14 STORING USER RECORDS IN A DATABASE

Add a new User

Login Information

User name:	Use text field	cs4admin
Host:	Local	localhost
Password:	Use text field	••••••
Re-type:		••••••
Generate Password:	Generate Copy	

Registered Users - Mozilla Firefox

http://dw/cs4/workfiles/ch14/list_users.php?

Registered Users

Register new user

Name	Username	Administrator
Tom Muck	tmuck	n
David Powers	dpowers	y

Radio Group Properties

Radio items:

Label	Value
Yes	y
No	n

Label: No

Value: n

Select value equal to: <?php echo htmlentities($row_getUser['ad

Dynamic websites take on a whole new meaning in combination with a database. Drawing content from a database allows you to present material in ways that would be impractical—if not impossible—with a static website. Examples that spring to mind are online stores, such as Amazon.com; news sites, such as the International Herald Tribune (`http://www.iht.com`); and the big search engines, including Google and Yahoo!. Database technology allows these websites to present thousands, sometimes millions, of unique pages with remarkably little underlying code. Even if your ambitions are nowhere near as grandiose, a database can increase your website's richness of content with relatively little effort.

Although PHP is capable of interacting with most popular databases (and some less well-known ones, too), Dreamweaver has made the choice for you. All the server behaviors are designed to work with MySQL. In one respect, this is a good choice, because it's widely available, free, and very fast and it offers an excellent range of features. The downside is that the server behaviors work *only* with MySQL. If you want to use a different database, such as PostgreSQL (`http://www.postgresql.org/`), SQLite (`http://www.sqlite.org/`), or Microsoft Access (`http://office.microsoft.com/access`), you have to do all the coding by hand.

Although Dreamweaver does a lot of the hard work for you when building a database-driven website, it's important to remember that you're combining several technologies. So, there's a lot to learn. A big mistake that most beginners make is to rush headlong into creating a database and cram it full of data without understanding how databases work (I know, I did it myself many years ago). I'll try not to overburden you with too much heavy theory, but this chapter starts with some of the basic knowledge that you'll need to start working with a MySQL database.

In this chapter, you'll learn about the following:

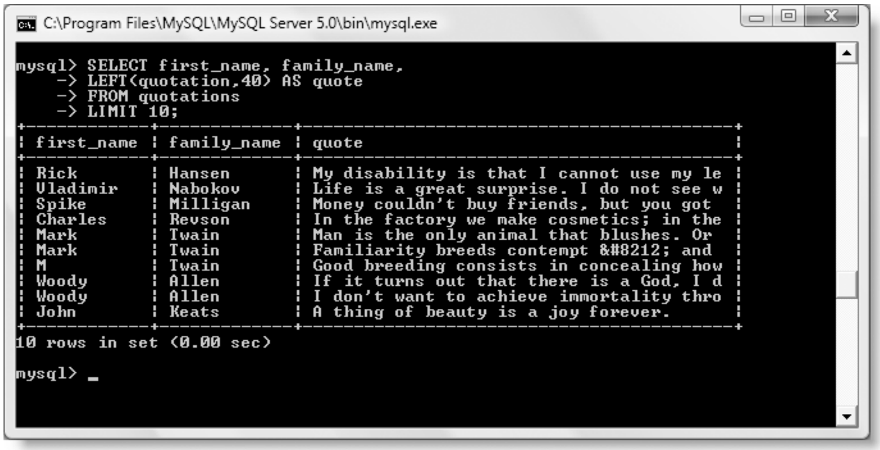
- Creating MySQL user accounts
- Defining a database table through the phpMyAdmin graphical interface
- Choosing the appropriate data types for database columns
- Using Dreamweaver server behaviors to insert, update, and delete records
- Creating a simple user registration system

I assume you already have access to a MySQL database, preferably in a local testing environment. The current stable version of MySQL is 5.0, but MySQL 5.1 should be released around the time this book is published. Dreamweaver is compatible with MySQL as far back as MySQL 3.23, but you should ideally be using a minimum of MySQL 4.1, because older versions do not support UTF-8 encoding.

Introducing MySQL

If you have ever worked with Microsoft Access, your first encounter with MySQL might come as something of a shock. For one thing, it doesn't have a glossy interface. As

Figure 14-1 shows, it looks like a throwback to the old days of DOS before the friendly interfaces of Mac and Windows. Its beauty lies, however, in its simplicity. What's more, most of the time you'll never see MySQL in its raw state like this. You'll use either Dreamweaver or a graphic front end. Several graphic front ends for MySQL are available—some free, others commercial products. The one I'll be using in this book is a free application called phpMyAdmin (`http://www.phpmyadmin.net/`). Best of all, you'll be designing your own personalized interface by creating PHP pages.



```
mysql> SELECT first_name, family_name,
    -> LEFT(quotation,40) AS quote
    -> FROM quotations
    -> LIMIT 10;
+------------+-------------+------------------------------------------+
| first_name | family_name | quote                                    |
+------------+-------------+------------------------------------------+
| Rick       | Hansen      | My disability is that I cannot use my le |
| Vladimir   | Nabokov     | Life is a great surprise. I do not see w |
| Spike      | Milligan    | Money couldn't buy friends, but you got  |
| Charles    | Revson      | In the factory we make cosmetics; in the |
| Mark       | Twain       | Man is the only animal that blushes. Or  |
| Mark       | Twain       | Familiarity breeds contempt &#8212; and  |
| M          | Twain       | Good breeding consists in concealing how |
| Woody      | Allen       | If it turns out that there is a God, I d |
| Woody      | Allen       | I don't want to achieve immortality thro |
| John       | Keats       | A thing of beauty is a joy forever.      |
+------------+-------------+------------------------------------------+
10 rows in set (0.00 sec)

mysql> _
```

Figure 14-1. The unadorned interface of MySQL in a Windows Command Prompt window

The other thing that comes as a surprise to Access users is that your database is not kept in a single file that you can upload to your remote server. MySQL keeps all databases in a central data folder, and each database table normally consists of three separate files. The way you transfer data from one server to another is by creating a text file that contains all the necessary commands to build the database and its contents—in other words, a backup file. All you need to know now is that there isn't "a database file"—there are lots of them, and normally, you should never handle them directly.

