box open and define the filter.

For instructions, see “Building the detail page using a filtered recordset” on page 175.

**Note:** The section on filtered recordsets refers to a results page. A results page is a type of master page.

- 6 If you plan to use a Move To Specific Record server behavior to find and display the record specified in the master page, click OK to close the Recordset dialog box, then add the server behavior.

For instructions, see “Building the detail page using a server behavior” on page 174.

**Note:** The section on the Move To Specific Record server behavior refers to a results page. A results page is a type of master page.

## Editing server behaviors on a page

You can delete or change the properties of any server behavior you add to a page. For example, you can make a repeated region on a page display more records.

### **To change the properties of a server behavior on a page:**

Double-click the server behavior in the Server Behaviors panel, change the properties in the dialog box, and click OK.

### **To delete a server behavior on a page:**

Select the server behavior in the Server Behaviors panel and click the Minus (-) button.

## CHAPTER 8

### Building Pages That Search Databases

---

You can use Dreamweaver UltraDev to build a set of pages to let users search your database. You need at least two pages to add this feature to your Web application. The first page is a page containing an HTML form in which users enter search parameters. Though the page doesn't do any actual searching, it is referred to as "the search page."

The second page you need is the results page, the workhorse of the page set. The results page performs the following tasks:

- Reads the search parameters submitted by the search page
- Connects to the database and searches for records
- Builds a recordset with the records found
- Displays the contents of the recordset

Optionally, you can add a detail page. A detail page gives users more information about a particular record on the results page.

If you have only one search parameter, UltraDev lets you add search capabilities to your Web application without worrying about SQL queries and variables. Simply design your pages and complete a few dialog boxes. If you have more than one search parameter, you need to write a SQL statement and define multiple variables for it.

## Creating the search page

A search page on the Web typically contains form fields in which the user enters search parameters. When the user clicks the form's Search button, the search parameters are sent to a results page on the server. The results page on the server, not the search page on the browser, is responsible for retrieving records from the database.

To get started on this part of your Web application, create two pages: a search page that lets users enter the search parameters, and a results page to display the records found. You can even combine the two pages into one. At minimum, your search page must have an HTML form with a Submit button.

### To add an HTML form to a search page:

- 1 Open the search page and select Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form's boundaries, which are represented by thin red lines.

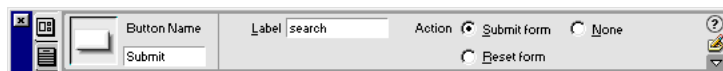
- 2 Add form objects for users to enter their search parameters by choosing Form Objects from the Insert menu.

Form objects include text fields, list menus, checkboxes, and radio buttons. You can add as many form objects as you want to help users refine their searches. However, keep in mind that the greater the number of search parameters on the search page, the more complex your SQL statement will be.

For more information on form objects, see "Creating Forms," in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

- 3 Add a Submit button to the form (Insert > Form Objects > Button).
- 4 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "search":



Next, you'll tell the form where to send the search parameters when the user clicks the Submit button.

- 5 Select the form by selecting the `<form>` tag in the tag selector at the bottom of the Document window, as shown:



- 6 In the Action box on the form's Property inspector, enter the file name of the results page that will conduct the database search.
- 7 In the Method pop-up menu, choose one of the following methods to determine how the form sends data to the server:
- `GET` sends the form data by appending it to the URL as a query string. Because URLs are limited to 8192 characters, don't use the `GET` method with long forms.
  - `POST` sends the form data in the body of a message.
  - `Default` uses the browser's default method (usually `GET`).

The search page is done. Next comes the results page.

## Building the results page

Once the user enters the search parameters, your application must retrieve the records from the database. This work is done by the results page.

Here's the job of a results page:

- Get the search parameters from the search page.
- Connect to the database and search for records.
- Build a recordset with the records found.
- Display the contents of the recordset.

If your search page has only one search parameter (a single text field, for example), you can build the results page without SQL queries and variables. You simply create a basic recordset, then add a filter to it to exclude records that don't meet the search parameters sent by the search page. For instructions, see "Searching with only one search parameter" on page 168.

If your search page has more than one search parameter, you need to write a SQL statement and define multiple variables for it. For instructions, see "Searching with multiple search parameters" on page 170.

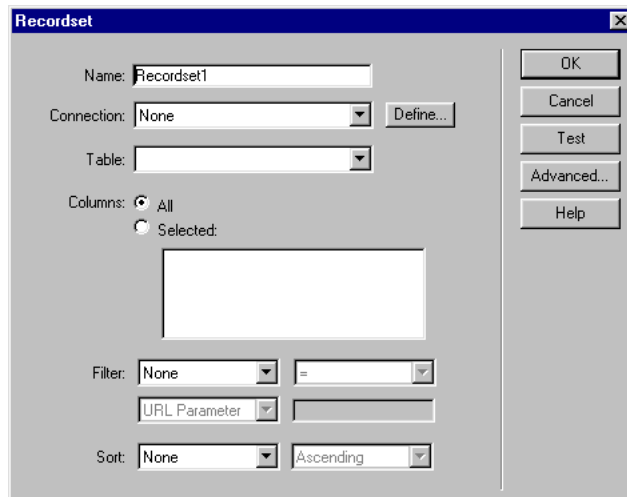
## Searching with only one search parameter

If the search page submits a single search parameter to the server, then you can build the results page without SQL queries and variables. You create a basic recordset with a filter that excludes records that don't meet the search parameter submitted by the search page.

**Note:** If you have more than one search condition, you must use the advanced Recordset dialog box to define your recordset. The simple Recordset dialog box only supports one search condition. For more information, see “Searching with multiple search parameters” on page 170.

**To create the recordset to hold the search results:**

- 1 Open your results page in UltraDev, then create a new recordset by opening the Data Bindings panel (Window > Data Bindings), clicking the Plus (+) button, and selecting Recordset (Query) from the pop-up menu.
- 2 Make sure the simple Recordset dialog box appears.



If the advanced Recordset dialog box appears instead, switch to the simple Recordset dialog box by clicking the Simple button.

- 3 Enter a name for the recordset and choose a connection.

The connection should be to a database containing data you want the user to search.

- 4 In the Table pop-up menu, select the table to be searched in the database.

**Note:** In a single-parameter search, you can search for records in only a single table. To search more than one table at a time, you must use the advanced Recordset dialog box and define a SQL query.



- 5 To include only some of the table's columns in the recordset, click **Selected** and choose the desired columns by **Control-clicking** (Windows) or **Command-clicking** (Macintosh) them in the list.

Pick columns containing information you want to display on the results page.

Leave the Recordset dialog box open for now. You'll use it next to fetch the parameters sent by the search page and create a recordset filter to exclude records that don't meet the parameters.

#### To create the recordset filter:

- 1 From the first pop-up menu in the Filter area, select a column in the table to compare against the search parameter sent by the search page.

For example, if the value sent by the search page is a city name, select the column in your table that contains city names.

- 2 From the pop-up menu beside the first menu, select the equal sign (it should be the default).

This choice states that the user wants only those records in which the selected table column is exactly the same as the one specified on the search page.

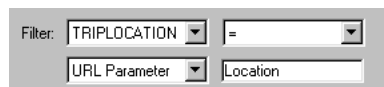
- 3 From the third pop-up menu, select **Form Variable** if the form on your search page uses the **POST** method, or **URL Parameter** if it uses the **GET** method.

This pop-up menu specifies where the value sent by the search page is currently stored on the server. In ASP, if the search form uses the **POST** method, the value is stored in the `Request.Form` collection. If the search form uses the **GET** method, the value is stored in the `Request.QueryString` collection.

- 4 In the fourth box, enter the name of the form object that accepts the search parameter on the search page.

You can get the name by switching to the search page, clicking the form object on the form to select it, and checking the object's name in the Property inspector.

For example, suppose you want to create a recordset that includes only adventure trips to a specific country. Assume you have a column in the table called `TRIPLOCATION`. Also assume the HTML form on your search page uses the **GET** method and contains a **Menu/List** object called `Location` that displays a list of countries. Here's how your Filter section should look:



Filter:

- 5 If you want, click Test, enter a test value, and click OK to connect to the database and create an instance of the recordset.

The test value simulates the value that would otherwise have been returned from the search page. Click OK to close the recordset.

- 6 If you're satisfied with the recordset, click OK.

UltraDev inserts a server-side script on your page that, when run on the server, checks each record in the database table. If the specified field in a record meets the filtering condition, the record is included in a recordset. The script in effect builds a recordset containing only the search results.

The next step is to display the recordset on the results page. For more information, see “Displaying the records” on page 172.

## Searching with multiple search parameters

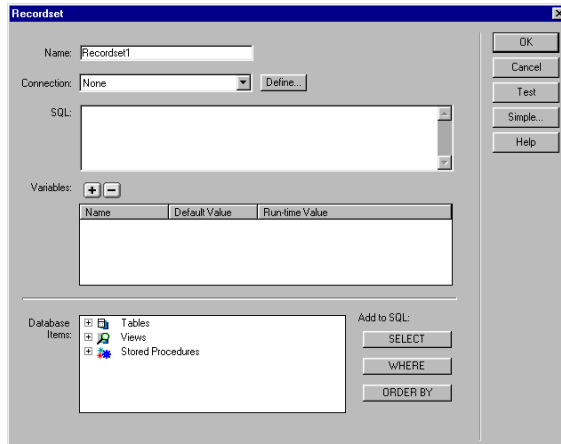
If the search page submits more than one search parameter to the server, then you must write a SQL query for the results page and use the search parameters in SQL variables.

After UltraDev inserts the SQL query on your page and the page runs on the server, each record in the database table is checked. If the specified field in a record meets your SQL query conditions, the record is included in a recordset. The SQL query in effect builds a recordset containing only the search results.

For example, field sales staff may have the ability to tell which customers in a certain area have incomes above a certain level. In a form on a search page, the sales associate enters a geographical area and a minimum income level, then clicks the Submit button to send the two values to a server. On the server, the values are passed to the results page's SQL statement, which then creates a recordset containing only customers in the specified area with incomes above the specified level.

**To search for records in a database using SQL:**

- 1 Open the results page in UltraDev, then create a new recordset by opening the Data Bindings panel (Window > Data Bindings), clicking the Plus (+) button, and selecting Recordset (Query) from the pop-up menu.
- 2 Make sure the advanced Recordset dialog box appears.



If the simple Recordset dialog box appears instead, switch to the advanced Recordset dialog box by clicking the Advanced button.

- 3 Enter a name for the recordset and choose a connection.

The connection should be to a database containing data you want the user to search.

- 4 Enter a Select statement in the SQL text area.

Make sure the statement includes a Where clause with variables to hold the search parameters. In the following example, the variables are called `varLastName` and `varDept`:

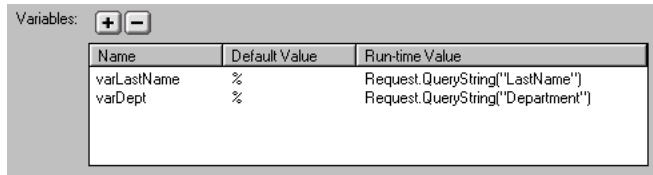
```
SELECT EMPLOYEEID, FIRSTNAME, LASTNAME, DEPARTMENT, EXTENSION ↵  
FROM EMPLOYEE WHERE LASTNAME LIKE 'varLastName' AND DEPARTMENT ↵  
LIKE 'varDept'
```

To reduce the amount of typing, you can use the tree of database items at the bottom of the advanced Recordset dialog box. For instructions, see “Defining a recordset using SQL” on page 128.

For help on SQL syntax, see “SQL Primer” on page 251.

- 5 Give the SQL variables the values of the search parameters by clicking the Plus (+) button in the Variables area and entering the variable's name, default value (the value the variable should take if no run-time value is returned), and run-time value (usually a server object holding a value sent by a browser, such as an request variable).

In the following ASP example, the HTML form on the search page uses the GET method and contains one text field called "LastName" and another called "Department".



Name	Default Value	Run-time Value
varLastName	%	Request.QueryString("LastName")
varDept	%	Request.QueryString("Department")

In ColdFusion, the run-time values would be `#LastName#` and `#Department#`. In JSP, the run-time values would be `request.getParameter("LastName")` and `request.getParameter("Department")`.

- 6 If you want, click Test to create an instance of the recordset using the default variable values.

The default values simulate the values that would otherwise have been returned from the search page. Click OK to close the test recordset.

- 7 If you're satisfied with the recordset, click OK.

## Displaying the records

After creating a recordset to hold the search results, you may want to display the information on the results page. Displaying the records is a simple matter of dragging individual columns from the Data Bindings panel to the results page. You can add navigation links to move forward and backward through the recordset, or you can create a repeated region to display more than one record on the page. You can also add links to a detail page.

To learn more about displaying dynamic content on a page, see the following chapters:

- "Adding Dynamic Content" on page 133
- "Displaying Database Records" on page 147

## Creating a detail page for a results page

Your set of search pages can include a detail page to display more information about specific records on the results page. On the results page, the records are typically displayed in a repeated region, and each record has a link. When a user clicks one of the links, the detail page opens and displays more information about the selected record.

### Modifying a results page to work with a detail page

Your results page should have a repeated region to display more than one record at a time, and each record in the repeated region should have a link to the detail page. The link must not only open the detail page, it must also tell the detail page what record the user selected. The Go to Detail Page server behavior creates this kind of link.

**To modify the results page to work with a detail page:**

- 1 Create a blank detail page (File > New) and name the file.

Set aside the page for now. You'll work on it later.

- 2 Open the results page in UltraDev.

- 3 Make sure the results are displayed in a repeated region.

For instructions, see "Displaying multiple records" on page 151.

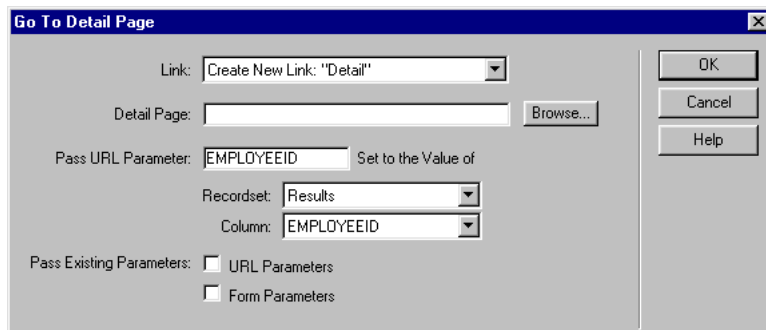
- 4 In the repeated region, select the text or image to double as a link.

If you're working in the Live Data window, select the text or image in the first region displayed.

**Note:** The text or image you select can be dynamic.

- 5 Create the link to the detail page by opening the Server Behaviors panel (Window > Server Behaviors), clicking the Plus (+) button, and choosing Go to Detail Page from the pop-up menu.

The Go to Detail Page dialog box appears.



- 6 In the Detail Page box, click Browse and locate the detail page file you created in step 1.
- 7 Specify the information you want to pass to the detail page by selecting a recordset and a column from the Recordset and Column pop-up menus.  
Typically, the information is a record's unique key ID.
- 8 Click OK.

The results page passes the value to the detail page in a URL parameter, which is simply a variable appended to the end of the URL used to open the detail page. For example, if the URL parameter is called `id` and the detail page is called `customerdetail.asp`, then the URL will look something like the following when the user clicks on the link:

```
http://www.mysite.com/customerdetail.asp?id=43
```

The first part of the URL, `http://www.mysite.com/customerdetail.asp`, opens the detail page. The second part, `?id=43`, is the URL parameter. It tells the detail page what record to find and display. The term `id` is the name of the URL parameter and `43` is its value. In this example, the URL parameter contains the record's ID number, `43`.

## Building the detail page using a server behavior

After modifying the results page to work with your detail page, complete the detail page. You can build the detail page with a combination of a regular recordset and a server behavior, or you can build one with only a filtered recordset. This section describes how to build the first kind of detail page. To learn how to build the second kind, see “Building the detail page using a filtered recordset” on page 175.

First, lay out the detail page using the Dreamweaver design tools. For more information, see Dreamweaver Help (Help > Using Dreamweaver) or the *Using Dreamweaver* book.

Second, define a recordset for the page, or copy and paste the recordset from the results page. The detail page will extract the record details from this recordset. For instructions, see “Defining a recordset” on page 125 and “Copying a recordset to another page” on page 131.

Third, bind the recordset columns to the page. In the Data Bindings panel (Window > Data Bindings), select columns in the recordset and drag them onto the page.

Fourth, add a server behavior that reads the record ID in the URL parameter passed by the results page and retrieves the record. If you omit this step, the server will retrieve the first record in the recordset.

**To retrieve a specific record using a server behavior:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button, choose Move to Record from the pop-up menu, then choose Move to Specific Record.

The Move to Specific Record dialog box appears.



- 2 In the Move to Record In pop-up menu, select the recordset you defined for the detail page.
- 3 In the Where Column pop-up menu, select the column that contains the value passed by the results page.

For example, if the results page passes a record ID number, choose the column containing record ID numbers.

- 4 In the Matches URL Parameter box, enter the name of the URL parameter passed by the results page.

For example, if the URL the results page used to open the detail page is `www.mysite.com/customerdetail.asp?id=43`, then enter `id` in the Matches URL Parameter box.