Understanding basic MySQL terminology

If you haven't worked with a relational database before, you may find your head spinning with some of the names that crop up throughout the rest of this book. So, here's a quick guide:

- **SQL**: Structured Query Language is the international standard behind all major relational databases. It's used to insert and otherwise manipulate data and is based on natural English. For instance, let's say you have a database table called `members` that stores each person's details as `first_name`, `family_name`, and `username`. You would use the following command (or SQL query) to find the `first_name` and `family_name` for the person whose `username` is `dpowers`:

```
SELECT first_name, family_name
FROM members
WHERE username = 'dpowers'
```

As you can see, it's very human-readable, unlike many other computer languages. Although SQL is a standard, all of the main databases have added enhancements on top of the basic language. If you have been using another database, such as Access or Microsoft SQL Server, be prepared for some slight differences in the use of functions. Some people pronounce SQL "sequel," while others say "Ess-queue-ell." Both are right.

- **MySQL**: This refers to the entire database system created by MySQL AB, originally a Swedish company but now part of Sun Microsystems (`http://www.sun.com`). It's always spelled in uppercase, except for the "y," and the official pronunciation is "My-ess-queue-ell." It's not just a single program, but a client/server system with a number of related programs that perform various administrative tasks. The two main components are `mysql` and `mysqld`, with both terms entirely in lowercase.
- `mysql`: This has three distinct meanings. The first is the client program used to feed requests to the database. `mysql` is also the name of the main administrative database that controls user accounts, and on Windows, it is the name of the Windows service that starts and stops the database server. Once you start working with MySQL, differentiating between the different meanings of "`mysql`" is not as confusing as it first seems.

Using MySQL with a graphic interface

Although you can use MySQL in a Windows Command Prompt window or Mac Terminal, it's a lot easier to use a graphic interface. There are several to choose from, both commercial and free. Among the free offerings are two from MySQL: MySQL Administrator and MySQL Query Browser (`http://www.mysql.com/products/tools`). Three other popular graphical front ends for MySQL are Navicat (`http://www.navicat.com`), a commercial product, and DBTools Manager (`http://www.dbtools.com.br/EN/dbmanagerpro/`) and SQLyog (`http://www.webyog.com`), which are available in both commercial and free versions.

However, the most popular graphical interface for MySQL is phpMyAdmin (`http://www.phpmyadmin.net`). It's a PHP-based administrative system for MySQL that has been around since 1998, and it constantly evolves to keep pace with MySQL developments. It works on Windows, Mac OS X, and Linux. What's more, many hosting companies provide it as the standard interface to MySQL. For that reason, I plan to use phpMyAdmin throughout the rest of this book.

If you installed XAMPP or MAMP, phpMyAdmin is already installed on your local computer. However, for the benefit of readers who need to install phpMyAdmin, the next section describes the process.

Setting up phpMyAdmin on Windows and Mac

Like a lot of open source applications, phpMyAdmin is constantly evolving. At the time of this writing, a major rewrite of the application, phpMyAdmin 3, had reached release candidate status. The upgrade to version 3 has been necessitated by the imminent release of MySQL 5.1, which is not supported by phpMyAdmin 2. Fortunately, the installation process for the new version remains unchanged—at least it was at the time I wrote the following instructions. Any changes of a substantial nature will be listed on my website at `http://foundationphp.com/dwcs4/updates.php`.

Downloading and installing phpMyAdmin

Since phpMyAdmin is PHP-based, all that's needed to install it is to download the files, unzip them to a website in your local testing environment, and create a simple configuration file. phpMyAdmin 3 requires a minimum of PHP 5.2 and MySQL 5.0. If you are running earlier versions, you must install phpMyAdmin 2.

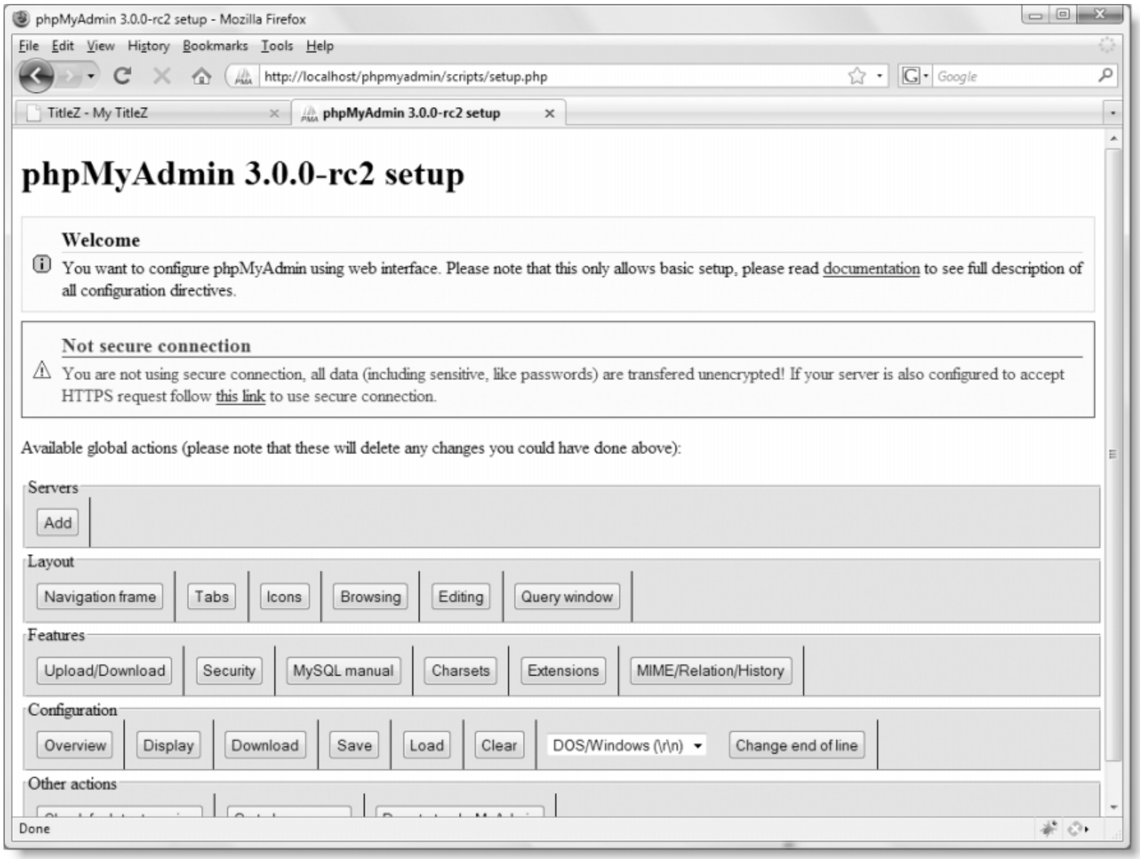
1. Go to `http://www.phpmyadmin.net`, and download the version you require. The files can be downloaded in three types of compressed file: BZIP2, GZIP, and ZIP. Choose whichever format you have the decompression software for.
2. Unzip the downloaded file. It will extract the contents to a folder called `phpMyAdmin-`*`x.x.x`*, where *x* represents the version number.
3. Highlight the folder icon, and cut it to your clipboard. On Windows, paste it inside the folder designated as your web server root (with an Apache server, this is usually a folder called `htdocs`). If you're on a Mac and want phpMyAdmin to be available to all users, put the folder in `Macintosh HD:Library:WebServer:Documents` rather than in your own `Sites` folder.
4. Rename the folder you have just moved to this: `phpMyAdmin`.
5. Create a new subfolder called `config` within the `phpMyAdmin` folder. Windows users skip to step 7. Mac users continue with step 6.
6. On Mac OS X, use Finder to locate the `config` folder you have just created. Ctrl-click and select Get Info. In Ownership & Permissions, expand Details, and click the lock icon so that you can make changes to the settings. Change the setting for Others to Read & Write. Close the config Info panel.
7. Open a browser, and type the following into the address bar:

   ```
   http://localhost/phpmyadmin/scripts/setup.php
   ```

   If you created the `phpMyAdmin` folder inside your `Sites` folder on a Mac, use the following address, substituting *`username`* with your Mac username:

   ```
   http://localhost/~username/phpmyadmin/scripts/setup.php
   ```

8. You should see the page shown in Figure 14-2.



phpMyAdmin 3.0.0-rc2 setup

Welcome

You want to configure phpMyAdmin using web interface. Please note that this only allows basic setup, please read documentation to see full description of all configuration directives.

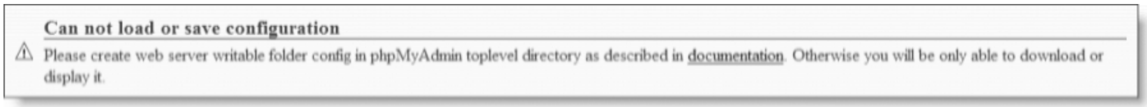
Not secure connection

You are not using secure connection, all data (including sensitive, like passwords) are transfered unencrypted! If your server is also configured to accept HTTPS request follow this link to use secure connection.

Available global actions (please note that these will delete any changes you could have done above):

Figure 14-2. A built-in script automates the configuration of phpMyAdmin.

Ignore any warning about the connection not being secure. This is intended for server administrators installing phpMyAdmin on a live Internet server. If, on the other hand, you see the following warning, it means you have not set up the `config` folder correctly and should go back to step 5.



Can not load or save configuration

Please create web server writable folder config in phpMyAdmin toplevel directory as described in documentation. Otherwise you will be only able to download or display it.

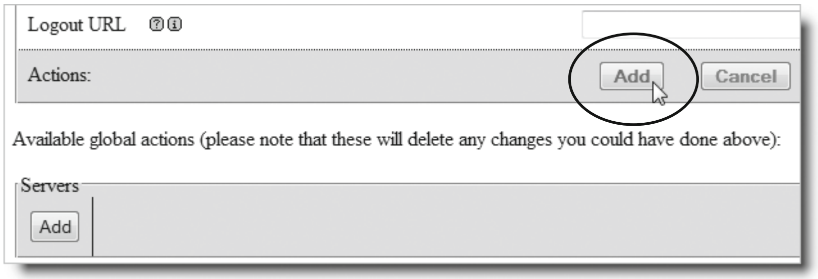
9. Click the Add button in the Servers section. This loads a form with most of the necessary information already filled in. Check the following settings:
   - Server hostname: localhost
   - Server port: Leave blank unless your web server is running on a nonstandard port, such as 8080

- Server socket: Leave blank
- Connection type: tcp
- PHP extension to use: mysqli

10. The default setting for Authentication type is config. If you don't need to password protect access to phpMyAdmin, check that User for config auth is set to root, and enter your MySQL root password in the next field, Password for config auth.

    If you want to restrict access to phpMyAdmin by prompting users for a password, change Authentication type to http, and delete root from the User for config auth field.

11. Scroll down to the Actions field, and click Add. As shown here, there are two Add buttons close to each other; click the one circled in the screenshot:



Available global actions (please note that these will delete any changes you could have done above):

12. The next screen will probably warn you that you didn't set up a phpMyAdmin database, so you won't be able to use all the phpMyAdmin features. This is not important. You can set up one later if you decide to use the advanced features of phpMyAdmin.