You can also get the name by switching to the results page, opening the Server Behaviors panel (Window > Server Behaviors), and double-clicking the Go to Detail Page server behavior. Check the Pass URL Parameter name.

- 5 Click OK.

## Building the detail page using a filtered recordset

Another approach to building a detail page is to filter your recordset so that only a single record remains—the record the user selected on the results page. This method can improve the performance of your application because the recordset contains only one record.

First, lay out the page using the Dreamweaver design tools. For more information, see Dreamweaver Help (Help > Using Dreamweaver) or the *Using Dreamweaver* book.

Second, define a recordset for the page, or copy and paste the recordset from the results page. The detail page will extract the record details from this recordset. For instructions, see “Defining a recordset” on page 125 and “Copying a recordset to another page” on page 131.

Third, create a recordset filter to retrieve the record specified on the results page. If you create a recordset using the simple Recordset dialog box, you can use the Filter boxes to create the filter. If you use the advanced Recordset dialog box, you can modify your SQL query to create the filter.

**To retrieve a specific record using a recordset filter:**

- 1 Make sure the detail page contains a recordset.
- 2 Open the recordset by double-clicking its name in the Data Bindings panel (Window > Data Bindings).

- 3 Make sure the simple Recordset dialog box appears.

If the advanced Recordset dialog box appears instead, switch to the simple Recordset dialog box by clicking Simple. If UltraDev informs you that it can't switch (usually because your query is too complex to display in the simple Recordset dialog box), you must use a SQL query to find the record; please skip to the next procedure in this section.

- 4 Complete the Filter section as follows to find and display the record specified in the URL parameter passed by the results page:
  - From the first pop-up menu in the Filter area, select the column in the database table containing values that match the value of the URL parameter passed by the results page.

For example, if the URL parameter contains a record ID number, choose the column containing record ID numbers.

- From the pop-up menu beside the first menu, select the equal sign (it should already be selected).
- From the third pop-up menu, select URL Parameter.

The results page passes information identifying the user's selection to the detail page in a URL parameter.

- In the fourth box, enter the name of the URL parameter passed by the results page.

For example, if the URL the results page used to open the detail page is `www.mysite.com/customerdetail.asp?id=43`, then enter `id`.

You can also get the name by switching to the results page, opening the Server Behaviors panel (Window > Server Behaviors), and double-clicking the Go to Detail Page server behavior. Check the Pass URL Parameter name.

- 5 Click OK.



- 6 If not already done, bind the recordset columns to the page by selecting the columns in the Data Bindings panel (Window > Data Bindings) and dragging them onto the page.

**To retrieve a specific record using a SQL query:**

- 1 Make sure the detail page contains a recordset.
- 2 Open the recordset by double-clicking its name in the Data Bindings panel (Window > Data Bindings).
- 3 Make sure the advanced Recordset dialog box appears.

If the simple Recordset dialog box appears instead, switch to the advanced Recordset dialog box by clicking the Advanced button.

- 4 Add a Where clause in your SQL statement to find the record the user selected on the results page.

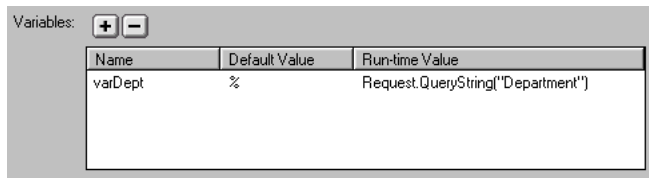
The Where clause should contain a variable to hold the value passed in the URL parameter. In the following example, the variable is called `varDept`:

```
SELECT * FROM EMPLOYEES  
WHERE DEPARTMENT = 'varDept'
```

For help on SQL syntax, see “SQL Primer” on page 251.

- 5 Give the variable the value the results page passed in the URL parameter by clicking the Plus (+) button in the Variables area and entering the variable's name, default value (the value the variable should take if no run-time value is returned), and run-time value.

In the following ASP example, the results page passes a URL parameter called `Department`.



Name	Default Value	Run-time Value
varDept	%	Request.QueryString("Department")

- 6 Click OK.
- 7 If not already done, bind the recordset columns to the page by selecting the columns in the Data Bindings panel (Window > Data Bindings) and dragging them onto the page.

## Working with related pages

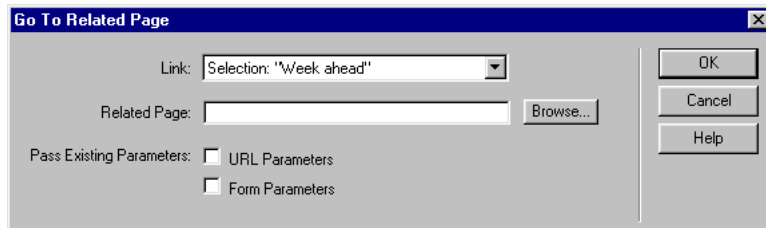
In some situations, you may want to display a page that is not a search page, a results page, or a detail page, but you don't want to lose the information the page has received from an HTML form or a URL parameter. Instead of using a standard link to open the related page, create the link using the Go to Related Page server behavior. The resulting link not only opens the related page, it passes existing parameters to that page. For example, you can pass search parameters from one page to another and save the user from entering the search parameters again.

Before adding a Go to Related Page server behavior to a page, make sure the page receives parameters from an HTML form (in other words, the form's ACTION attribute specifies the page), or from a URL parameter, such as when the page is the link destination of another page with a Go to Related Page server behavior.

**To create a link that passes existing form parameters to a related page:**

- 1 On the page, select the text string or images to act as a link to the related page.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Go to Related Page from the pop-up menu.

The Go to Related Page dialog box appears.



- 3 In the Related Page box, click Browse and locate the related page file.  
If the current page submits data to itself, enter the current page's file name.
- 4 If the parameters you want to pass were received directly from an HTML form using the GET method, or are contained in the page's query string, select the URL Parameters option.
- 5 If the parameters you want to pass were received directly from an HTML form using the POST method, select the Form Parameters option.
- 6 Click OK.

When the new link is clicked, the page passes the parameters to the related page using a query string. For example, suppose a form text field is called "lastname" and the related page is called special\_offer.cfm. The URL will look something like the following when the user clicks the link:

`http://www.mysite.com/special_offer.cfm?lastname=Anderson`

The first part of the URL, `http://www.mysite.com/special_offer.cfm`, opens the related page. The second part, `?lastname=Anderson`, is the URL parameter that passes the form parameter to the related page.



## CHAPTER 9

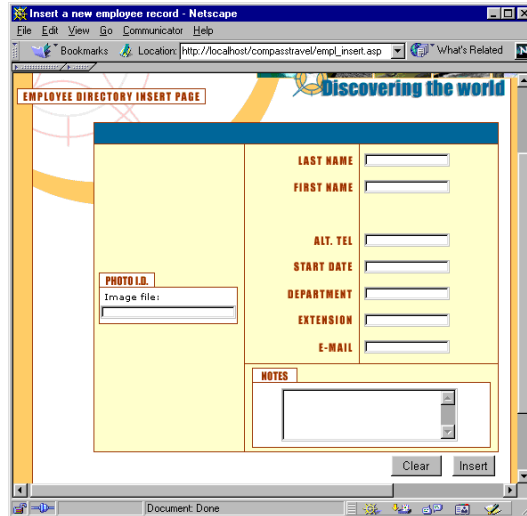
### Building Pages That Edit Database Records

---

Macromedia UltraDev comes with a set of server behaviors that let users add, update, and delete records with their Web browsers. You can also use live objects to create fully functional HTML forms for inserting or updating records.

## Building a page to insert records

Your application can contain a page that lets users insert new records in a database. For example, the following page inserts a new record in an employee database:



An insert page requires two building blocks:

- An HTML form that lets users enter data
- An Insert Record server behavior that updates the database

You can add these building blocks in a single operation using the Record Insertion Form live object, or you can add them separately using the Dreamweaver form tools and the Server Behaviors panel.

**Note:** The insert page can contain only one record-editing server behavior at a time. For example, you cannot add an Update Record or a Delete Record server behavior to the insert page.

### Building the insert page in one operation

You can add the basic building blocks of an insert page in a single operation using the Record Insertion Form live object. The live object adds both an HTML form and an Insert Record server behavior to the page.

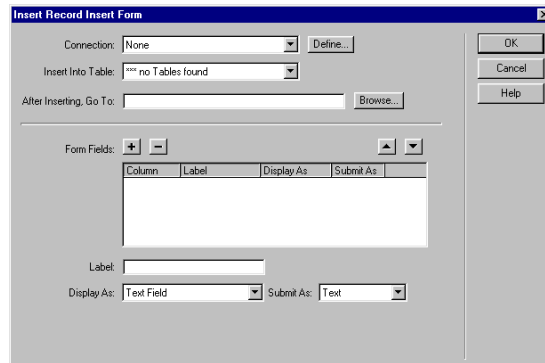
You can also add the building blocks separately using the form tools and the Server Behaviors panel. For more information, see “Building the insert page block by block” on page 184.

After placing the building blocks on the page, you can use the Dreamweaver design tools to customize the form to your liking, or the Server Behaviors panel to edit the Insert Record server behavior.

**To build the insert page with the Record Insertion Form live object:**

- 1 Open the page in Design view, then choose Insert > Live Objects > Record Insertion Form.

The Insert Record Insertion Form dialog box appears.



- 2 Specify the database table into which the record should be inserted by completing the Connection and Insert Into Table pop-up menus.

If your site does not have a connection to the database yet, click Define to create one.

- 3 In the “After Inserting, Go To” box, enter the page to open after the record is inserted into the table.

Next, you’ll create an HTML form for data entry by completing the bottom half of the dialog box.

- 4 In the Form Fields area, specify the form objects you want to include on the insert page’s HTML form, and which columns in your database table each form object should update.

By default, UltraDev creates a form object for each column in the database table. If your database automatically generates unique key IDs for each new record created, remove the form object corresponding to the key column by selecting it in the list and clicking the Minus (-) button. This eliminates the risk that the user of the form will enter an ID value that already exists.

You can also change the order of the form objects on the HTML form by selecting a form object in the list and clicking the up or down arrow on the right side of the dialog box.

**5** Specify how each data-entry field should be displayed on the HTML form by clicking a row in the Form Fields table and entering the following information in the boxes below the table:

- In the Label box, enter a descriptive label to display beside the data-entry field. By default, UltraDev displays the table column's name in the label.
- In the Display As pop-up menu, choose a form object to serve as the data-entry field. You can choose Text Field, Text Area, Menu, Checkbox, Radio Group, and Text. For read-only entries, choose Text.
- In the Submit As pop-up menu, choose the data format expected by your database table. For example, if the table column only accepts numeric data, choose Numeric.
- Set the form object's properties. You have different options depending on the form object you choose as your data-entry field. For text fields, text areas, and text, you can enter an initial value. For menus and radio groups, you open another dialog box to set the properties. For checkboxes, select the Checked or Unchecked option.

**6** Click OK.

UltraDev adds both an HTML form and an Insert Record server behavior to your page. The form objects are laid out in a basic table, which you can freely customize using the Dreamweaver page design tools. (Make sure all the form objects remain within the form's boundaries.)

To edit the server behavior, open the Server Behaviors panel (Window > Server Behaviors) and double-click the Insert Record behavior.

## Building the insert page block by block

You can add the basic building blocks of an insert page separately using the form tools and the Server Behaviors panel.

You can also add the building blocks all at once using the Record Insertion Form live object. For more information, see “Building the insert page in one operation” on page 182.

The first step is to add an HTML form to the page to let users enter data.



**To add an HTML form to an insert page:**

- 1 Create a new page (File > New) and lay out your page using the Dreamweaver design tools.
- 2 Add an HTML form by placing the insertion point where you want the form to appear and choosing Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form's boundaries, which are represented by thin red lines.

- 3 Name the HTML form by clicking the `<form>` tag at the bottom of the Document window to select the form, opening the Property inspector (Window > Properties), and entering a name in the Form Name box.

You don't have to specify an `action` or `method` attribute for the form to tell it where and how to send the record data when the user clicks the Submit button. The Insert Record server behavior sets these attributes for you.

- 4 Add a form object (Insert > Form Objects) for each column in the database table you want to insert records into.

The form objects are for data entry. Text fields are common for this purpose, but you can also use list/menus, checkboxes, and radio buttons.

For more information on form objects, see "Creating Forms," in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

- 5 Add a Submit button to the form (Insert > Form Objects > Button).
- 6 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "Insert Record":

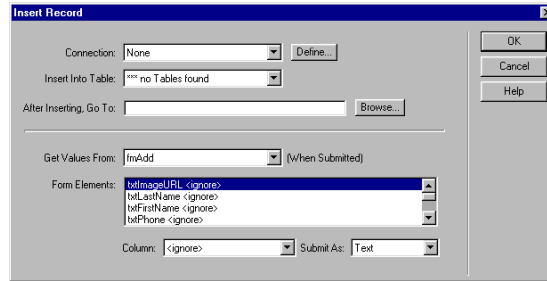


The next step is to add the Insert Record server behavior to insert records in a database table.

**To add a server behavior to insert records in a database table:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Insert Record from the pop-up menu.

The Insert Record dialog box appears.



- 2 Use the Connection and Insert Into Table pop-up menus to specify the database table into which the record should be inserted.
- 3 In the “After Inserting, Go To” box, enter the page to open after the record is inserted into the table.
- 4 In the Get Values From pop-up menu, choose the HTML form used to enter the data.

UltraDev will automatically choose the first form on your page.

- 5 Specify what each object on your form will update in the database table by selecting a form object in the Form Elements list, then choosing a table column from the Column pop-up menu and data type from the Submit As pop-up menu.

The data type is the kind of data the column in your database table is expecting (text, numeric, Boolean checkbox values). For example, if the column in the table accepts only numeric values, choose Numeric from the Submit As pop-up menu.

The Submit As pop-up menu lists two Date data types. Choose the Date MS Access data type for Microsoft Access databases; choose the Date data type for any other type of database.

Repeat the procedure for each form object in the Form Elements list.

- 6 Click OK.

## Building a page to update records

Your application can contain a page that lets users update existing records in a database table. An update page is usually a detail page working in tandem with a results page. The results page lets the user choose a record to update, then passes the choice to the update page.

An update page has three building blocks:

- A filtered recordset to retrieve the record from a database table
- An HTML form to let users modify the record's data
- An Update Record server behavior to update the database table

You can add the HTML form and the server behavior to the page in a single operation using the Record Update Form live object, or you can add them separately using the Dreamweaver form tools and the Server Behaviors panel.

**Note:** The update page can contain only one record-editing server behavior at a time. For example, you cannot add an Insert Record or a Delete Record server behavior to the update page.

### Identifying the record to update

When users want to update a record, they must first find that record in the database. Accordingly, you need a search and a results page to work with the update page. For instructions on creating a search and a results page, see “Building Pages That Search Databases” on page 165.

The results page tells the update page which record to update by passing it a URL parameter. Accordingly, make sure the results page has a Go to Detail Page server behavior that names the update page as the detail page. For instructions, see “Modifying a results page to work with a detail page” on page 173.

### Retrieving the record to update

After the results page passes a URL parameter to the update page identifying the record to update, the update page must read the parameter, retrieve the record from the database table, and store it temporarily in a recordset.

**To retrieve the record to update:**

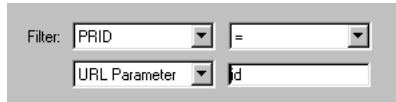
- 1 In the Data Bindings panel (Window > Data Binding), click the Plus (+) button and choose Recordset (Query).

If the advanced Recordset dialog box appears, click the Simple button to open the simple Recordset dialog box.

- 2 Name the recordset and specify where the data you want to update is located using the Connection and Table pop-up menus.

- 3 Click the Selected option and select a key column (usually the record ID column) and the columns containing the data to be updated.
- 4 Configure the Filter area so that the value of your key column equals the value of the corresponding URL parameter passed by the results page.

This kind of filter creates a recordset that contains only the record specified by the results page. For example, if your key column contains record ID information and is called PRID, and if the results page passes the corresponding record ID information in the URL parameter called `id`, then here's how your Filter area should look:



The screenshot shows a configuration interface for a filter. It consists of a 'Filter:' label followed by a dropdown menu containing 'PRID'. To the right of this is an equals sign, followed by another dropdown menu. Below these is a 'URL Parameter' dropdown menu containing 'id', and a text input field.

For more information, see “Building the detail page using a filtered recordset” on page 175.

- 5 Click OK.

When the user selects a record on the results page, the update page will generate a recordset containing only the selected record.

## Completing the update page in one operation

You can add the final two building blocks of an update page in a single operation using the Record Update Form live object. The live object adds both an HTML form and an Update Record server behavior to the page.

Before you can use the live object, your Web application must be able to identify the record to update, and your update page must be able to retrieve it. (See “Identifying the record to update” on page 187 and “Retrieving the record to update” on page 187.)

You can also add the HTML form and the Update Record server behavior separately using the form tools and the Server Behaviors panel. (See “Completing the update page block by block” on page 191.)

After the live object places the building blocks on the page, you can use the Dreamweaver design tools to customize the form to your liking, or the Server Behaviors panel to edit the Update Record server behavior.

To build the update page with the Record Update Form live object:

- 1 Open the page in Design view, then choose Insert > Live Objects > Record Update Form.