13. Scroll down to the Configuration section near the bottom of the page, and click Save.

14. Open the `config` folder in Explorer or Finder. You should see a new file called `config.inc.php`. Move it to the main `phpMyAdmin` folder. The official instructions tell you to delete the `config` folder, but this isn't necessary in a local testing environment.

Launching phpMyAdmin

To use phpMyAdmin, launch a browser, and enter http://localhost/phpMyAdmin/index.php in the address bar (on a Mac, use `http://localhost/~`*`username`*`/phpMyAdmin/index.php` if you put `phpMyAdmin` in your `Sites` folder). If you stored your root password in `config.inc.php`, phpMyAdmin should load right away, as shown in Figure 14-3. If you chose to password protect phpMyAdmin, enter root as the username and whatever you specified as the MySQL root password when prompted.

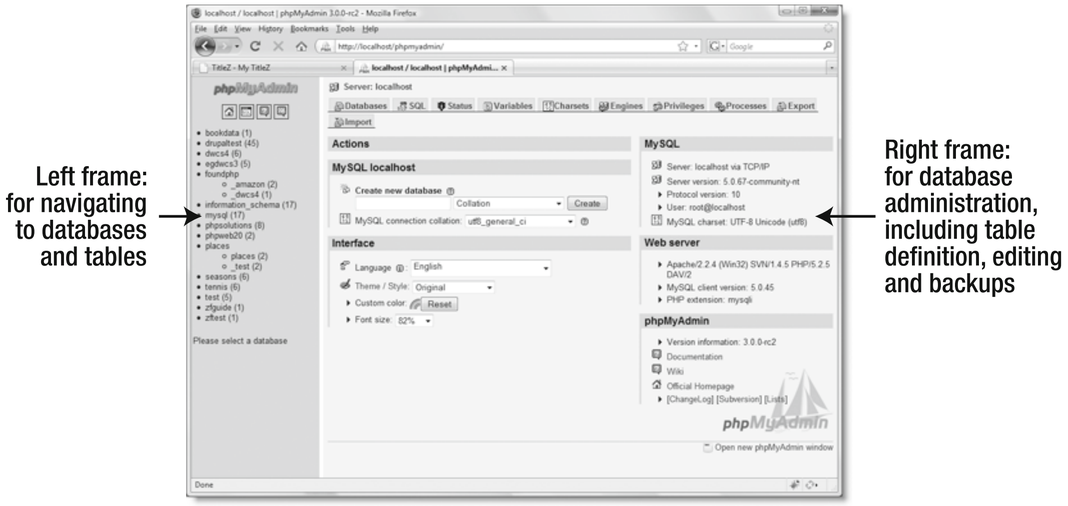

Left frame: for navigating to databases and tables
Right frame: for database administration, including table definition, editing and backups

Figure 14-3. phpMyAdmin is a very user-friendly and stable graphical interface to MySQL.

I have a large number of databases on my computer, so you'll have a much shorter list in the left frame than shown in Figure 14-3. If you're used to glossy software design, your initial impression of phpMyAdmin may not be all that favorable, particularly if you don't have a large monitor. The interface is sorely in need of a face-lift, but don't let that fool you; phpMyAdmin is both powerful and easy to use. The layout is slightly different in phpMyAdmin 2, so don't be surprised if your version doesn't look exactly like the screenshots in this book.

Troubleshooting

The following common errors occur when launching phpMyAdmin:

- If you get a message saying that the server is not responding or that the socket is not correctly configured, make sure that the MySQL server is running.
- If you get a message that the `mysqli` module cannot be loaded, there's a mistake in your installation of PHP. This normally happens only on Windows. Display your server's PHP configuration details by creating a script with `<?php phpinfo(); ?>` in it (and nothing else), and load it in a browser. Scroll roughly halfway down the page to locate sections for mysql and mysqli. If they're not there, you need to reinstall PHP and select both MySQL and MySQLi in the list of extensions to be enabled.
- If you get messages about failing to write session data or not being able to start a session without errors, it means that the folder PHP uses to save session information doesn't exist or is read-only. Use `phpinfo()` to display your PHP configuration details, and find the value of `session.save_path` (it's close to the bottom of the page in the session section). Make sure that the folder exists and is writable.

  If the folder doesn't exist, create it. On Windows, you don't normally need to set any permissions to make the folder writable. On a Mac, select the folder in Finder, and press Cmd+I to display a Get Info panel. In the Ownership & Permissions section, expand Details, and set Others to Read & Write.

Logging out of phpMyAdmin

If you opted to password protect phpMyAdmin, a Log out link is added to the front page. When you click the link, you are immediately prompted for your username and password. Click Cancel, and you are presented with a screen informing you that you supplied the wrong username/password—in other words, you have been logged out. Odd, but that's the way it works. You cannot log back in to phpMyAdmin from the wrong username/password screen. You must enter the original URL into the browser address bar.

Setting up a database in MySQL

MySQL isn't a single database, but a relational database management system (RDBMS). The screenshot in Figure 14-3 was taken on my development computer, which contains more than a dozen databases listed in the left frame of phpMyAdmin. However, if you examine a new installation of MySQL in phpMyAdmin, you'll see it contains the following three databases:

- `information_schema`: This is a virtual database that contains details of other databases within the RDBMS.
- `mysql`: This contains all the user account and security information and should never be edited directly unless you're really sure what you're doing.
- `test`: This contains nothing. You can either use it for testing or delete it.

The numbers phpMyAdmin displays in parentheses alongside each database name indicate how many tables that database contains.

If you're using a remote server and your hosting company provides phpMyAdmin, the list of databases will be limited to those on your account, or you may be limited to only one database.

Creating a local database for testing

Assuming you have set up a local testing environment, you need to create a test database to work with the remaining chapters. I'm going to call the database `dwcs4`, and that's how I'll refer to it from now on. However, if you already have a hosting package, I suggest you use the name of a database on your remote server, because this will make things a lot easier when it comes to testing your pages on the Internet.

Type the name of the database in the field labeled Create new database in the phpMyAdmin welcome screen, and click Create, as shown in Figure 14-4. Most readers can leave Collation in its default position. However, if you're working in a language other than English, Swedish, or Finnish, *and* your remote server runs MySQL 4.1 or later, read "Understanding collation" before clicking Create.

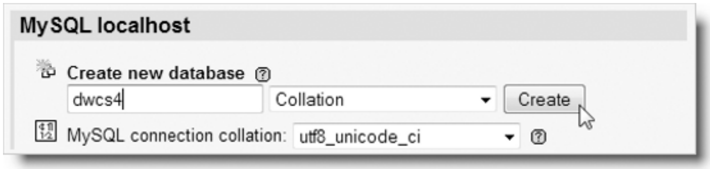


Figure 14-4. To create a new database, just type its name into the phpMyAdmin welcome screen, and click Create.

Because phpMyAdmin is a browser-based application, the precise layout of what you see onscreen depends on the size of your monitor and browser viewport. The layout of phpMyAdmin 3 also differs slightly from phpMyAdmin 2. However, the basic functionality remains the same.

The database should be created instantly, and phpMyAdmin will invite you to create a new table. Before doing that, you need to create at least one user account for the database. Leave phpMyAdmin open.

Understanding collation

Collation determines the sort order of records. Different languages have their own sorting rules, so MySQL 4.1 and above let you set the default sort order at different levels: for the entire database, for individual tables, and for individual columns. MySQL was originally developed in Sweden, so the default sort order is `latin1_swedish_ci`. English and Finnish share the same sort order.

If you work in a different language and your remote server is MySQL 4.1 or above, click the Charsets tab (or the Character Sets and Collations link in phpMyAdmin 2) on the phpMyAdmin welcome screen to see the full range of supported sort orders. When defining a new database or table, select the appropriate sort order from the Collation drop-down menu.

You can change the collation of an existing database or table in phpMyAdmin by selecting it in the left frame and then clicking the Operations tab. Since collation can be set at different levels, this sets the default only for new tables or columns. Existing tables and columns preserve their original collation unless you edit them individually.

If you are working in a language, such as Spanish or French, that uses accented characters, MySQL 3.23 and 4.0 do not support UTF-8 (Unicode). This affects the way accented characters are stored. If accented characters are garbled when retrieving records from MySQL, change the default encoding of your web pages from UTF-8 to the encoding appropriate for your language. Alternatively, store accented characters as HTML entities (for example, `&eacute;` *for é). Better still, upgrade to MySQL 5.*

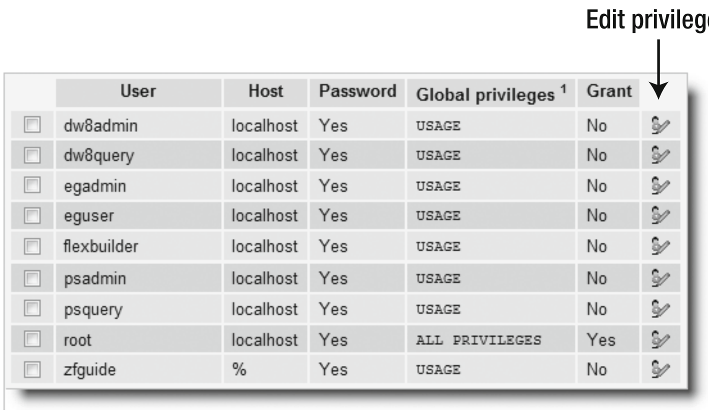
Creating user accounts for MySQL

A new installation of MySQL has only one registered user—the superuser account called **root**, which has complete control over everything. A lot of beginners use root for everything and don't even bother setting up a password for root. This is a big mistake. The root user should *never* be used for anything other than administration, and you should get into the good habit of using a password for root. XAMPP and MAMP both have instructions for setting the root password. If you need to change the root password in phpMyAdmin, follow the instructions in the next section. Otherwise, skip ahead to "Granting user privileges."

Changing the MySQL root password in phpMyAdmin

Changing the MySQL root password in phpMyAdmin is quick and easy. Just follow these steps:

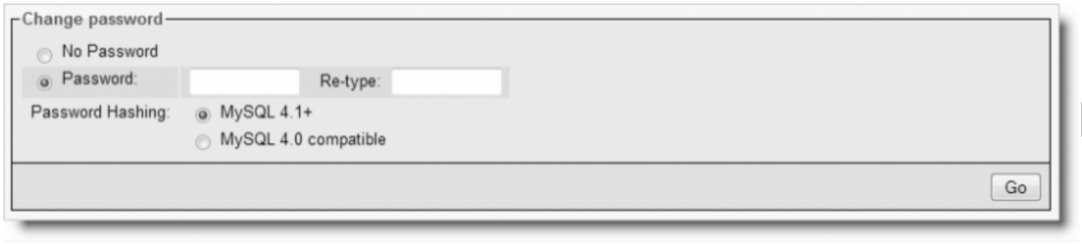
1. Launch phpMyAdmin, and click the Privileges tab in the welcome screen (it's a link on the left side of the main frame in phpMyAdmin 2).
2. This displays a list of MySQL user accounts, as shown in Figure 14-5 (if it's a new installation, the only one listed is root). Click the Edit privileges icon alongside root.