The Insert Record Update Form dialog box appears.

Insert Record Update Form

Connection: None Define...

Table to Update: \*\*\* no Tables found

Select Record From: Update

Unique Key Column: EMPLOYEEID ☒ Numeric

Alter Updating, Go To: Browse...

Form Fields:

Column	Label	Display As	Submit As

Label:

Display As: Text Field Submit As: Text

OK Cancel Help

- 2 Use the Connection and Table to Update pop-up menus to specify the database table containing the records to be updated.
- 3 In the Select Record From pop-up menu, specify the recordset that contains the record displayed in the HTML form.
- 4 In the Unique Key Column pop-up menu, select a key column (usually the record ID column) to identify the record in the database table.

If the value is a number, select the Numeric option. A key column usually accepts only numeric values, but sometimes it accepts text values.

- 5 In the “After Updating, Go To” box, enter the page to open after the record is updated in the table.

Next, you’ll create an HTML form for data entry by completing the bottom half of the dialog box.

- 6 In the Form Fields area, specify the form objects you want to include on the update page's HTML form, and which columns in your database table each form object should update.

By default, UltraDev creates a form object for each column in the database table. If your database automatically generates unique key IDs for each new record created, remove the form object corresponding to the key column by selecting it in the list and clicking the Minus (-) button. This eliminates the risk that the user of the form will enter an ID value that already exists.

You can also change the order of the form objects on the HTML form by selecting a form object in the list and clicking the up or down arrow on the right side of the dialog box.

- 7 Specify how each data-entry field should be displayed on the HTML form by clicking a row in the Form Fields table and entering the following information in the boxes below the table:

- In the Label box, enter a descriptive label to display beside the data-entry field. By default, UltraDev displays the table column's name in the label.
- In the Display As pop-up menu, choose a form object to serve as the data-entry field. You can choose Text Field, Text Area, Menu, Checkbox, Radio Group, and Text. For read-only entries, choose Text.
- In the Submit As pop-up menu, choose the data format expected by your database table. For example, if the table column only accepts numeric data, choose Numeric.
- In the Default Value box for text fields, text areas, and text make sure the field displays the current value from the database (UltraDev enters a script to perform this task by default). Click the lightning-bolt icon to change the database value to be displayed.
- Set the properties of other form objects. For menus and radio groups, open another dialog box to set the properties. For checkboxes, define a comparison between the current record's value for the checkbox and a given value to determine whether the checkbox is checked or not when the record is displayed.

- 8 Click OK.

The live object adds both an HTML form and an Update Record server behavior to your page. The form objects are laid out in a basic table, which you can freely customize using the Dreamweaver page design tools. (Make sure all the form objects remain within the form's boundaries.)

To edit the server behavior, open the Server Behaviors panel (Window > Server Behaviors) and double-click the Update Record behavior.

## Completing the update page block by block

You can add the final two basic building blocks of an update page separately using the form tools and the Server Behaviors panel.

Before you can add the building blocks, your Web application must be able to identify the record to update, and your update page must be able to retrieve it. (See “Identifying the record to update” on page 187 and “Retrieving the record to update” on page 187.)

You can also add the remaining building blocks all at once using the Record Update Form live object. (See “Completing the update page in one operation” on page 188.)

The first step is to add an HTML form to the page to let users modify the data.

### To add an HTML form to an update page:

- 1 Create a new page (File > New) and lay out your page using the Dreamweaver design tools.
- 2 Add an HTML form by placing the insertion point where you want the form to appear and choosing Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form’s boundaries, which are represented by thin red lines.

- 3 Name the HTML form by clicking the `<form>` tag at the bottom of the Document window to select the form, opening the Property inspector (Window > Properties), and entering a name in the Form Name box.

You don’t have to specify an `action` or `method` attribute for the form to tell it where and how to send the record data when the user clicks the Submit button. The Update Record server behavior sets these attributes for you.

- 4 Add a form object (Insert > Form Objects) for each column you want to update in the database table.

The form objects are for data entry. Text fields are common for this purpose, but you can also use list/menus, checkboxes, and radio buttons.

Each form object should have a corresponding column in the recordset you defined earlier. The only exception is the unique key column, which should have no corresponding form object.

For more information on form objects, see “Creating Forms,” in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

- 5 Add a Submit button to the form (Insert > Form Objects > Button).

- 6 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "Update Record":



The next step is to display the record in the form by binding the form objects to database table columns.

#### To display the record in the form:

- 1 Make sure you defined a recordset to hold the record the user wants to update.

For more information, see "Retrieving the record to update" on page 187.

- 2 Drag a database table column from the Data Bindings panel (Window > Data Bindings) to its corresponding form object on the page.

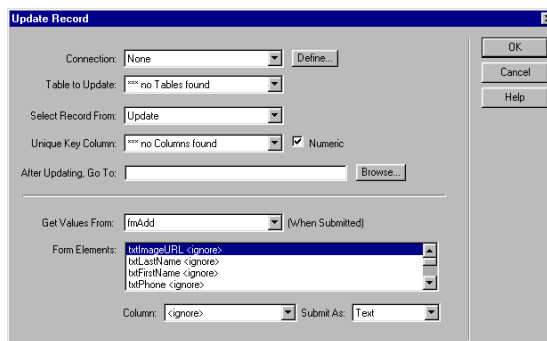
For more information, see "Making form objects dynamic" on page 138.

The final step is to add the Update Record server behavior to update the database table after the user modifies the record.

#### To add a server behavior to update the database table:

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Update Record from the pop-up menu.

The Update Record dialog box appears.



- 2 Use the Connection and Table to Update pop-up menus to specify the database table containing the records to be updated.



**3** In the Select Record From pop-up menu, specify the recordset that contains the record displayed in the HTML form.

**4** In the Unique Key Column pop-up menus, select a key column (usually the record ID column) to identify the record in the database table.

Select the Numeric option if the value is a number. A key column usually accepts only numeric values, but sometimes it accepts text values.

**5** In the “After Updating, Go To” box, enter the page to open after the record has been updated in the table.

**6** In the Get Values From pop-up menu, choose the HTML form used to edit the record data.

UltraDev will automatically choose the first form on your page.

**7** Specify what each object on your form will update in the database table by selecting a form object in the Form Elements list, then choosing a table column from the Column pop-up menu and a data type from the Submit As pop-up menu.

The data type is the kind of data the column in your database table is expecting (text, numeric, Boolean checkbox values). For example, if the column in the table accepts only numeric values, choose Numeric from the Submit As pop-up menu.

The Submit As pop-up menu lists two Date data types. Choose the Date MS Access data type for Microsoft Access databases; choose the Date data type for any other type of database.

Repeat the procedure for each form object in the Form Elements list.

**8** Click OK.

## Building a page to delete a record

Your application can contain a page that lets users delete existing records in a database table. A delete page is usually a detail page working in tandem with a results page. The results page lets the user choose a record to delete, then passes the choice to the delete page.

A delete page has four building blocks:

- A filtered recordset to retrieve the record from a database table
- A read-only display of the data about to be deleted
- A Submit button to send the delete command to the server
- A Delete Record server behavior to update the database table

**Note:** The delete page can contain only one record-editing server behavior at a time. For example, you cannot add an Insert Record or an Update Record server behavior to the delete page.

## Identifying the record to delete

When users want to delete a record, they must first find that record in the database. Accordingly, you need a search and a results page to work with the delete page. For instructions on creating a search and a results page, see “Building Pages That Search Databases” on page 165.

The results page tells the delete page which record to delete by passing it a URL parameter. Accordingly, make sure the results page has a Go to Detail Page server behavior that names the delete page as the detail page. For instructions, see “Modifying a results page to work with a detail page” on page 173.

## Retrieving the record to delete

After the results page passes a URL parameter to the delete page identifying the record to delete, the delete page must read the parameter, retrieve the record from the database table, and temporarily store the record in a recordset.

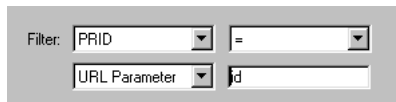
**To retrieve the record to delete:**

- 1 In the Data Bindings panel (Window > Data Binding), click the Plus (+) button and choose Recordset (Query).

If the advanced Recordset dialog box appears, click the Simple button to open the simple Recordset dialog box.

- 2 Name the recordset and specify where the data you want to delete is located using the Connection and Table pop-up menus.
- 3 In the Columns area, select the All option to select all the columns in the database table.
- 4 Configure the Filter area so that the value of your key column equals the value of the corresponding URL parameter passed by the results page.

This kind of filter creates a recordset that contains only the record specified by the results page. For example, if your key column contains record ID information and is called PRID, and if the results page passes the corresponding record ID information in the URL parameter called `id`, then here’s how your Filter area should look:



Filter: PRID =  
URL Parameter id

For more information, see “Building the detail page using a filtered recordset” on page 175.

- 5 Click OK.

When the user selects a record on the results page, the delete page will generate a recordset containing only the selected record.

Next you'll add a read-only display of the data to be deleted.

## Displaying the data to be deleted

It is good form to display the record before the user deletes it to confirm that the user wants to delete it.

**To add a read-only display of the record to be deleted:**

- 1 Make sure you defined a recordset to hold the record the user wants to delete.

For more information, see “Retrieving the record to delete” on page 194.

- 2 Drag a column from the Data Bindings panel (Window > Data Bindings) to the page.

Dynamic content appears on the page. You can drop the dynamic content on the page as is, or you can drop it in an HTML table. For more information, see “Making text dynamic” on page 134.

## Sending the delete command to the server

The delete page uses a Submit button to send the delete command to the server. To add a Submit button to your page, you must create an HTML form. The form can consist only of the Submit button.

**To add a Submit button to a delete page:**

- 1 In Design view, place the insertion point where you want the Submit button to appear and choose Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form's boundaries, which are represented by thin red lines.

- 2 Name the HTML form by clicking the `<form>` tag at the bottom of the Document window to select the form, opening the Property inspector (Window > Properties), and entering a name in the Form Name box.

You don't have to specify an `action` or `method` attribute for the form to tell it where and how to send the record data when the user clicks the Submit button. The Delete Record server behavior sets these attributes for you.

- 3 Add a Submit button to the form (Insert > Form Objects > Button).

- 4 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "Delete Record":



Next, you add the Delete Record server behavior to update the database table after the user clicks the Submit button.

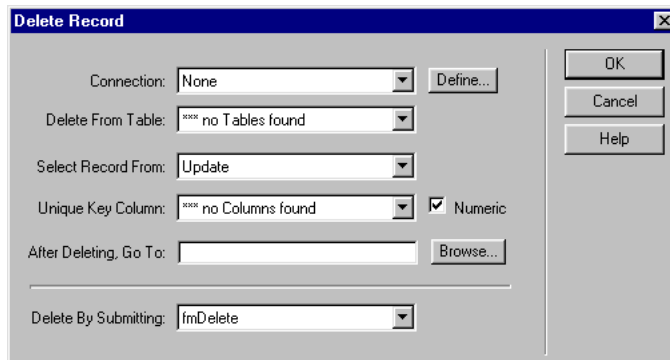
## Deleting the record from the database table

The final step is to add the Delete Record server behavior to update the database table after the user clicks the Submit button.

**To add a server behavior to delete the database table:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Update Record from the pop-up menu.

The Delete Record dialog box appears.



- 2 Use the Connection and Delete From Table pop-up menus to specify the database table containing the records to be deleted.
- 3 In the Select Record From pop-up menu, specify the recordset that contains the record to be deleted.
- 4 In the Unique Key Column pop-up menu, select a key column (usually the record ID column) to identify the record in the database table.

If the value is a number, select the Numeric option. A key column usually accepts only numeric values, but sometimes it accepts text values.

- 5 In the “After Deleting, Go To” box, enter the page to open after the record has been deleted from the database table.
- 6 In the Delete By Submitting pop-up menu, specify the HTML form with the Submit button that sends the delete command to the server.  
UltraDev will automatically choose the first form on your page.
- 7 Click OK.



## CHAPTER 10

### Building Pages That Restrict Access to Your Site

.....

You can use Dreamweaver UltraDev to build the following pages to restrict access to your site:

- A page that requires users to register the first time they visit the site
- A page that lets registered users log in to the site
- Pages that only authorized users can view

## Building a registration page

Your Web application can contain a page that requires users to register the first time they visit your site. For example, the following page asks first-time users to register:

The screenshot shows a web browser window with the title "Macromedia Membership Center - Create an Account". The browser's menu bar includes "File", "Edit", "View", "Go", "Communicator", and "Help". The page content is titled "Become a Member" and includes the text: "Signing up for membership only takes a second. Please enter the following information to set up an ID and password." The form contains the following fields and elements:

- "courtesy title" with a dropdown menu showing "<Select>".
- "first name\*" and "last name\*" text input fields, with a small "MI" input field between them.
- "preferred greeting name" text input field.
- "e-mail address\*" text input field.
- "Macromedia ID\*" text input field.
- "password\*" text input field, with a note below it: "Your password must be between 4 and 25 characters and contain at least one alpha character."
- "retype password\*" text input field.
- "remember me" checkbox, which is checked.

The browser's status bar at the bottom shows "Document" and various navigation icons.

A registration page is made up of the following building blocks:

- A database table to store log-in information about the users.
- An HTML form that lets users choose a user name and password. You can also use the form to obtain other personal information from users.
- An Insert Record server behavior to update the database table of site users.
- A Check New Username server behavior to make sure the user name entered by the user is not taken by another user.

**Note:** You can delete or change the properties of any server behavior you add to a page. See "Editing server behaviors on a page" on page 164.

### Storing log-in information about users

A registration page requires a database table to store the log-in information entered by users. Make sure your database table contains a user name and a password column. If you want logged-in users to have different access privileges, include an access privilege column. (See "Storing access privileges in the user database" on page 209.)



If you want to set a common password for all users of the site, configure your database application (Microsoft Access, Microsoft SQL Server, Oracle, and so on) to enter the password in each new user record by default. In most database applications, you can set a column to a default value each time a new record is created. Set the default value to the password.

You can also use the database table to store other useful information about the user.

## Letting users choose a user name and password

You add an HTML form to the registration page to let users choose a user name and password (if applicable).

### To let users choose a user name and password:

- 1 Create a new page (File > New) and lay out your registration page using the Dreamweaver design tools.
- 2 Add an HTML form by placing the insertion point where you want the form to appear and choosing Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form's boundaries, which are represented by thin red lines.

- 3 Name the HTML form by clicking the `<form>` tag at the bottom of the Document window to select the form, opening the Property inspector (Window > Properties), and entering a name in the Form Name box.

You don't have to specify an `action` or `method` attribute for the form to tell it where and how to send the record data when the user clicks the Submit button. The Insert Record server behavior sets these attributes for you (see "Updating the table of users in the database" on page 202).

- 4 Add text fields (Insert > Form Objects > Text Field) to let the user enter a user name and password.

The form can also have more form objects to record other personal data.

You should add labels (either as text or images) beside each form object to tell users what they are. You should also line up the form objects by placing them inside an HTML table.

For more information on form objects, see "Creating Forms," in the *Using Dreamweaver* guide or in Dreamweaver Help (Help > Using Dreamweaver).

- 5 Add a Submit button to the form (Insert > Form Objects > Button).

- 6 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "Register":



The next step is to add the Insert Record server behavior to insert records in the table of users in the database.

## Updating the table of users in the database

You add an Insert Record server behavior to update the table of users in the database.

**To update the table of users in the database:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Insert Record from the pop-up menu.

The Insert Record dialog box appears.

- 2 Use the Connection and Insert Into Table pop-up menus to specify the table of users in the database.
- 3 In the "After Inserting, Go To" box, enter the page to open after the record is inserted into the table.
- 4 In the Get Values From pop-up menu, choose the HTML form used to obtain the user's user name and password.

UltraDev will automatically choose the first form on your page.

- 5 Specify what each object on your form will update in the database table by selecting a form object in the Form Elements list, then choosing a table column from the Column pop-up menu and a data type from the Submit As pop-up menu.

The data type is the kind of data the column in your database table is expecting (text, numeric, Boolean checkbox values). Password or user name columns usually expect text.

For example, in the Form Elements list, click the password text field, choose the column in the database table where the password should be stored, and then choose the Text data type.

Repeat the procedure for each form object in the Form Elements list.

- 6 Click OK.

The final step is to make sure the user name is not used by another registered user.

## Making sure the chosen user name is unique

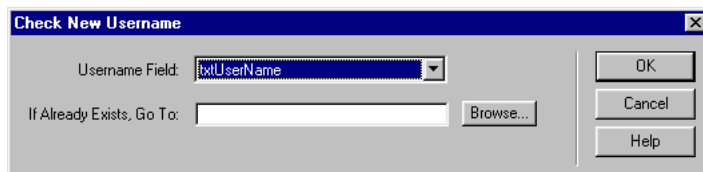
You add a server behavior to make sure the user name entered is not taken by another registered user.

When the user clicks the Submit button on the registration page, the server behavior compares the user name entered by the user against the user names stored in a database table of registered users. If no matching user name is found in the database table, the server behavior carries out the insert record operation normally. If a matching user name is found, the server behavior cancels the insert record operation and opens a new page (usually a page alerting the user that the user name is already taken).

**To make sure the chosen user name is unique:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose User Authentication > Check New Username from the pop-up menu.

The Check New Username dialog box appears.



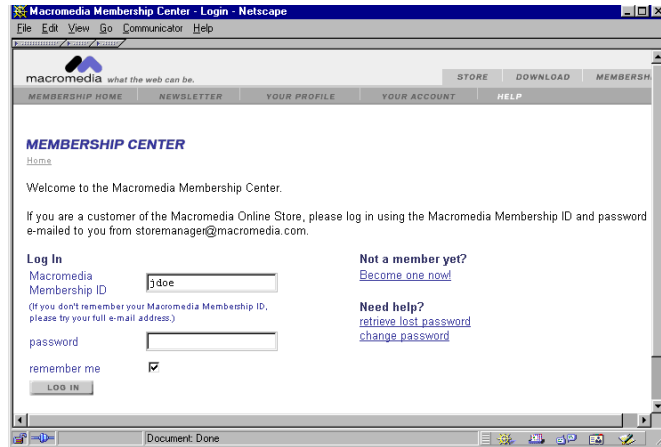
- 2 In the Username Field pop-up menu, choose the form text field your visitors use to enter a user name.
- 3 In the “If Already Exists, Go To” box, specify a page to open if a matching user name is found in the database table.

The opened page should alert the user that the user name is already taken and let the user try again.

- 4 Click OK.

## Building a log-in page

Your Web application can contain a page that lets registered users log in to the site. For example, the following page asks registered users to log in:



A log-in page is made up of the following building blocks:

- A database table of registered users
- An HTML form to let users enter a user name and password
- A Log In User server behavior to make sure the entered user name and password are valid

A session variable consisting of the user name is created for the user when the user logs in successfully.

**Note:** You can delete or change the properties of any server behavior you add to a page. See “Editing server behaviors on a page” on page 164.

## Creating a database table of registered users

You need a database table of registered users to verify that the user name and password entered in the log-in page are valid. Use your database application and a registration page to create the table. For more information, see “Building a registration page” on page 200.

## Letting users log in

You add an HTML form to the page to let users log in by entering a user name and password.

### To let users log in:

- 1 Create a new page (File > New) and lay out your log-in page using the Dreamweaver design tools.
- 2 Add an HTML form by placing the insertion point where you want the form to appear and choosing Form from the Insert menu.

An empty form is created on the page. You may have to turn on Invisible Elements (View > Visual Aids > Invisible Elements) to see the form's boundaries, which are represented by thin red lines.

- 3 Name the HTML form by clicking the `<form>` tag at the bottom of the Document window to select the form, opening the Property inspector (Window > Properties), and entering a name in the Form Name box.

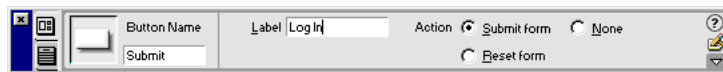
You don't have to specify an `action` or `method` attribute for the form to tell it where and how to send the record data when the user clicks the Submit button. The Log In User server behavior sets these attributes for you (see "Checking the user name and password" on page 206).

- 4 Add a user name and a password text field (Insert > Form Objects > Text Field) to the form.

Add labels (either as text or images) beside each text field, and line up the text fields by placing them inside an HTML table and setting the table's `border` attribute to 0.

- 5 Add a Submit button to the form (Insert > Form Objects > Button).
- 6 If you wish, change the label of the Submit button by selecting the button, opening the Property inspector (Window > Properties), and entering a new value in the Label box.

For example, here's the Property inspector of a button labeled "Log In":



The next step is to add the Log In User server behavior to make sure the entered user name and password are valid.

## Checking the user name and password

You add a Log In User server behavior to make sure the user name and password users enter are valid.

When a user clicks the Submit button on the log-in page, the Log In User server behavior compares the values entered by the user against the values for registered users. If the values match, the server behavior opens one page (usually the site's start page). If the values do not match, the server behavior opens another page (usually a page alerting the user that the log-in failed).

**To check the user name and password:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose User Authentication > Log In User from the pop-up menu.

The Log In User dialog box appears.

The screenshot shows the 'Log In User' dialog box. It has a title bar with 'Log In User' and a close button. The dialog is divided into several sections. The first section, 'Get Input From Form:', has a dropdown menu set to 'Login'. Below this, 'Username Field:' has a dropdown set to 'txtUserName', and 'Password Field:' has a dropdown set to 'txtPassword'. The next section, 'Validate Using Connection:', has a dropdown set to 'None'. Below this, 'Table:' has a dropdown showing '\*\*\* no Tables found'. Then, 'Username Column:' and 'Password Column:' have empty dropdown menus. The 'If Log In Succeeds, Go To:' section has a text field and a 'Browse...' button, with a checkbox 'Go To Previous URL (if it exists)' below it. The 'If Log In Fails, Go To:' section has a text field and a 'Browse...' button. The final section, 'Restrict Access Based On:', has two radio buttons: 'Username and Password' (which is selected) and 'Username, Password, and Access Level'. Below the second radio button is a 'Get Level From:' dropdown menu. On the right side of the dialog, there are three buttons: 'OK', 'Cancel', and 'Help'.

- 2 Specify the form and the form objects visitors use to enter their user name and password.
- 3 Specify the database table and columns that contain the user names and passwords of all the registered users.

The server behavior compares the user name and password a visitor enters on the log-in page against the values in these columns.

- 4 Specify a page to open if the log-in succeeds.

The specified page is usually the site's start page.

- 5 Specify a page to open if the log-in fails.

The specified page usually alerts the user that the log-in failed and lets the user try again.

- 6 Specify whether to grant access to the page based on user name and password alone, or based on authorization level too.

For more information, see the next section.

- 7 Click OK.

## Building a page only authorized users can access

Your Web application can contain a protected page that only authorized users can access. For example, if a user attempts to bypass the log-in page by typing the protected page's URL in a browser, the user is redirected to another page. Similarly, if you set the authorization level for a page to Administrator, then only users with Administrator access privileges can view the page. If a logged-in user attempts to access the protected page without the proper access privileges, the user is redirected to another page.

You can also use authorization levels to review newly registered users before granting them full access to the site. For example, you may want to receive payment before allowing a user access to the member pages of the site. To do so, you can protect the member pages with a Member authorization level and only grant newly registered users Guest privileges. After receiving payment from the user, you can upgrade the user's access privileges to Member (in the database table of registered users).

If you do not plan to use authorization levels, you can protect any page on your site simply by adding a Restrict Access To Page server behavior to the page. The server behavior redirects to another page any user who has not successfully logged in.

If you do plan to use authorization levels, you can protect any page on your site with the following building blocks:

- An extra column in your users database table to store each user's access privileges
- A Restrict Access To Page server behavior to redirect unauthorized users to another page

In this case, the server behavior redirects to another page any user who does not have the required access privileges.

In either case, you can add a link to the protected page that lets a user log out and clears any session variables. For more information, see "Logging out users" on page 210.

## Redirecting unauthorized users to another page

To prevent unauthorized users from accessing a page, you add a Restrict Access To Page server behavior to it. The server behavior redirects the user to another page if the user attempts to bypass the log-in page by typing the protected page's URL in a browser, or if the user is logged in but attempts to access the protected page without the proper access privileges.

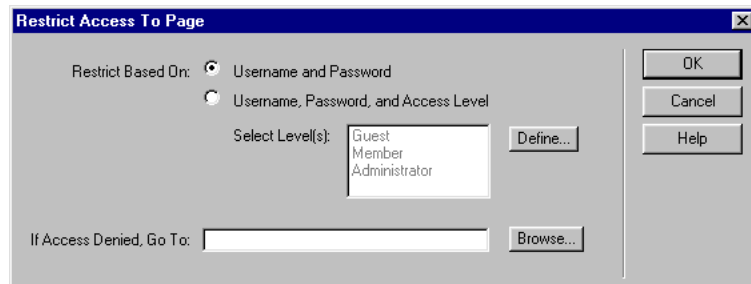
**Note:** The Restrict Access To Page server behavior can only protect HTML pages. It does not protect other site resources such as image files and audio files.

If you want to give many pages on your site the same access rights, you can copy and paste access rights from one page to another.

### To redirect unauthorized users to another page:

- 1 Open the page you want to protect.
- 2 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose User Authentication > Restrict Access To Page from the pop-up menu.

The Restrict Access To Page dialog box appears.



- 3 If you want only users with certain access privileges to view the page, select the “Username, Password, and Access Level” option and specify the authorization levels for the page.

For example, you can specify that only users with Administrator privileges can view the page by selecting Administrator in the authorization levels list.

To add authorization levels to the list, click Define. In the Define Access Levels list that appears, enter a new authorization level and click the Plus (+) button. UltraDev stores the level, saving you from typing it for each page. Make sure you enter strings that exactly match the strings in your user database.



- 4 If you want to set more than one authorization level for a page, Control-click (Windows) or Command-click (Macintosh) the levels in the list.

For example, you can specify that any user with Guest, Member, or Administrator privileges can view the page.

- 5 Specify the page to open if an unauthorized user attempts to open the protected page.

Make sure the page you choose is not itself protected.

- 6 Click OK.

#### **To copy and paste a page's access rights to other pages on the site:**

- 1 Open the protected page and select the Restrict Access To Page server behavior listed in the Server Behaviors panel (not the one in the Plus (+) pop-up menu).

- 2 Click the arrow button in the top right corner of the panel and choose Copy from the pop-up menu.

The Restrict Access To Page server behavior is copied to your system's Clipboard.

- 3 Open another page you want to protect in the same way.

- 4 In the Server Behaviors panel (Window > Server Behaviors), click the arrow button in the top right corner and choose Paste from the pop-up menu.

- 5 Repeat steps 3 and 4 for each page you want to protect.

### **Storing access privileges in the user database**

This building block is required only if you want certain logged-in users to have different access privileges. If you simply require users to log in, then you don't need to store access privileges.

If you want certain logged-in users to have different access privileges, make sure your database table of users contains a column specifying each user's access privileges (Guest, User, Administrator, and so on). The access privileges of each user should be entered in the database by the site administrator.

In most database applications, you can set a column to a default value each time a new record is created. Set the default value to the most common access privilege on your site (for example, Guest), then manually change the exceptions (for example, changing Guest to Administrator). The user is now allowed to access all administrator pages.

Make sure each user in the database has a single access privilege, such as Guest or Administrator, not multiple privileges like “User, Administrator”. If you want to set multiple access privileges for your pages (for example, all guests and administrators can see this page), then set those privileges at the page level, not the database level. For more information, see “Redirecting unauthorized users to another page” on page 208.

## Logging out users

When a user logs in successfully, a session variable is created that consists of the user name. When the user leaves your site, you can use the Log Out User server behavior to clear the session variable and redirect the user to another page (usually a “goodbye” or “thank you” page).

You can invoke the Log Out User server behavior when the user clicks a link or when a specific page loads.

### To add a link to let users log out:

- 1 On the page, select text or an image to serve as the link.
- 2 In the Server Behaviors panel, click the Plus (+) button and choose User Authentication > Log Out User.

The Log Out User dialog box appears.

- 3 Specify a page to open when the user clicks the link.  
The page is usually a “goodbye” or “thank you” page.
- 4 Click OK.

### To log out a user when a specific page loads:

- 1 Open the page that will load in UltraDev.  
The page is usually a “goodbye” or “thank you” page.
- 2 In the Server Behaviors panel, click the Plus (+) button and choose User Authentication > Log Out User.

The Log Out User dialog box appears.

- 3 Select the “Log Out When Page Loads” option.
- 4 Click OK.

## CHAPTER 11

### Customizing UltraDev

---

Dreamweaver UltraDev has the tools you need to edit existing data formats, create new data formats, install and create new server behaviors, and edit existing server behaviors.

#### Editing and creating data formats

You apply a data format to dynamic text to display data in more user-friendly ways. For example, you can make the date “3/29/00” in your recordset appear as “March 29, 2000” on the page. For more information on applying data formats, see “Making text dynamic” on page 134.

You can edit the various data formats available in UltraDev, or create new ones.

##### To edit an UltraDev data format:

- 1 Open a page containing dynamic text in Design view.
- 2 Select any dynamic text.
- 3 Make sure the Data Bindings panel is open (Window > Data Bindings), and click the down arrow in the Format column.  
If the down arrow is not visible, expand the panel.
- 4 Select Edit Format List from the pop-up menu.  
The Edit Format List dialog box appears.
- 5 Double-click any of the listed formats.
- 6 Make your changes and click OK.
- 7 Click OK to close the Edit Format List dialog box.

**To create a new UltraDev data format:**

- 1 Open a page containing dynamic text in Design view.
- 2 Select any dynamic text.
- 3 Make sure the Data Bindings panel is open (Window > Data Bindings), and click the down arrow in the Format column.  
If the down arrow is not visible, expand the panel.
- 4 Select Edit Format List from the pop-up menu.  
The Edit Format List dialog box appears.
- 5 Click the Plus (+) button and select a format type—for example, Currency.
- 6 Define the format and click OK.
- 7 Enter a name for the new format in the Name column.
- 8 Click OK to close the Edit Format List dialog box.

## Installing more server behaviors

To give your Web application more functionality, you can install additional server behaviors. For example, you can download and install a server behavior from the Macromedia Exchange for UltraDev site (Help > UltraDev Exchange). You can also access the Macromedia Exchange from the Server Behaviors panel (Window > Behaviors) by clicking the Plus (+) button and choosing Get More Server Behaviors.

To install an extension in UltraDev, launch the Package Manager by selecting Commands > Manage Extensions, then select File > Install Package in the Package Manager. For more information, see the help system that comes with the Package Manager.

If you're an experienced application developer proficient in JavaScript, VBScript, Java, or ColdFusion, you can write your own server behaviors.

## Creating server behaviors

Creating a server behavior consists of writing one or more code blocks and specifying where each block belongs in a page's HTML source code. When you apply the server behavior to a page, it inserts the code blocks in the location you specified. For instructions, see "Writing a code block" on page 213 and "Positioning a code block" on page 215.

If the page designer needs to supply certain parameters before the code can be written in the page, you must create a dialog box prompting the designer to enter the parameter values. For instructions, see “Creating a dialog box for the server behavior” on page 216.

Finally, thoroughly test your server behavior before making it available to others. For guidelines, see “Testing server behaviors” on page 221.

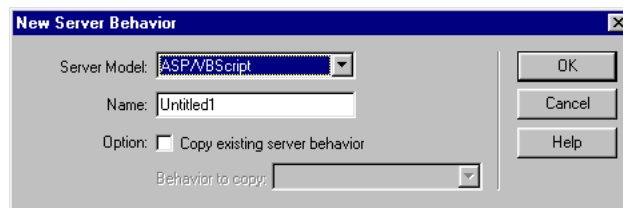
## Writing a code block

Use the Server Behavior Builder to write the code block or blocks the behavior will insert in a page.

**To write code blocks for the server behavior:**

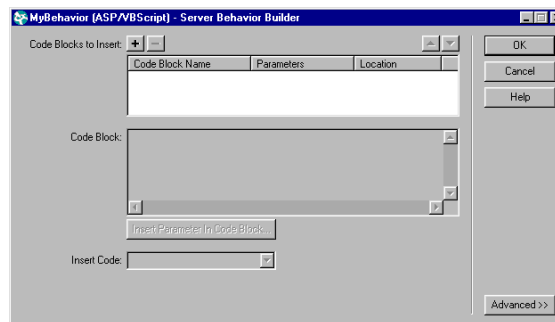
- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose New Server Behavior from the pop-up menu.

The New Server Behavior dialog box appears.



- 2 Select a server model, enter a name for your server behavior, and click OK.

The Server Behavior Builder appears.



- 3 Start a code block by clicking the Plus (+) button, entering a name for the code block, and clicking OK.

One convention is to name code blocks as follows:

*NameOfBehavior\_NameOfBlock*

Here is an example:

`MoveToRecord_Init`

- 4 In the Code Block text area, enter the run-time code to be inserted in the page.

The run-time code must be a single tag or ASP/JSP script block. If you need to insert multiple tags or script blocks, split them into separate code blocks.

You can copy and paste code from other pages.

For more information, see “Coding guidelines” on page 219.

- 5 If the page designer needs to supply certain parameters before the code can be written in the page, enter parameter markers in the code.

For instructions, see “Creating a dialog box for the server behavior” on page 216.

- 6 Specify where to insert the code block in the page’s HTML source code by using the Insert Code pop-up menu.

Code blocks are inserted relative to tags in the page or relative to a tag selected by the page designer. For more information, see “Positioning a code block” on page 215.

- 7 To specify more advanced settings, click Advanced.

For more information, see “Using the advanced options” on page 219.

- 8 Repeat steps 3 to 7 for each code block in your server behavior.

- 9 If you have two or more code blocks with the same insert location, you can change their positions relative to each other by clicking the up and down arrows.

- 10 If you did not define any designer-supplied parameters in your code, click OK.

UltraDev creates the server behavior with no dialog box. The new server behavior appears in the Plus (+) menu of the Server Behaviors panel.

- 11 If you did define designer-supplied parameters in your code, UltraDev prompts you to configure a dialog box for the server behavior before creating it.

For more information, see “Creating a dialog box for the server behavior” on page 216.

## Positioning a code block

When writing code blocks in the Server Behavior Builder, you must specify where to insert the code block in the page's HTML source code.