	User	Host	Password	Global privileges [1]	Grant	
☐	dw8admin	localhost	Yes	USAGE	No	✎
☐	dw8query	localhost	Yes	USAGE	No	✎
☐	egadmin	localhost	Yes	USAGE	No	✎
☐	eguser	localhost	Yes	USAGE	No	✎
☐	flexbuilder	localhost	Yes	USAGE	No	✎
☐	psadmin	localhost	Yes	USAGE	No	✎
☐	psquery	localhost	Yes	USAGE	No	✎
☐	root	localhost	Yes	ALL PRIVILEGES	Yes	✎
☐	zfguide	%	Yes	USAGE	No	✎

Figure 14-5. Click the icon in the right column to edit a user's privileges.

3. This opens the Edit Privileges screen. Scroll down until you find the following section:


Change password

○ No Password
● Password: ______ Re-type: ______

Password Hashing: ● MySQL 4.1+
○ MySQL 4.0 compatible

4. Select the Password radio button, and enter the new password in both fields. Unless you are using an old version of MySQL, leave Password Hashing on the default, MySQL 4.1+.
5. Click Go in the Change password section. There are several Go buttons on the page. Make sure you select the one in the right section.
6. If you selected `config` as the authentication type when setting up phpMyAdmin, don't forget to update `config.inc.php`. You can do this manually by opening the file and changing the following line:

```
$cfg['Servers'][$i]['password'] = 'newRootPassword';
```

Granting user privileges

MySQL stores all databases in a common directory. So, on shared hosting, your database—with all its precious information—rubs shoulders with everyone else's. Clearly, you need a way to prevent unauthorized people from seeing or altering your data. The answer is to create user accounts that have the fewest number of privileges necessary to perform essential tasks, preferably on a single database.

You normally want visitors to your site to be able to see the information it contains but not to change it. However, as administrator, you need to be able to insert new records and update or delete existing ones. This involves four types of privileges, all named after the equivalent SQL commands:

- `SELECT`: Retrieves records from database tables
- `INSERT`: Inserts records into a database
- `UPDATE`: Changes existing records
- `DELETE`: Deletes records but not tables or databases (the command for that is `DROP`)

In an ideal setup, you create two separate user accounts: one for administrators, who require all four privileges, and another one for visitors, limited to `SELECT`. If your hosting company lets you set up user accounts with different privileges, I suggest you create two accounts like this. However, if you have no choice, set up one account and use the same username and password as on your remote server.

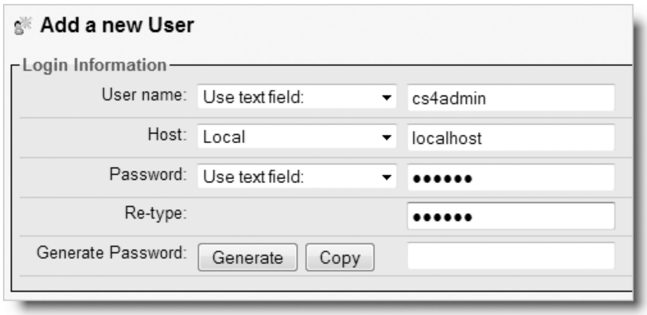
Setting up MySQL user accounts

These instructions show you how to set up user accounts in a local testing environment. You can skip this section if you are using your remote server as your testing server.

1. Click the home icon at the top of the left frame in phpMyAdmin to return to the welcome screen, and then click Privileges. In phpMyAdmin 3, this is a tab at the top of the screen. In phpMyAdmin 2, it's a link in the left column of the main frame.

> *To create a new user account, you must use the link in the welcome screen. The* Privileges *tab in other screens displays details of existing accounts only.*

2. The User overview screen opens. Click Add a new User halfway down the page.
3. In the page that opens, enter the name of the user account that you want to create in the User name field. Select Local from the Host drop-down menu. This automatically enters localhost in the field alongside. This option restricts the user to connecting to MySQL only from the same computer. Enter a password in the Password field, and confirm it in the Re-type field. The Login Information table should look like this:

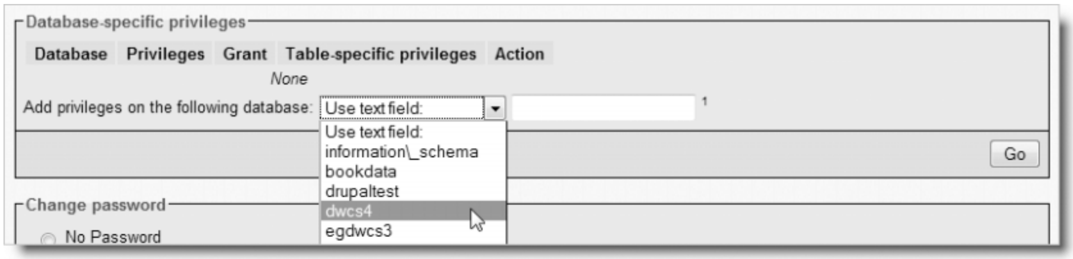

Add a new User

Login Information

User name:	Use text field:	cs4admin
Host:	Local	localhost
Password:	Use text field:	••••••
Re-type:		••••••
Generate Password:	Generate Copy	

Dreamweaver needs these details later to make a connection to the database. The password I used for the `cs4admin` *user when creating the download files is* `humpty`. *If you want to use the download files exactly as they are, you need to use the same password and username as I did. However, I suggest you use your own username and password both here and when creating the MySQL connection in Dreamweaver later in the chapter.*