In the Insert Code pop-up menu, you can choose to insert the code block above the opening `<html>` tag, below the closing `</html>` tag, relative to another tag in the page, or relative to a tag selected by the page designer.

### To position a code block above the `<html>` tag:

- 1 In the Insert Code pop-up menu, choose Above the `<html>` Tag.
- 2 Specify a location above the tag by choosing an option in the Relative Position pop-up menu.

You can insert the block at the beginning of the file, just before code blocks that open recordsets, just after code blocks that open recordsets, or just above the `<html>` tag. You can also specify a custom position.

- 3 If you want to specify a custom position, choose Custom Position from the Relative Position pop-up menu, then assign a weight to the code block.

UltraDev assigns a weight of 50 to all recordset-opening code blocks inserted above the `<html>` tag. If the weight of two or more blocks match, UltraDev randomly sets the order among the blocks.

Use the Custom Position option when you need to insert more than one code block in a particular order. For example, if you want to insert an ordered series of three code blocks after the code blocks that open recordsets, you could enter a weight of 60 for the first block, 65 for the second, and 70 for the third.

### To position a code block below the closing `</html>` tag:

- 1 In the Insert Code pop-up menu, choose Below the `</html>` Tag.
- 2 Specify a location below the tag by choosing an option in the Relative Position pop-up menu.

You can insert the block just after the `</html>` tag, just before code blocks that close recordsets, just after code blocks that close recordsets, or just before the end of the file. You can also specify a custom position.

- 3 If you want to specify a custom position, choose Custom Position from the Relative Position pop-up menu, then assign a weight to the code block.

UltraDev assigns a weight of 50 to all recordset-closing code blocks inserted below the `</html>` tag. If the weight of two or more blocks match, UltraDev randomly sets the order among the blocks.

Use the Custom Position option when you need to insert more than one code block in a particular order. For example, if you want to insert an ordered series of three code blocks before the code blocks that close recordsets, you could enter a weight of 30 for the first block, 35 for the second, and 40 for the third.

**To position a code block relative to another tag on the page:**

- 1 In the Insert Code pop-up menu, choose Relative To a Specific Tag.
- 2 In the Tag box, enter the tag or select one from the pop-up menu.  
If you enter a tag, don't include the angled brackets (< >).
- 3 Specify a location relative to the tag by choosing an option in the Relative Position pop-up menu.

You can insert your code block just before or just after the opening or closing tags. You can also replace the tag with the code, insert the code as the value of an attribute of the tag (a box appears to let you choose the attribute), or insert the code inside the opening tag.

**To position a code block relative to a tag selected by the page designer:**

- 1 In the Insert Code pop-up menu, choose Relative To the Selection.
- 2 Specify a location relative to the selection by choosing an option in the Relative Position pop-up menu.

You can insert your code block just before or just after the selection. You can also replace the selection with your code block, or you can wrap the code block around the selection.

If you want to wrap the code block around a selection, the selection must consist of an opening and closing tag with nothing in between. Example:

```
<CFIF Day="Monday"></CFIF>
```

The opening tag piece of your code block is inserted before the selection's opening tag and the closing tag piece of your code block is inserted after the selection's closing tag.

## Creating a dialog box for the server behavior

If the page designer needs to supply certain parameters before the code can be written in the page, you must create a dialog box prompting the designer to enter the parameter values.

You begin creating the dialog box by defining the designer-supplied parameters in your code. After defining all the designer-supplied parameters in your code, you can generate the dialog box for your server behavior.

**Note:** A parameter is added to your code block without your intervention if you specify that your code should be inserted relative to a specific tag chosen by the page designer (that is, you chose Relative to a Specific Tag in the Insert Code pop-up menu). The parameter adds a tag menu to the behavior's dialog box to let the page designer choose a tag.



**To define a designer-supplied parameter in your code:**

Type a parameter marker at the point in the code where you want to insert the designer-supplied parameter value. Use this format:

```
@@parameterName@@
```

For example, suppose your code contains the following line:

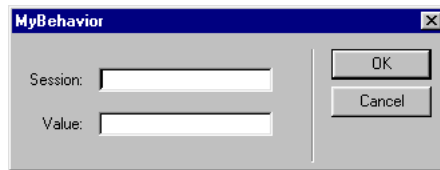
```
<% Session("abcd") = 5; %>
```

To let the page designer supply the name and value of the session variable, replace the `abcd` and `5` strings in the code with the following parameter markers:

```
<% Session("@@Session@@") = @@Value@@; %>
```

You can also highlight the string (for example, `abcd`), then click the Insert Parameter In Code Block button. Enter a parameter name (for example, `Session`) and click OK. UltraDev replaces every instance of the highlighted string with parameter markers.

In the above example, `Session` and `Value` are added to the parameters column of the list at the top of the window. When you click the Next button on the Server Behavior Builder, UltraDev looks for the parameter markers and creates a dialog box for your server behavior containing two controls: a text box labeled `Session` and another text box labeled `Value`.

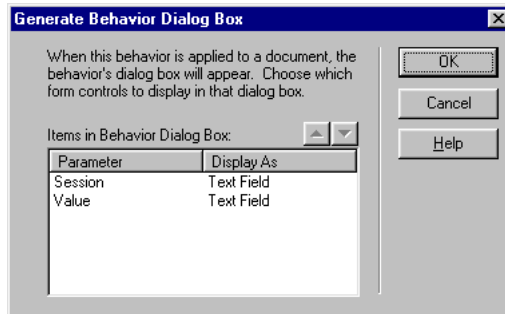


UltraDev uses the parameters' names to label the controls on the dialog box.

**To generate the dialog box for your server behavior:**

- 1 In the Server Behavior Builder, click Next.

A dialog box appears listing all of the designer-supplied parameters you defined in your code.



- 2 If you want, change the display order of the dialog box controls by selecting a parameter and clicking the up and down arrows.
- 3 If you want, change a parameter's control by selecting the parameter and choosing another control in the Display As column.
- 4 Click OK.

UltraDev generates a dialog box for your server behavior. To view the dialog box, click the Plus (+) button in the Server Behaviors panel (Window > Server Behaviors), and select your server behavior from the pop-up menu.

**To edit the dialog box of a server behavior you created:**

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Edit Server Behaviors from the pop-up menu.
- 2 Select your server behavior from the list, and click Open.

The Server Behavior Builder appears with your server behavior.

- 3 Click Next.

A dialog box appears listing all the designer-supplied parameters you defined in your code.

- 4 If you want, change the display order of the dialog box controls by selecting a parameter and clicking the up and down arrows.
- 5 If you want, change a parameter's control by selecting the parameter and choosing another control in the Display As column.
- 6 Click OK.

## Coding guidelines

In general, your server behavior's code should be compact and robust. Web application developers are very sensitive to the code added to their pages. Follow generally accepted coding practices for your language (JavaScript, VBScript, ColdFusion, or Java). In the interest of brevity, include minimal or no comments in the code.

An important requirement is error checking: your code should handle error cases gracefully. Try to foresee every eventuality: What if a parameter request fails? What if no records are returned from a query?

Your code should be clearly identifiable and avoid name collisions with existing code. For example, if the page contains a function called `hideLayer()` and a global variable called `ERROR_STRING`, and your server behavior inserts code that uses those names too, you will have problems.

Macromedia precedes all functions and globals with the prefix `MM_` to prevent them from conflicting with your code.

```
var MM_ERROR_STRING = "...";  
function MM_hideLayer() {
```

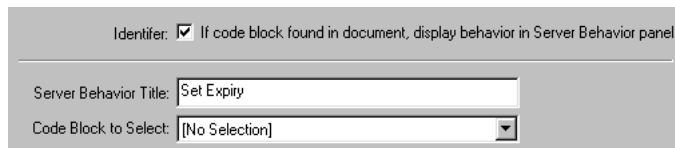
Create your own prefix for run-time functions and globals that you insert in a page. One convention is to use your initials. Never use the `MM_` prefix: it is reserved for Macromedia use only.

Make sure the code in your blocks don't resemble too closely the code in other blocks. If a code block looks too much like another code block on the page, the Server Behavior panel might mistakenly identify the first code block as an instance of the second code block (or conversely). A simple solution is to add a comment to a code block to make it more unique.

## Using the advanced options

Once you specify the source code and insert location for each code block, the server behavior is completely defined. In most cases, you don't need to specify any additional information.

If you are an advanced user, you may want to modify additional parameters by clicking the Advanced button in the Server Behavior Builder. The builder expands to display several new options.



The screenshot shows a dialog box titled "Identifier: ☒ If code block found in document, display behavior in Server Behavior panel". Below this, there are two fields: "Server Behavior Title:" with a text input containing "Set Expiry", and "Code Block to Select:" with a dropdown menu showing "[No Selection]".

**Identifier** specifies whether or not the code block should be treated as an identifier.

By default, every code block is an identifier. If UltraDev finds an identifier code block anywhere in a document, it lists the behavior in the Server Behaviors panel. Use the Identifier checkbox to specify whether the code block should be treated as an identifier.

At least one of the server behavior's code blocks must be an identifier. A code block should not be an identifier if one of the following conditions applies:

- The same code block is used by some other server behavior
- The code block is so simple that it might occur naturally on the page

**Server Behavior Title** specifies the title of the behavior in the Server Behaviors panel.

When the page designer clicks the Plus (+) button on the Server Behaviors panel, the new server behavior's title will appear in the pop-up menu. When a designer applies an instance of a server behavior to a document, the behavior appears in the list of applied behaviors in the Server Behaviors panel. Use the Server Behavior Title box to specify the contents of the Plus (+) pop-up menu and the list of applied behaviors.

The initial value in the box is the name you supplied in the New Server Behavior dialog box. As parameters are defined, the name is automatically updated so that the parameters appear inside parentheses after the server behavior name.

Set Session Variable (@@Name@@, @@Value@@)

If the user accepts the default value, everything before the parentheses will appear in the Plus (+) pop-up menu (for example, Set Session Variable). The name Plus the parameters will appear in the list of applied behaviors—for example, Set Session Variable ("abcd", "5").

**Code Block to Select** specifies what code block is selected when the user selects the behavior in the Server Behaviors panel.

When you apply a server behavior, one of the code blocks within the behavior is designated the “code block to select.” If you apply the server behavior and then select the behavior in the Server Behaviors panel, in the Document window UltraDev automatically selects the designated block. By default, UltraDev selects the first code block that is not above the `<html>` tag. If all the code blocks are above the `<html>` tag, then UltraDev selects the first one. Advanced users can specify which code block is the selected one.

## Testing server behaviors

The Macromedia Exchange recommends you perform the following tests on each server behavior you create:

- Apply the behavior from the Server Behaviors panel. If it has a dialog box, enter valid data in each field and click OK. Verify that no error occurs when the behavior is applied. Verify that the run-time code for the server behavior appears in the Code inspector.
- Apply the server behavior again and enter invalid data in each field of the dialog box. Try leaving the field blank, using large or negative numbers, using invalid characters (such as /, ?, :, \*, and so on), and using letters in numeric fields. Write form validation routines to handle invalid data. Validation routines involves hand-coding, which is beyond the scope of this book.

After successfully applying your server behavior to the page, verify the following:

- Check the Server Behaviors panel to make sure the name of the server behavior appears in the list of behaviors added to the page.
- If applicable, verify that server-side script icons show up on the page. The generic server-side script icons are gold shields. To see the icons, turn on Invisible Elements (View > Visual Aids > Invisible Elements).
- Open the Code inspector (Window > Code Inspector) and verify that no invalid code is generated (invalid HTML code is highlighted in yellow).

In addition, if your server behavior inserts code in the document establishing a connection to a database, create a test database to test the code inserted in the document. Verify the connection by defining queries that produce different sets of data, and different sizes of data sets.

Finally, upload the page to the server and open it in a browser. View the page's HTML source code and verify that no invalid HTML has been generated by the server-side scripts.

## Editing server behaviors

When you apply a server behavior to a page, it inserts code in the page.

Experienced coders can modify the default code the server behavior inserts in a page. This ability is useful for developers whose coding practices differ from the Macromedia style.

You can change the code written by any custom server behavior. You can also change the code written by any server behavior that ships with UltraDev by making a copy of the behavior, changing the code written by the copy, then using the copy in your projects instead of the UltraDev server behavior.

**Note:** You can't directly edit the server behaviors that ship with UltraDev. You must make a copy of the behavior and edit the copy.

If you apply a behavior to a page and then edit the behavior in UltraDev, instances of the old behavior will no longer appear in the Server Behaviors panel. The Server Behaviors panel searches the page for code that matches the code of known server behaviors. If the code of a server behavior known to the panel changes, the panel will no longer recognize earlier versions of the behavior on the page.

If you want both the old and new versions of the behavior to appear in the panel, click the Plus (+) button on the Server Behaviors panel, choose New Server Behavior, and create a copy of the old server behavior.

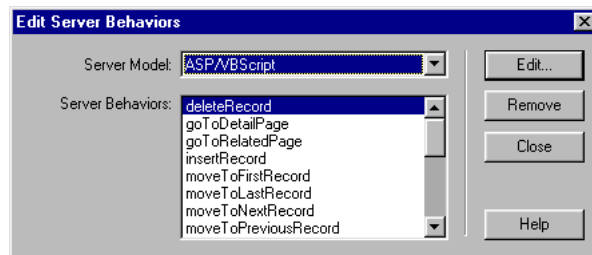
## Editing custom server behaviors

You can edit any server behavior created with the Server Behavior Builder.

**To modify the run-time code of a server behavior created with the Server Behavior Builder:**

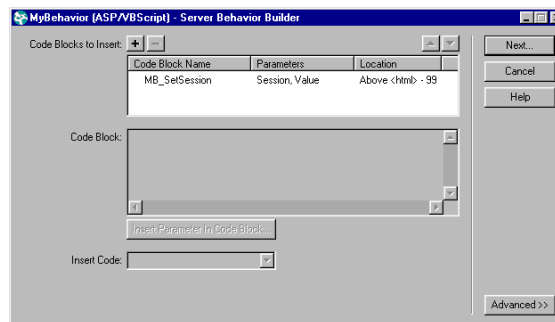
- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose Edit Server Behaviors from the pop-up menu.

The Edit Server Behaviors dialog box appears, displaying all the behaviors for the current server technology.



- 2 Select the server behavior and click Edit.

The Server Behavior Builder dialog box appears.



- 3 Select the appropriate code block and modify the code to be inserted in pages.
- 4 If you want, change or add parameter markers to the code.

For instructions, see “Creating a dialog box for the server behavior” on page 216.

- 5 If you want, change where the code block is inserted in the page’s HTML source code by choosing another option in the Insert Code pop-up menu.

For instructions, see “Positioning a code block” on page 215.

- 6 If the modified code does not contain any designer-supplied parameters, click OK.

UltraDev regenerates the server behavior without a dialog box. The new server behavior appears in the Plus (+) pop-up menu of the Server Behaviors panel.

- 7 If the modified code does contain designer-supplied parameters, click Next.

UltraDev asks you whether you want to create a new dialog box, overwriting the old one. Make your changes and click OK.

UltraDev saves all your changes in the behavior’s XML files.

## Editing the UltraDev server behaviors

You can make changes to the server behaviors that ship with UltraDev by making a copy of the behavior, changing the code written by the copy, then using the copy in your projects instead of the UltraDev server behavior.

### To modify the run-time code of one of the server behaviors that ship with UltraDev:

- 1 In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and choose New Server Behavior from the pop-up menu.

The New Server Behavior dialog box appears.

- 2 Select a server model and enter a name for your server behavior.
- 3 Select the Copy Existing Server Behavior option and choose the UltraDev server behavior in the Behavior To Copy pop-up menu.
- 4 Click OK.

The Server Behavior Builder dialog box appears.

- 5 Select the appropriate code block and modify the code to be inserted in pages.

- 6 If you want, change or add parameter markers to the code.

For instructions, see “Creating a dialog box for the server behavior” on page 216.

If you added or modified designer-supplied parameters in the code, you must update the behavior’s HTML file by hand to change the behavior’s dialog box. For more information, see the *Extending Dreamweaver & UltraDev* book or help pages.

- 7 If you want, change where the code block is inserted in the page’s HTML source code by choosing another option in the Insert Code pop-up menu.

For instructions, see “Positioning a code block” on page 215.

- 8 Click OK.

UltraDev saves all your changes in the behavior’s XML files.

Some UltraDev server behaviors are graphically represented on the page. For example, the Repeat Region server behavior is represented by a thin gray outline and a tab. The graphic representation is specified inside the `Translation` tag in the XML file. If, after editing one of these behaviors in the Server Behavior Builder, the behavior is no longer graphically represented on the page, you must hand-code the regular expressions in the `SearchPatterns` section inside the XML file’s `Translator` tag so the translator recognizes your modified version of the behavior.

## Creating other UltraDev extensions

To create other kinds of UltraDev extensions, see the *Extending Dreamweaver & UltraDev* book or help pages.



# APPENDIX A

## Beginner's Guide to Databases

.....

This appendix is intended for Dreamweaver UltraDev users who have little or no experience working with databases or database connections. It explains general concepts, not specific procedures. To see how these concepts apply in practice, see the rest of the user guide.

This appendix does not explain how to create a database in an application such as Microsoft Access. To do that, consult the printed or online documentation that came with your database application.