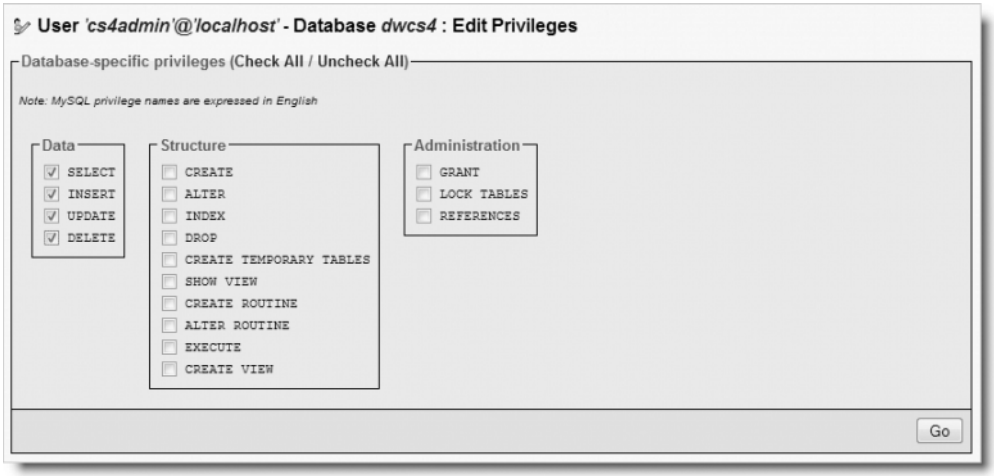
4. Beneath the Login Information table is one labeled Global privileges. Granting such extensive privileges is insecure, so scroll past the Global privileges table, and click the Go button at the bottom of the page.
5. The next page confirms that the user has been created and displays many options, beginning with the Global privileges again. Scroll down to the section labeled Database-specific privileges. Activate the drop-down menu, as shown here, to display a list of all databases. Select the name for the database you plan to use for testing.


Database-specific privileges

Database	Privileges	Grant	Table-specific privileges	Action
		None		

Add privileges on the following database: Use text field:

6. The next screen allows you to set the user's privileges for just this database. You want the admin user to have all four privileges listed earlier, so click the SELECT, INSERT, UPDATE, and DELETE checkboxes (if you hover your mouse pointer over each option, phpMyAdmin displays a tooltip describing what it's for). After selecting the four privileges, as shown here, click the top Go button.



User 'cs4admin'@'localhost' - Database dwcs4 : Edit Privileges

Database-specific privileges (Check All / Uncheck All)

Note: MySQL privilege names are expressed in English

Data
- [✓] SELECT
- [✓] INSERT
- [✓] UPDATE
- [✓] DELETE

Structure
- [ ] CREATE
- [ ] ALTER
- [ ] INDEX
- [ ] DROP
- [ ] CREATE TEMPORARY TABLES
- [ ] SHOW VIEW
- [ ] CREATE ROUTINE
- [ ] ALTER ROUTINE
- [ ] EXECUTE
- [ ] CREATE VIEW

Administration
- [ ] GRANT
- [ ] LOCK TABLES
- [ ] REFERENCES

phpMyAdmin frequently offers you a variety of options on the same page, each of which normally has its own Go *button. Always click the one at the foot of or alongside the section that relates to the options you want to set.*

7. phpMyAdmin presents you with confirmation that the privileges have been updated for the user account. The page displays the Database-specific privileges table again, in case you need to change anything. Assuming you got it right, click the Privileges tab at the top right of the page. You should now see the new user listed in the User overview.

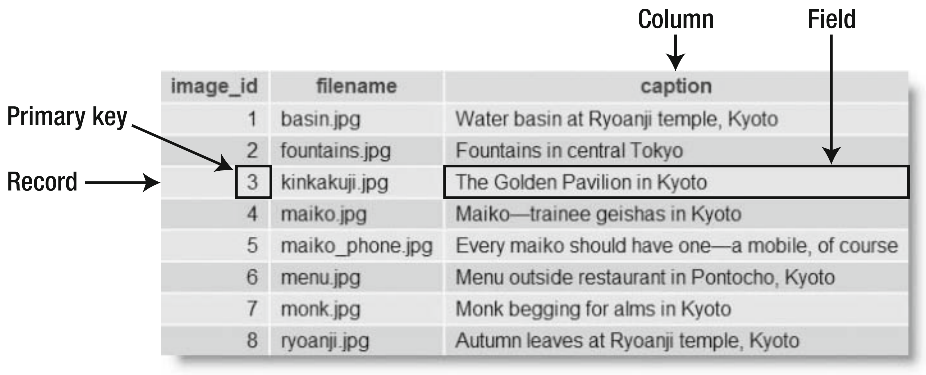
   If you ever need to make any changes to a user's privileges, click the Edit Privileges icon to the right of the listing (see Figure 14-5 in the previous section). You can also delete users by selecting the checkbox to the left of the User column and then clicking Go.

8. If your hosting company permits you to create multiple user accounts, click Add a new User, and repeat steps 3–7 to create a second user account. If you want to use the same username and password as in the download files, call the account `cs4user`, and give it the password `dumpty`. This user will have restricted privileges, so in step 6, check only the SELECT option.

Now that you have a database and at least one user account, you can start adding tables to store information. However, first, you need to understand the principles behind table construction.

How a database stores information

All data in MySQL is stored in tables, with information organized into rows and columns very much like a spreadsheet. Figure 14-6 shows a simple database table as seen in phpMyAdmin.



image_id	filename	caption
1	basin.jpg	Water basin at Ryoanji temple, Kyoto
2	fountains.jpg	Fountains in central Tokyo
3	kinkakuji.jpg	The Golden Pavilion in Kyoto
4	maiko.jpg	Maiko—trainee geishas in Kyoto
5	maiko_phone.jpg	Every maiko should have one—a mobile, of course
6	menu.jpg	Menu outside restaurant in Pontocho, Kyoto
7	monk.jpg	Monk begging for alms in Kyoto
8	ryoanji.jpg	Autumn leaves at Ryoanji temple, Kyoto

Figure 14-6. Information in a database table is stored in rows and columns, just like in a spreadsheet.

Each **column** has a name (`image_id`, `filename`, and `caption`) indicating what it stores.