### About databases

The building block of a database is the record. A record is a collection of related data treated as a single entity. For example, a hockey trading card could be called a record: it brings together the name, photograph, team, and statistics of one player. Using database terms, each of these related pieces of information would be called a field: each hockey card “record” has a name field, a photograph field, a team field, and various statistic fields.

A collection of records that share the same fields is called a table because this kind of information can easily be presented in table format: each column represents a field and each row represents a record. In fact, the word *column* is synonymous with the word *field*, and the word *row* is synonymous with the word *record*.

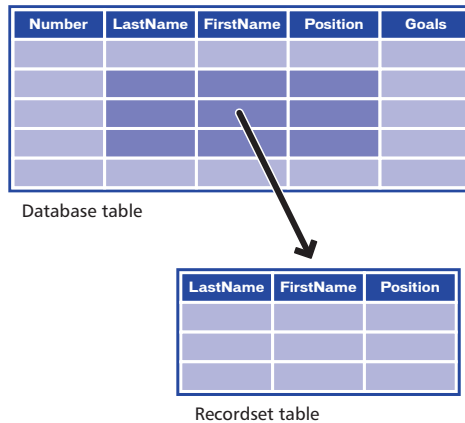
Fields (columns)

Number	LastName	FirstName	Position	Goals

Records (rows)

A database can contain more than one table, each with a unique name. These tables can be related or independent from one another.

A subset of data extracted from one or more tables is called a recordset. A recordset is also a table because it's a collection of records that share the same columns. For example, a hockey team roster listing the names and positions of the players could be called a recordset: it consists of a subset of all the possible information about the players, including goals, assists, penalty minutes, and so on.



To create a recordset, you run a database query. A query consists of search criteria. For example, the query can specify that only certain columns be included in the recordset, or that only certain records be included. For more information, see “Defining a recordset” on page 125.

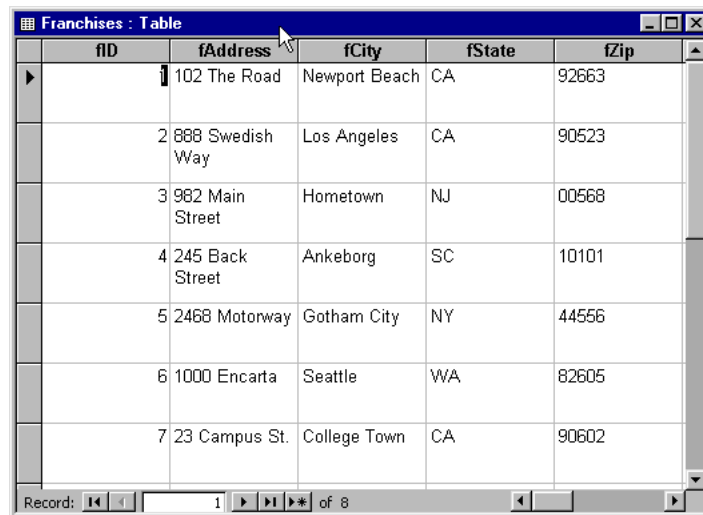
## About database connections

If you plan to use a database with your Web application, you need to create at least one database connection. Without one, the application won't know where to find the database or how to connect to it. You create a database connection in UltraDev by providing the information—or the “parameters”—the application needs to establish contact with the database.

A database should already exist before you create a connection to it. You can connect to a file-based database, such as one created in Microsoft Access, or you can connect to a server-based database system, such as one created in Microsoft SQL Server, Oracle 8i, or IBM DB2.

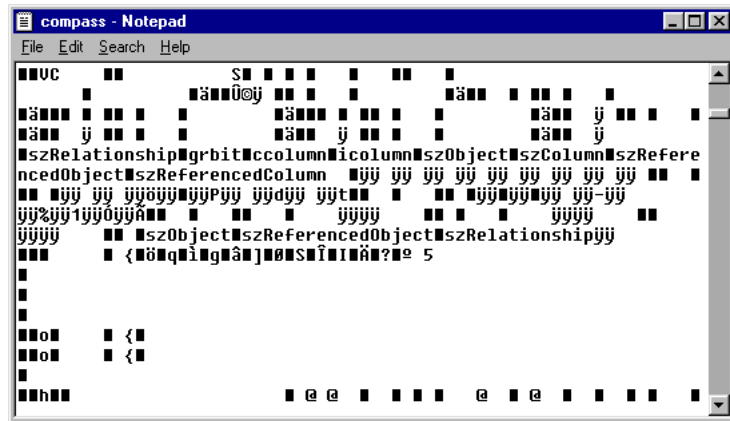
## Interfacing with the database

Data stored in a database is normally in a proprietary format in the same way text in a word-processor file is in a proprietary format. For example, here's what data looks like in Microsoft Access:



fID	fAddress	fCity	fState	fZip
1	102 The Road	Newport Beach	CA	92663
2	888 Swedish Way	Los Angeles	CA	90523
3	982 Main Street	Hometown	NJ	00568
4	245 Back Street	Ankeborg	SC	10101
5	2468 Motorway	Gotham City	NY	44556
6	1000 Encarta	Seattle	WA	82605
7	23 Campus St.	College Town	CA	90602

Here's what the same database looks like in Notepad:



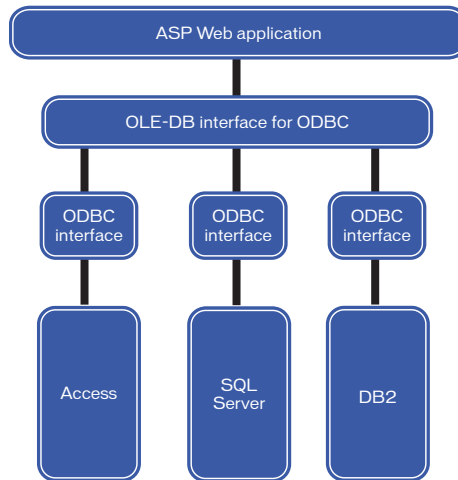
Your Web application faces the same problem as Notepad or any other application trying to access data in an unknown format: the application is unable to decipher the data. A software interface is needed between your Web application and the database allowing the application and the database to talk to each other.

Three common interfaces let applications communicate with databases. The first is called Open Database Connectivity, or ODBC; the second is called OLE DB (object linking & embedding database); and the third is called Java Database Connectivity, or JDBC.

The job of these interfaces is to act like language interpreters at diplomatic functions. For example, when a speech is given in English at the United Nations, one interpreter translates the speech for French-speaking delegates and another interpreter translates the speech for German-speaking delegates. Similarly, you use one interface for OLE DB-speaking applications, another interface for ODBC-speaking Web applications, and still another interface for JDBC-speaking applications. JSP applications are JDBC speakers, ASP applications speak OLE DB, and ColdFusion applications speak ODBC and OLE DB. (ColdFusion Server also provides native drivers to communicate with databases.)

ASP applications are also fluent ODBC speakers thanks to a built-in OLE DB/ODBC interpreter. For example, suppose you want your application to communicate with a Microsoft Access database by using a certain ODBC interface. In ASP, if you specify only the ODBC interface and no OLE DB interface, by default the application will use an OLE DB/ODBC interpreter to translate the OLE DB into ODBC, then it will use the ODBC/Access interpreter you specified to translate the ODBC into something Access can understand.

The following illustration gives you an idea of the process:



**Note:** SQL Server and DB2 are server-based database applications from Microsoft and IBM, respectively.

## Understanding the ADO wrapper

When an ASP Web application needs to interact with a database, it sends instructions to the OLE DB interface, which translates and passes the instructions to the database (or to an intervening ODBC interface, if an OLE DB interface does not exist for your database). If the database sends a response, the OLE DB interface translates it and passes it back to the ASP application.

Unfortunately, the OLE DB interface only understands instructions received from an application if the instructions are written in C++, a powerful but advanced programming language. To get around this problem, Microsoft created ActiveX Data Objects (ADO) and included it in its ASP server technology (among other technologies).

ADO is known as a wrapper: its role in ASP is to hide the complexity of OLE DB. Like the Document Object Model (DOM) of Web browsers, ADO provides ASP developers with a hierarchical series of objects that they can easily manipulate in the programming language of their choice, including JavaScript and VBScript. Common ASP objects include the request, session, and application objects.

## Using database drivers to interface with your database

The ODBC, OLE DB, and JDBC interfaces are implemented by database drivers (or “data providers” in OLE DB), which are simply pieces of software. When your Web application communicates with your database, it does so through the intermediary of a driver.

Database drivers are database-specific. For example, you can use Microsoft Access, SQL Server, and dBase drivers. Similarly, you can use OLE DB providers such as the OLE DB provider for SQL Server. Your choice depends on your database.

Drivers are written by database vendors such as Microsoft and Oracle, and by a variety of third-party software vendors. Most database drivers implement either the ODBC or the JDBC interface. New drivers (or “providers”) are slowly appearing implementing the OLE DB interface.

In Windows 95 or 98, a selection of Microsoft ODBC drivers is installed in the background when you install Microsoft Office 2000 or when you install Microsoft Data Access Components (MDAC) 2.5. The drivers installed support the following databases: Access databases, SQL Server databases, and dBASE databases.

**Note:** You can download MDAC 2.5 from the Microsoft Web site at <http://www.microsoft.com/data/download.htm>. MDAC is installed on your system when you install Office 2000.

To find out which drivers are installed on your Windows system, do the following:

- In Windows 95, 98, or NT, choose Start > Settings > Control Panel, and double-click the ODBC Data Sources icon. (Depending on your system, the icon could also be called ODBC or 32bit ODBC.) Next, click the Drivers tab.
- In Windows 2000, choose Start > Settings > Control Panel > Administrative Tools > Data Sources, then click the Drivers tab.

A list of ODBC drivers installed on the Windows system appears.

Because the Macintosh is rarely used as a database platform, few ODBC drivers exist for it.

Some common JDBC drivers include the I-net JDBC driver for Microsoft SQL Server databases, the Oracle Thin driver for Oracle databases, and the JDBC Driver for DB2 for IBM DB2 databases. For more information on JDBC drivers and their vendors, see the searchable database of JDBC drivers on the Sun Web site at <http://industry.java.sun.com/products/jdbc/drivers>.

## Invoking database drivers

An application must invoke a database driver to establish two-way communications with a database. A Web application invokes a driver by using a connection string. A connection string consists of all the information (or parameters) required to establish a connection to a database. In its simplest form, a connection string specifies a driver and a database, as in this example:

```
Driver={Microsoft Access Driver (*.mdb)};  
DBQ=C:\Inetpub\wwwroot\Scaal\scaalcoffee.mdb
```

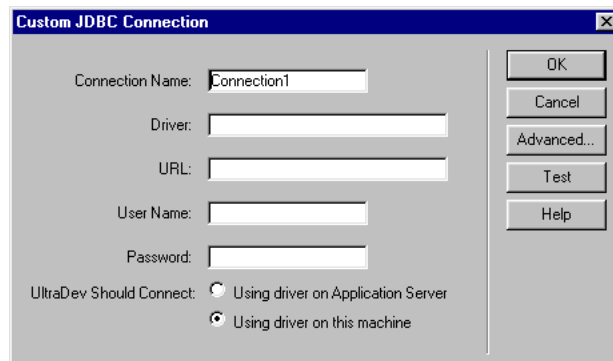
ASP connection strings can contain a Provider parameter specifying an OLE DB driver. If you omit this parameter, by default ASP uses the OLE DB provider for ODBC drivers. In the above example, the OLE DB driver for ODBC drivers would communicate with the ODBC driver, Microsoft Access Driver, which in turn would communicate with the Access database, scaalcoffee.mdb.

The parameters in a connection string may vary depending on the driver. Here's a connection string for a SQL Server database called Cases located on a server named Hoover:

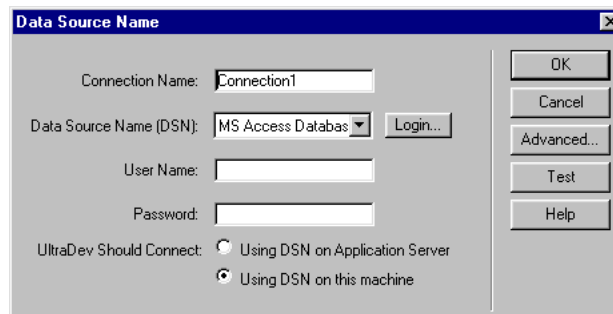
```
Driver={SQL Server};Server=Hoover;Database=Cases;  
UID=DanaS;PWD=Queequeg
```

**Note:** UID stands for user ID; PWD for password.

UltraDev simplifies the process of inserting connection strings into your pages by providing you with a dialog box in which to enter the different connection parameters. For example, here's how the dialog box to define a connection looks when you're developing a JSP application:



Here's what the same dialog box looks like when you're developing a ColdFusion application:



After you complete the dialog box and click OK, UltraDev inserts the connection string in an include file in your site.

## Using a DSN in a connection string

You can specify data source names (DSNs) in some connections. A DSN is a type of shortcut you create in Windows or ColdFusion for a connection string. Once defined, you can simply refer to the connection string by name. For example, a connection string may consist of the following parameters:

```
Driver={SQL Server};Server=Clinic-6;Database=Patients;  
UID=dholmes;PWD=stetson2
```

After defining a DSN called `patients` in Windows using the above parameters, you can use the connection string in your application by specifying a single parameter:

```
dsn=patients
```

If your application server is running on a Windows system and you defined a DSN on that system, then you can use the DSN to define an ASP or ColdFusion connection.

If you do not have physical access to a server—and so are unable to define a DSN on it—then you must use a connection string to connect to the database.

For more information on DSNs, see “Setting Up a DSN in Windows” on page 249.



## APPENDIX B

### Detailed Requirements for Building Web Applications

---

To build Web applications in Dreamweaver UltraDev, you need the following:

- A Web server
- An application server that runs on your Web server, or a Web server that doubles as an application server, such as Microsoft Personal Web Server (PWS) or Internet Information Server (IIS)
- A database or database system
- A database driver that supports your database system

The exact requirements vary depending on whether you use UltraDev to create Active Server Pages (ASP) applications, ColdFusion applications, or JavaServer Pages (JSP) applications. For more information on these technologies, see “About dynamic pages” on page 76.

### Requirements for ASP developers

To develop ASP applications in UltraDev, you’ll need a Web server, an ASP application server, a database, and a driver for your database.

## Web server

You need access to a Web server to host your ASP site. You can use any Web server that works with your chosen ASP application server.

If you're a Windows 95, 98 or NT Workstation user, you can install and run Microsoft Personal Web Server (PWS) on your local computer. For more information, see "Installing Microsoft Personal Web Server" on page 243. If you're a Windows 2000 user, you can install and run Microsoft enterprise-strength Web server, Internet Information Server (IIS) 5.0, on your local computer. (IIS is included in the Windows 2000 package.) These two Web servers have the advantage of also being ASP application servers.

## ASP application server

An application server is software that processes dynamic pages before the pages are served up to requesting browsers by the Web server.

To develop ASP pages in UltraDev, you need an application server that supports Microsoft Active Server Pages 2.0. Here are some popular choices:

- Microsoft Internet Information Server (IIS), which comes with Windows NT Server and Windows 2000, and the hardware to run it.
- Microsoft Personal Web Server (PWS), a scaled-down version of IIS that runs in Windows 95, 98, and NT Workstation.

If you have Windows 95 or NT Workstation, you can download PWS for free from the Microsoft Web site at <http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp>. If you have Windows 98, you'll find a copy in the Add-Ons/PWS folder on the Windows 98 CD.

- Chili!Soft ASP.

Microsoft IIS and PWS are both Web servers and ASP application servers. If you prefer not to use IIS or PWS, you can use another ASP engine such as Chili!Soft ASP with your existing Web server. (The Chili!Soft product also runs on Linux and Solaris platforms, among others.)

If you're a Windows 98 or NT Workstation user, you can turn your local computer into an ASP development platform by installing Microsoft PWS. Once installed, PWS functions as both your Web server and your application server. For more information, see "Installing Microsoft Personal Web Server" on page 243.

## Database

You can use almost any database with your Web application, so long as you have the appropriate database driver for it.

If you plan to build small low-cost applications, you can use a file-based database, such as one created in Microsoft Access. If you plan to build robust, business-critical applications, you can use a server-based database, such as one created in Microsoft SQL Server, Oracle 8i, or IBM DB2.

If your database is located on a system other than your Web server, make sure you have a fast connection between the two systems so that your Web application can operate quickly and efficiently.

## Database driver

A database driver acts like a kind of translator between your ASP application and your database. Data stored in a database is normally in a proprietary format, just as text in a word-processor file is in a proprietary format. A database driver lets your ASP application read and manipulate data that would otherwise be undecipherable.

The appropriate database driver for your database depends on your application and your database. ASP applications “speak” OLE DB (or ODBC through a built-in OLE DB interpreter); therefore, you need an ODBC or OLE DB database driver. To learn more about ODBC and OLE DB drivers, see “Interfacing with the database” on page 227.

The driver must also be specific to your database. If you use a Microsoft Access database, your server must have an ODBC or OLE DB driver for Microsoft Access. If you use an Oracle database, your server must have an ODBC or OLE DB driver for Oracle.

Microsoft offers a number of ODBC drivers for the most popular database packages, such as Microsoft Access, Microsoft SQL Server, and Oracle. The drivers, which only run on the Windows platform, are automatically installed with Microsoft Office and with Windows 2000. They are also provided with the Microsoft Data Access Components (MDAC) 2.5 package, which you can download for free from the Microsoft Web site at <http://www.microsoft.com/data/download.htm>.

If you need a specific ODBC driver and your Web server runs on a Windows system, you can easily find out whether or not the ODBC driver you need is installed on your system.

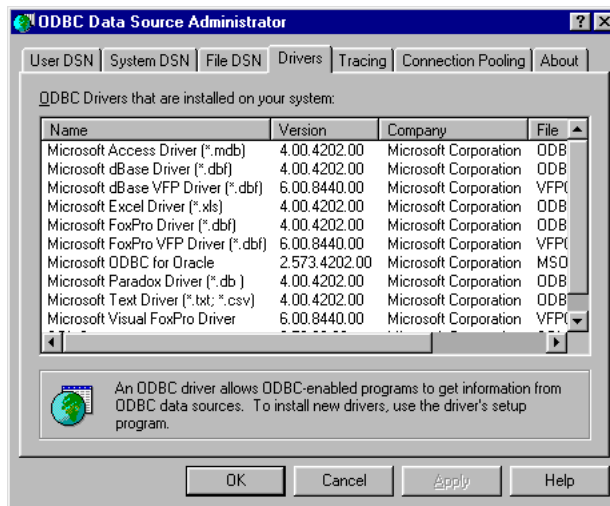
To view the ODBC drivers installed on a Windows system:

1 Open the ODBC Data Source Administrator as follows:

- In Windows 95, 98, or NT, choose Start > Settings > Control Panel, then double-click the ODBC Data Sources icon. Depending on your system, the icon could also be called ODBC or 32bit ODBC.
- In Windows 2000, choose Start > Settings > Control Panel > Administrative Tools > Data Sources.

2 Click the Drivers tab.

A list of ODBC drivers installed on the system appears.



## Typical system configurations

Here are typical system configurations for ASP developers:

Database	Database driver	App server	Web server
MS Access	Microsoft Access Driver (ODBC)	PWS IIS	PWS IIS
MS SQL Server	Microsoft SQL Server Driver (ODBC) Microsoft SQL Server Provider (OLE DB)	IIS Chili!Soft ASP	IIS Any Chili!Soft-compatible server
Oracle	Microsoft Oracle Driver (ODBC)	IIS Chili!Soft ASP	IIS Any Chili!Soft-compatible server

If you're using UltraDev for Windows with Microsoft Access databases, you can configure your local computer as your server. This approach lets you develop and test your ASP sites locally before deploying them on a remote server. For instructions on installing PWS on your local computer, see "Installing Microsoft Personal Web Server" on page 243. If you're a Windows 2000 user, you can install IIS 5.0, which is included in the Windows 2000 package.

## Requirements for ColdFusion developers

To develop ColdFusion applications in UltraDev, you'll need a Web server, Allaire ColdFusion Server, a database, and a driver for your database.

### Web server

You need access to a Web server to host your ColdFusion site. You can use any Web server that works with ColdFusion Server.

If you're a Windows 98 or NT Workstation user, you can install and run Microsoft Personal Web Server (PWS) on your local computer. For more information, see "Installing Microsoft Personal Web Server" on page 243. If you're a Windows 2000 user, you can install and run Microsoft enterprise-strength Web server, Internet Information Server (IIS) 5.0, on your local computer. (IIS is included in the Windows 2000 package.) ColdFusion Server works well with these two Web servers.

### ColdFusion application server

An application server is software that processes dynamic pages before the pages are served up to requesting browsers by the Web server.

To develop ColdFusion pages in UltraDev, you need Allaire ColdFusion Server 4.0 or 4.5.

If your Web server runs on a Windows, Linux, or Solaris system, you can download an evaluation copy of ColdFusion Server from the Allaire Web site at <http://www.allaire.com/download/index.cfm>. (You need to register if you're not already registered.)

If you're a Windows user, you can turn your local computer into a ColdFusion development platform by installing PWS (or IIS) along with the single-user copy of ColdFusion Server included on the UltraDev CD. Once installed, PWS functions as your Web server while ColdFusion Server functions as your application server. For more information, see "Installing Microsoft Personal Web Server" on page 243, and Appendix D, "Installing Allaire ColdFusion Server" on page 245.

## Database

You can use almost any database with your Web application, so long as you have the appropriate database driver for it.

If you plan to build small, low-cost applications, you can use a file-based database, such as one created in Microsoft Access. If you plan to build robust, business-critical applications, you can use a server-based database, such as one created in Microsoft SQL Server, Oracle 8i, or IBM DB2.

If your database is located on a system other than your Web server, make sure you have a fast connection between the two systems so that your Web application can operate quickly and efficiently.

## Database driver

A database driver acts like a kind of translator between your ColdFusion application and your database. Data stored in a database is normally in a proprietary format, just as text in a word-processor file is in a proprietary format. A database driver lets your ColdFusion application read and manipulate data that would otherwise be undecipherable.

The appropriate database driver for your database depends on your application and your database. ColdFusion applications “speak” both ODBC and OLE DB; therefore, you need either an ODBC or OLE DB database driver. To learn more about ODBC and OLE DB drivers, see “Interfacing with the database” on page 227.

The driver must also be specific to your database. If you use a Microsoft Access database, your server must have an ODBC or OLE DB driver for Microsoft Access. If you use an Oracle database, your server must have an ODBC or OLE DB driver for Oracle. ColdFusion also provides native drivers for Oracle databases.

**Note:** There is currently no Microsoft Access ODBC driver or OLE DB provider for the UNIX version of ColdFusion.

Microsoft offers a number of ODBC drivers for the most popular database packages, such as Microsoft Access, Microsoft SQL Server, and Oracle. The drivers, which only run on the Windows platform, are automatically installed with Microsoft Office. They are also provided with the Microsoft Data Access Components (MDAC) 2.5 package, which you can download for free from the Microsoft Web site at <http://www.microsoft.com/data/download.htm>.

If you need a specific ODBC driver and your Web server runs on a Windows system, you can easily find out whether or not the ODBC driver you need is installed on the system.

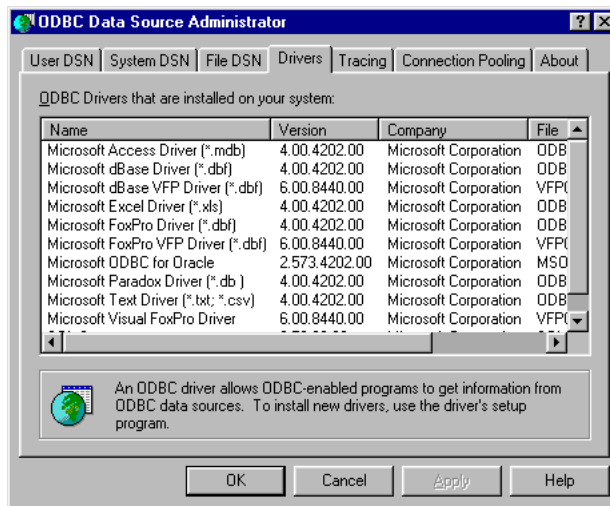
To view the ODBC drivers installed on a Windows system:

1 Open the ODBC Data Source Administrator as follows:

- In Windows 95, 98, or NT, choose Start > Settings > Control Panel, then double-click the ODBC Data Sources icon. Depending on your system, the icon could also be called ODBC or 32bit ODBC.
- In Windows 2000, choose Start > Settings > Control Panel > Administrative Tools > Data Sources.

2 Click the Drivers tab.

A list of ODBC drivers installed on the system appears.



## Typical system configurations

Here are typical system configurations for ColdFusion sites:

Database	Database driver	App server	Web server
MS Access	Microsoft Access Driver (ODBC)	ColdFusion Server	PWS IIS
MS SQL Server	Microsoft SQL Server Driver (ODBC) Microsoft SQL Server Provider (OLE DB)	ColdFusion Server	IIS
Oracle	Microsoft Oracle Driver (ODBC) ColdFusion native driver for Oracle	ColdFusion Server	IIS Any HTTP server for the Windows, Linux, or Solaris platform

If you're using UltraDev for Windows with Microsoft Access databases, you can configure your local computer as your server. This approach lets you develop and test your ColdFusion sites locally before deploying them on a remote server. For instructions on installing PWS on your local computer, see "Installing Microsoft Personal Web Server" on page 243. For instructions on installing the single-user copy of ColdFusion Server on the UltraDev CD, see "Installing Allaire ColdFusion Server" on page 245.

## Requirements for JSP developers

To develop JSP applications in UltraDev, you'll need a Web server, a JSP application server, a database, and a JDBC driver for your database.

### Web server

You need access to a Web server to host your JSP site. You can use any Web server that's compatible with your chosen JSP application server.



## JSP application server

An application server is software that processes dynamic pages before the pages are served up to requesting browsers by the Web server.

To develop JSP pages in UltraDev, you need an application server that supports Sun JavaServer Pages 1.0 specification. Here are a few examples:

- IBM WebSphere 3.0 or 3.5 with the operating system and hardware to run it
- Allaire JRun 3.0 with the operating system and hardware to run it
- Apache Tomcat 3.1 with the operating system and hardware to run it

If your Web server is IIS, you can install and run IBM WebSphere Application Server, which is included in the UltraDev package. (The CD also includes a Web server if you prefer not to use IIS.)

If you're a Windows user, you can turn your local computer into a JSP development platform by installing PWS (or IIS) along with the copy of Allaire JRun included on the UltraDev CD. Once installed, PWS functions as your Web server while JRun functions as your application server. For more information, see "Installing Microsoft Personal Web Server" on page 243.

If your Web server runs on a Linux, Solaris, or UNIX system, you can download a development, evaluation, or commercial copy of JRun from the Allaire Web site at <http://www.allaire.com/download/index.cfm>. (You need to register if you're not already registered.)

You can download a copy of Apache Tomcat from the Jakarta Project Web site at <http://jakarta.apache.org/tomcat/>.

## Database

You can use almost any database with your Web application, so long as you have the appropriate JDBC database driver for it.

If you plan to build small, low-cost applications, you can use a file-based database, such as one created in Microsoft Access. If you plan to build robust, business-critical applications, you can use a server-based database, such as one created in Microsoft SQL Server, Oracle 8i, or IBM DB2.

If your database is located on a system other than your Web server, make sure you have a fast connection between the two systems so that your Web application can operate quickly and efficiently.

## JDBC database driver

A database driver acts like a kind of translator between your JSP application and your database. Data stored in a database is normally in a proprietary format, just as text in a word-processor file is in a proprietary format. A database driver lets your JSP application read and manipulate data that would otherwise be undecipherable.

The appropriate database driver for your database depends on your application and your database. JSP applications “speak” JDBC; therefore, you need a JDBC database driver. To learn more about JDBC drivers, see “Interfacing with the database” on page 227. You can also use an ODBC driver if you have a JDBC-ODBC bridge driver. A JDBC-ODBC bridge driver is software that turns your JDBC-speaking application into an ODBC-speaking one.

The driver must also be specific to your database. For example, if you use a Microsoft SQL Server database, your server must have a JDBC driver for Microsoft SQL Server. One such driver is the I-net JDBC driver available from i-net software at <http://www.inetsoftware.de/>. If you have a JDBC-ODBC bridge driver, you can use an ODBC driver for Microsoft SQL Server instead.

If you use either a JDBC driver or a JDBC-ODBC bridge driver to connect to your database, make sure the Java Development Kit (JDK) is installed on your server. You can download the JDK from the Sun Web site at <http://java.sun.com/products/jdk/1.1/>.

## Typical system configurations

Here are typical system configurations for JSP sites:

Database	Database driver	App server	Web server
MS Access	Any JDBC driver for Access A JDBC-ODBC bridge driver with the Microsoft Access Driver (ODBC)	WebSphere JRun Tomcat	Microsoft IIS IBM HTTP Server Apache HTTP Server
MS SQL Server	Any JDBC driver for SQL Server A JDBC-ODBC bridge driver with the Microsoft SQL Server Driver (ODBC)	WebSphere JRun Tomcat	Microsoft IIS IBM HTTP Server Apache HTTP Server
Oracle	Any JDBC driver for Oracle	WebSphere JRun Tomcat	Microsoft IIS IBM HTTP Server Apache HTTP Server

## APPENDIX C

### Installing Microsoft Personal Web Server

---

This appendix is intended for Windows users interested in developing and testing ASP or ColdFusion applications locally. It provides basic instructions on installing and configuring Microsoft Personal Web Server (PWS), which doubles as an ASP application server. You can also install a ColdFusion application server on the same system to work with PWS. For instructions, see “Installing Allaire ColdFusion Server” on page 245.

PWS, which runs in Windows 95, 98, and NT Workstation, is a scaled-down version of Microsoft enterprise-strength Internet Information Server (IIS). (You don't need PWS if you're a Windows 2000 user because that operating system includes IIS.)

**Note:** Macromedia does not provide technical support for third-party software such as Microsoft Personal Web Server. If you need assistance, contact Microsoft technical support.

### Installing PWS

You can install Personal Web Server on the same Windows system that runs Dreamweaver UltraDev. If you're a Windows 98 user, you'll find a copy of PWS in the Add-Ons/PWS folder on your Windows 98 CD. If you're a Windows 95 or NT Workstation user, you can download PWS from the Microsoft Web site at <http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp>. (If you're a Windows NT Server or Windows 2000 user, you can use the copy of IIS included with your operating system.)

Before installing PWS, make sure Microsoft Internet Explorer 4.01 or later is installed on your system: PWS won't install without it.

**To install PWS on your system:**

- 1 Double-click the PWS installation file on the Windows 98 CD, or the file you downloaded from the Microsoft Web site.
- 2 Follow the installation wizard.
- 3 When asked your default Web publishing home directory, accept the default directory:  
`C:\Inetpub\wwwroot`
- 4 Click Finish to end the installation process.

## Configuring PWS

Once PWS is installed, you can use it to run ASP applications.

**To configure PWS to run an ASP application:**

- 1 In Windows, create a subdirectory in the C:\Inetpub\wwwroot\ directory.  
  
Alternatively, you can create a directory anywhere else on your local drive and define it as a virtual directory in PWS.  
  
To define a virtual directory, start Microsoft Personal Web Manager and click the Advanced icon. The Advanced Options dialog box appears. Click Add, then click Browse and select the directory you created. Enter an alias for the directory (an alias is a stand-in for the path to the directory), then click OK to create the virtual directory.
- 2 Make sure the Read and Scripts permissions are enabled for the directory.  
  
In Microsoft Personal Web Manager, click the Advanced icon. The Advanced Options dialog box appears. Select the directory and click Edit Properties. The Edit Directory dialog box appears. Make sure the Read and Scripts options are selected.

PWS is now configured to run an ASP application in the directory you created. For more information, see “Configuring your system” on page 22.

## APPENDIX D

### Installing Allaire ColdFusion Server

---

This appendix is intended for Windows users interested in developing and testing ColdFusion applications locally. It provides basic instructions for installing and configuring the single-user copy of Allaire ColdFusion Server included on the Dreamweaver UltraDev CD. This copy is a scaled-down version of Allaire enterprise-strength application server of the same name.

You can also download an evaluation copy of ColdFusion Server from the Allaire Web site at <http://www.allaire.com/download/index.cfm>. (You need to register if you're not already registered.)

Because ColdFusion Server is an application server that works in tandem with a Web server, a Web server such as Microsoft Personal Web Server (PWS) or Internet Information Server (IIS) must be installed on your local computer. For instructions on installing PWS, see "Installing Microsoft Personal Web Server" on page 243.

**Note:** Macromedia does not provide technical support for third-party software such as Allaire ColdFusion Server. If you need assistance, contact Allaire technical support.

# Installing and configuring ColdFusion Server

Before you begin, make sure PWS or IIS is installed on your local Windows system.

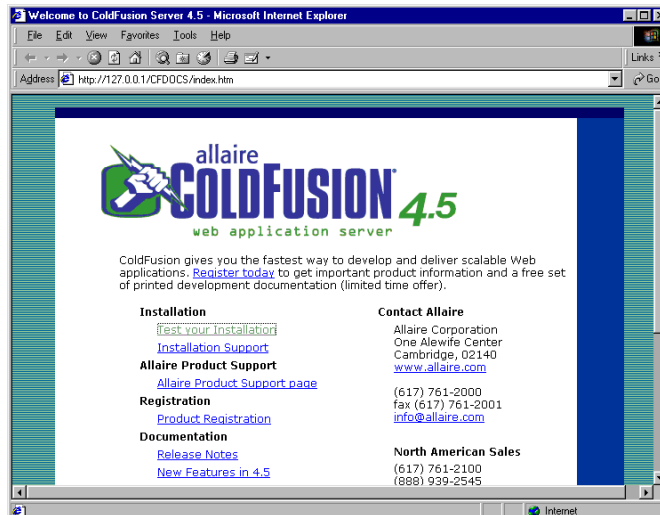
To install and configure ColdFusion Server on your local Windows system:

- 1 Double-click the ColdFusion.exe file in the ColdFusion Server folder on the UltraDev CD, or the file you downloaded from the Allaire Web site.
- 2 Follow the onscreen instructions to install the program.

**Note:** The serial number should be located on the UltraDev CD sleeve.

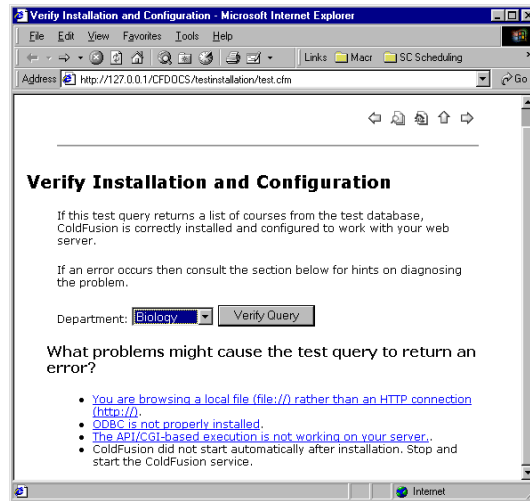
- 3 When the installation is finished, restart your computer.

When it restarts, the computer launches your Web browser and displays the ColdFusion welcome page.



- 4 Click Test Your Installation to verify that ColdFusion is correctly installed and configured to work with your Web server.

The verification page opens.



- 5 Follow the onscreen instructions to test your installation.

Once you're finished, you can use your local Web server to publish and test ColdFusion pages created in UltraDev.





## APPENDIX E

### Setting Up a DSN in Windows

---

This appendix applies only if your database is located on a system that supports ODBC data source names (DSNs)—systems such as Microsoft Windows and Windows NT (but not the Macintosh).

A DSN is a kind of shortcut you use to establish a database connection (see “Using a DSN in a connection string” on page 232). Before you can use one in your Web application, you must set it up on your local system or remote server.

Before you begin, make sure your system has the proper driver for your database. For a list of ODBC drivers on a Windows 95, 98, or NT system, choose Start > Settings > Control Panel, then double-click the ODBC Data Sources icon. (Depending on your system, the icon could also be called ODBC or 32bit ODBC.) When you click the Drivers tab, you’ll see a list of drivers installed on the system. In Windows 2000, choose Start > Settings > Control Panel > Administrative Tools > Data Sources, then click the Drivers tab.

**To set up a DSN in Windows:**

- 1** Open Windows' ODBC Data Source Administrator as follows:
  - In Windows 95, 98, or NT, choose Start > Settings > Control Panel, then double-click the ODBC Data Sources icon. Depending on your system, the icon could also be called ODBC or 32bit ODBC.
  - In Windows 2000, choose Start > Settings > Control Panel > Administrative Tools > Data Sources.
  - In the Dreamweaver UltraDev dialog box you use to create a DSN connection, click the Define button.
- 2** In the ODBC Data Source Administrator dialog box, click the System DSN tab.

The tab displays the list of DSNs currently on your system.
- 3** Click Add to add a new DSN to the list.

The Create New Data Source dialog box appears, listing all the drivers currently loaded on your system.
- 4** Select a driver from the list, then click Finish.

For example, if your database is a Microsoft Access file, select Microsoft Access Driver (\*.mdb). If a driver for your product does not appear in the list, you'll have to download the driver from a vendor's Web site and install it. For more information, see "Using database drivers to interface with your database" on page 229.
- 5** In the dialog box that appears, enter a name for the DSN and specify the connection parameters.

The dialog boxes for specifying parameters differ depending on the driver you selected. For the Microsoft Access Driver, you enter a name, click Select, locate the database file on the hard disk, and click OK.
- 6** Click OK to close the dialog box.

The new DSN is added to your list of system DSNs.

## APPENDIX F

### SQL Primer

.....

This appendix presents a short primer on writing simple SQL queries to create recordsets.

The most common SQL statement for creating a recordset is the `SELECT` statement, which extracts specified columns from one or more database tables to build the recordset. Here's the basic syntax for a `SELECT` statement:

```
SELECT ColumnName FROM TableName
```

You can add line breaks, tabs, and other white space to your statements to clarify the logic: SQL ignores all white space. For example, the following is a valid statement:

```
SELECT PaidDues  
FROM Members
```

**Note:** Macromedia does not provide technical support for third-party technologies such as SQL.

### Including an entire table

To include the full contents of a table in your recordset, use the asterisk (\*) as a wildcard character to include all the columns in the table. For example, suppose you have a table called `Customers`. To extract all the columns, you would type the following `SELECT` statement:

```
SELECT * FROM Customers
```

## Limiting the number of columns

Suppose you only need the data contained in two columns of the `Customers` table: the `YearBorn` column and the `DateLastPurchase` column. To create a recordset containing only the data from these two columns, you would type the following `SELECT` statement:

```
SELECT YearBorn, DateLastPurchase FROM Customers
```

## Limiting the number of records

Use a `WHERE` clause to limit the number of records in the recordset. For example, you may want to include only those customers who earn more than \$50,000 a year. Assume you have a column in your table called `Earnings` that tells you how much each customer earns. Your `SELECT` statement would read as follows:

```
SELECT YearBorn, DateLastPurchase FROM Customers  
WHERE Earnings > 50000
```

## Specifying a condition in the WHERE clause

You specify a condition in a `WHERE` clause to limit the number of records in a recordset. Here's a list of conditional operators you can use:

Operator	Meaning
=	Equal to (case sensitive)
LIKE	Equal to (case insensitive)
<>	Not equal to (case sensitive)
NOT LIKE	Not equal to (case insensitive)
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

If the item being compared is text, place it in single quotes as in this example:

```
...WHERE Country = 'Germany'
```

If the item being compared is a date, and you're working with a Microsoft Access database, enclose it with `#` signs:

```
...WHERE DateOfBirth < #01/01/1970#
```

Other databases have their own date conventions. Consult the database vendor's documentation.

You can use wildcard characters in conditional expressions. The standard wildcard character is the percentage sign (%):

```
...WHERE LastName LIKE 'Mc%'
```

For an Access database, the asterisk (\*) also works as a wildcard character:

```
...WHERE CompanyName = '*soft'
```

## Specifying more than one condition in the WHERE clause

You can specify more than one condition in the WHERE clause to limit the number of records in your recordset. You combine the conditions using the AND, OR, and NOT logical operators.

If you want all the conditions to be true for a record to be included in the recordset, use the AND operator:

```
...WHERE Country = 'Germany' AND Car = 'Ford'
```

If you want any one of the conditions to be true for a record to be included in the recordset, use the OR operator:

```
...WHERE Country = 'Germany' OR Country = 'Hungary'
```

If you want one condition to be true but not another, use the NOT operator:

```
...WHERE Country = 'Germany' NOT Car = 'BMW'
```

You can use parentheses to group clauses:

```
...WHERE (Country = 'Germany' AND DateOfBirth < #01/01/1970#) OR ¬  
Country = 'Hungary'
```

## Specifying a range of values in the WHERE clause

You can specify a range of values in the WHERE clause to limit the number of records in your recordset. You specify the range using the BETWEEN...AND keywords.

For example, suppose you want to include all employees born between January 1, 1960 and December 31, 1974. Your WHERE clause might look as follows:

```
...WHERE DateOfBirth BETWEEN #01/01/1960# AND #12/31/1974#
```

## Sorting the records

Use the `ORDER BY` clause to sort the records in your recordset. For example, suppose you want to sort the records in the recordset by customer earnings, from the lowest to the highest. In the SQL statement you would order the records as follows:

```
SELECT LastName, FirstName, Earnings FROM Customers ↵  
ORDER BY Earnings
```

By default, the `ORDER BY` clause sorts records in ascending order (1, 2, 3... or A, B, C...). If you want to sort them in descending order, from the highest earnings to the lowest, you would use the `DESC` keyword as follows:

```
ORDER BY Earnings DESC
```

# INDEX

## A

- access privileges
  - adding to pages 208
  - example 207
  - storing in a database 209
- Active Server Pages (ASP)
  - application servers 234
  - database connections 89
  - documentation 110
  - request objects 110
  - requirements 233
  - selecting a server model 25
  - typical configurations 236
- ActiveX objects, making dynamic 144
- adding dynamic content 133
- ADO wrapper 229
- advanced options, Server Behavior Builder 219
- advanced Recordset dialog box
  - Database Items tree 129
  - entering SQL 129
  - using 128
- Allaire's ColdFusion Server 245
- Application Server category 25
- application servers
  - ASP 234
  - JSP 241
  - requirements 8
  - specifying in UltraDev 26
- application variables 118
- attributes, making dynamic 142
- authorization levels 207
- Auto Refresh 84

## C

- caching data sources 123
- Callable (Stored Procedure) option 119
- catalogs 107
- checkboxes, making dynamic 139
- code
  - editing 87
  - other text editors 88

- code blocks
  - coding guidelines 219
  - parameter markers 216
  - positioning 215
  - writing 213
- Code inspector 87
- Code view 87
- coding guidelines 219
- ColdFusion
  - client variables 113
  - database connections 98
  - documentation 110
  - form variables 113
  - requirements 237
  - selecting a server model 25
  - typical configurations 240
  - URL variables 113
- ColdFusion DSN 98
- ColdFusion Server 245
- Command (Stored Procedure) option 119
- configurations
  - ASP 236
  - ColdFusion 240
  - general procedures 22
  - JSP 242
  - quick start for Macintosh 16
  - quick start for Windows 11
- connection strings 97
- Connections command 89
- copying and pasting recordsets 131

## D

- Data Bindings panel
  - adding data sources 109
  - adding dynamic text 134
  - creating a record counter 155
  - defining a recordset 126
  - deleting data sources 123
  - Format column 136
  - making forms dynamic 138
  - making HTML attributes dynamic 142
  - viewing recordset columns 127

- data formats
  - applying 136
  - editing and creating 211
- data source name
  - setting up 249
- data sources
  - adding 109
  - application variables 118
  - ASP variables 110
  - caching 123
  - ColdFusion variables 113
  - deleting 123
  - JavaBeans 120
  - JSP variables 117
  - recordsets 110
  - session variables 117
  - stored procedure objects 119
- database connections
  - ASP 89
  - at design time 105
  - basics 227
  - ColdFusion 98
  - connection strings 97
  - editing or deleting 104
  - JDBC 101
  - JSP 101
  - ODBC 89
  - OLE DB 94
  - remote database connectivity 105
- database drivers
  - basics 227
  - JDBC 101
  - JDBC parameters 102
  - ODBC 89
  - OLE DB 94
  - requirements 8
  - viewing installed drivers 236
- Database Items tree, using 129
- databases
  - basics 225
  - requirements 8
  - schemas and catalogs 107
- delete page, building 193
- Delete Record behavior 196
- deleting a recordset 132
- deleting dynamic content 145

- design time connections 105
- detail pages
  - creating link to 173
  - find specified record 163, 175
- Discussion Group 33
- displaying multiple records 151
- Document window 81
- DSN
  - ColdFusion 98
  - ODBC 90
- DSN-less connections 92
- Dynamic Checkbox dialog box 139
- dynamic content
  - adding 133
  - attributes 142
  - forms 138
  - images 136
  - objects 144
  - removing 145
  - replacing 144
  - text 134
  - text placeholders 81
- Dynamic Data dialog box 143
- Dynamic List/Menu dialog box 141
- dynamic pages
  - basics 76
  - defined 75
- Dynamic Radio Buttons dialog box 140

## E

- Edit Format List dialog box 211
- editing
  - code 87
  - recordsets 132
  - server behaviors 221
- extensibility 32
- extensions
  - creating 212
  - installing 212

## F

- Filter 127
- finding a specific record 161, 175
- Flash objects, making dynamic 144
- Format column 136



## forms

- making checkboxes dynamic 139
- making image fields dynamic 139
- making radio buttons dynamic 140
- making text fields dynamic 138
- using to gather data 166

## G

- gathering data from users 166
- Generator objects, making dynamic 144
- Go to Detail Page behavior 162, 173
- Go to Related Page behavior 178
- going to a detail page 173
- going to a related page 178
- Guided Tour 30

## H

- Help systems 30
- hiding links 151
- home directory 28
- HTML attributes, making dynamic 142
- HTML forms. *See* forms

## I

- images, making dynamic 136
- insert page, building 182
- Insert Record behavior 186
- installing UltraDev 10

## J

- Java applets, making dynamic 144
- JavaServer Pages (JSP)
  - application servers 241
  - database connections 101
  - documentation 110
  - JavaBeans 120
  - requirements 240
  - resultset 77
  - selecting a server model 25
  - typical configurations 242
- JDBC
  - connection parameters 102
  - drivers 101
- JSP. *See* JavaServer Pages

## L

- learning UltraDev 33
- Lessons 32
- links, hiding 151
- List view in Property inspector 143
- list/menu objects, making dynamic 141
- Live Data Settings dialog box 85
- Live Data window
  - Auto Refresh 84
  - described 82
  - how it works 83
  - missing files 84
  - providing expected parameters 85
  - removing content highlight 84
  - URL parameters on toolbar 85
  - using 84
- live objects
  - defined 79
  - Master/Detail Page Set 158
  - Record Insertion Form 182
  - Record Update Form 188
  - Recordset Navigation Bar 148
  - Recordset Navigation Status 154
- Local Info category 23
- local site, defining 23
- localhost 28
- logging out users 210
- log-in pages 204

## M

- Macromedia Exchange 212
- maintaining state information 178
- Master/Detail Page Set live object 158
- master/detail pages 156, 173
- Microsoft Personal Web Server 243
- Move to Record behavior 150
- Move to Specific Record behavior 175

## N

- navigation links for records 147
- new features 29

## O

objects, making dynamic 144

ODBC

- drivers 89

- viewing installed drivers 236

OLE DB 94

## P

pages

- delete 193

- detail 156, 173

- insert 182

- log-in 204

- master 156

- related 178

- restricting access to 207

- results 167

- search 166

- update 187

- user registration 200

parameter markers 216

Parameters dialog box 144

passwords

- checking during log-in 206

- letting users choose 201

- storing 200

placeholders 81

placeholders for dynamic text 135

plug-ins, making dynamic 144

positioning code blocks 215

Preview in Browser 86

Property inspector

- editing a recordset 145

- List view 143

- making HTML attributes dynamic 143

- Standard view 143

PWS 243

## Q

Quick Tag Editor 88

## R

radio buttons, making dynamic 140

record counter, building 153

record editing behaviors 181

Record Insertion Form live object 182

record navigation bar

- creating 147

- hiding 151

Record Update Form live object 188

records 225

- building a counter 153

- deleting 193

- displaying more than one 151

- inserting 182

- navigation links 147

- updating 187

recordset

- basics 125

- caching 123

- copying and pasting 131

- defining with SQL 128

- defining without SQL 126

- editing or deleting 132

- in a detail page 163

- in a results page 167

- limiting the number of records 127

- using SQL 251

Recordset (Query) option 126

Recordset dialog box

- advanced 128

- simple 126

Recordset Navigation Bar live object 148

Recordset Navigation Status live object 154

regions, hiding 151

registration page 200

related pages 178

Remote Info category 23

remote site, defining 23

Repeat Region behavior 151

requirements

- application server 8

- database 8

- database driver 8

- Web applications 233

- Web server 8

restricting site access 199

restricting tables 107

results pages

- going to a detail page 173

- using a simple recordset 168

- using an advanced recordset 170

resultset, JSP 77

## S

- schemas 107
- scripting languages 25
- search pages 166
- security 199
- Server Behavior Builder 212
- server behaviors
  - coding guidelines 219
  - creating 212
  - creating dialog boxes for 216
  - defined 75
  - deleting records 196
  - editing custom behaviors 222
  - editing on a page 164
  - editing records 181
  - editing UltraDev behaviors 223
  - going to a detail page 162, 173
  - going to a related page 178
  - hiding regions 151
  - inserting records 186
  - installing more 212
  - moving to a specific record 175
  - moving to records 150
  - repeating regions 151
  - testing 221
  - updating records 192
- server model *See* server technology
- server objects
  - application objects 118
  - ASP request objects 110
  - ColdFusion variables 113
  - session objects 117
  - stored procedure objects 119
- server technology
  - defined 75
  - specifying 25
- server-side scripts 76
- session variables 117
- Shockwave objects, making dynamic 144
- Show Region behavior 151
- simple Recordset dialog box 126
- site security 199
- SQL 251
  - sample statements 129
  - using variables 129

- Standard view in Property inspector 143
- Stored Procedure (ColdFusion) option 119
- stored procedures
  - creating a stored procedure object 119
  - using to define a recordset 131
- Support centers 32
- system requirements
  - Macintosh 11
  - Windows 10

## T

- tables
  - basics 226
  - restricting 107
- testing server behaviors 221
- text
  - formatting 134
  - making dynamic 134
- Tutorial 35
- typographical conventions 7

## U

- UltraDev
  - data sources 109
  - database connections 105
  - workflow 77
- update page, building 187
- Update Record behavior 192
- updating records 187
- URL prefix
  - explained 28
  - specifying in UltraDev 27
- user names
  - checking during log-in 206
  - checking for uniqueness 203
  - letting users choose 201
  - storing 200

## V

- views 87
- virtual directory 28

## W

- Web application
  - common pages 77
- Web applications
  - defined 75
  - other useful resources 34
  - requirements 233
  - workflow 77
- Web servers
  - configuring 22
  - requirements 8
- workflow 77
- working environments
  - Code inspector 87
  - Code view 87
  - Document window 81
  - Live Data window 82
  - other text editors 88
- writing code blocks 213