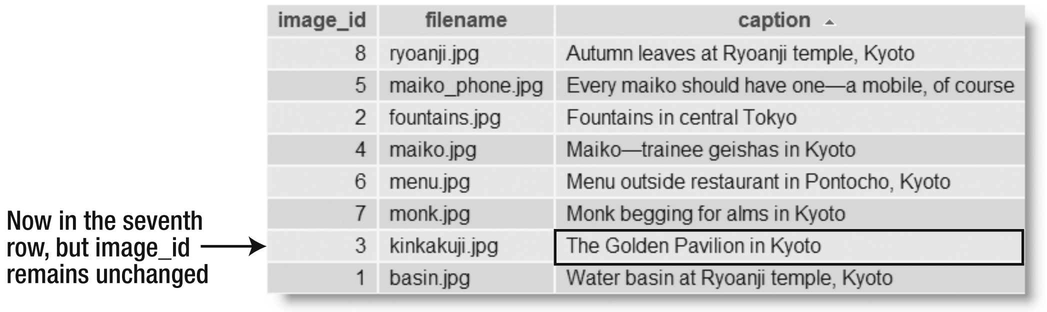
The rows aren't labeled, but the first column (`image_id`) contains a unique identifier known as a **primary key**, which can be used to identify the data associated with a particular row. Each row contains an individual **record** of related data. The significance of primary keys is explained in the next section.

The intersection of a row and a column, where the data is stored, is called a **field**. So, for instance, the `caption` field for the third record in Figure 14-6 contains the value `The Golden Pavilion in Kyoto`, and the primary key for that record is 3.

The terms "field" and "column" are often used interchangeably. A field holds one piece of information for a single record, whereas a column contains the same field for all records.

How primary keys work

Although Figure 14-6 shows `image_id` as a consecutive sequence from 1 to 8, they're not row numbers. Figure 14-7 shows the same table with the captions sorted in alphabetical order. The field highlighted in Figure 14-6 has moved to the seventh row, but it still has the same `image_id` and `filename`.



image_id	filename	caption ▲
8	ryoanji.jpg	Autumn leaves at Ryoanji temple, Kyoto
5	maiko_phone.jpg	Every maiko should have one—a mobile, of course
2	fountains.jpg	Fountains in central Tokyo
4	maiko.jpg	Maiko—trainee geishas in Kyoto
6	menu.jpg	Menu outside restaurant in Pontocho, Kyoto
7	monk.jpg	Monk begging for alms in Kyoto
3	kinkakuji.jpg	The Golden Pavilion in Kyoto
1	basin.jpg	Water basin at Ryoanji temple, Kyoto

Figure 14-7. Even when the table is sorted in a different order, each record can be identified by its primary key.

Although the primary key is rarely displayed, it identifies the record and all the data stored in it. If you know the primary key, you can update a record, delete it, or use it to display data. Don't worry about how you find the primary key; it's easy using Structured Query Language (SQL), the standard means of communicating with all major databases. The important thing is to assign a primary key to every record.

- A primary key doesn't need to be a number, but *it must be unique*.
- Social Security, staff ID, or product numbers make good primary keys. They may consist of a mixture of numbers, letters, and other characters but are always different.
- MySQL will generate a primary key for you automatically.
- Once a primary key has been assigned, it should never—repeat, never—be changed.

Because a primary key must be unique, MySQL doesn't normally reuse the number when a record is deleted, leaving holes in the sequence. *Don't even think about renumbering.* By changing the numbers to close the gaps, you put the integrity of your database at serious risk. Some people want to remove gaps to keep track of the number of records, but you can easily get the same information with SQL.

Although Figures 14-6 and 14-7 show the similarity between a database table and a spreadsheet, there's an important difference. With a spreadsheet, you can enter data without the need to specify beforehand what type of data it is or how it's to be structured. You can't do that with a database.

Designing a database table

Before entering data, you need to define the table structure. This involves the following decisions:

- The name of the table
- How many columns it will have

- The name of each column
- What type of data will be stored in each column
- Whether the column must always have data in each field
- Which column contains the table's primary key

Don't be tempted to choose the first thing that comes into your head. Experienced database developers often say at least half the total development time is spent deciding the structure of a database. Although the structure of a database can be altered, some decisions tie your hands so badly you need to redesign everything from scratch. That's not much fun when the database contains several thousand records. The time spent on these early decisions can save a lot of agony and frustration later.

Because each database is different, it's impossible to prescribe one simple formula, but the next few pages should help guide you in the right direction. Don't attempt to commit everything to memory at the first read-through. Come back later when you need to refresh your memory or check a particular point.

Choosing the table name

The basic MySQL naming rules for databases, tables, and columns are as follows:

- Names can be up to 64 characters long.
- Legal characters are numbers, letters, the underscore, and $.
- Names can begin with a number but cannot consist exclusively of numbers.

Some hosting companies seem blissfully ignorant of these rules and assign clients databases that contain one or more hyphens (an illegal character) in their names. If a name contains spaces or illegal characters, you must surround it by backticks (`) in SQL queries. Note that this is not a single quote (') but a different character. Dreamweaver and phpMyAdmin normally do this for you automatically.

Choose names that are meaningful. Tables hold groups of records, so it's a good strategy to use plural nouns. For example, use `products` rather than `product`. Don't try to save on typing by using abbreviations, particularly when naming columns. Explicit names make it much easier to build SQL queries to extract the information you want from a database. SQL is designed to be as human-readable as possible, so don't make life difficult for yourself by using cryptic naming conventions.

When choosing column names, there is a danger that you might accidentally choose one of MySQL's many reserved words (`http://dev.mysql.com/doc/refman/5.0/en/reserved-words.html`), such as `date` or `time`. A good technique is to use compound words, such as `arrival_date`, `arrival_time`, and so on. These names also tell you much more about the data held in the column.

Case sensitivity of names

Windows and Mac OS X treat MySQL names as case-insensitive. However, Linux and Unix servers respect case sensitivity. To avoid problems when transferring databases and PHP code from your local computer to a remote server, I recommend you use only lowercase in database, table, and column names. Using camel case (for example, `arrivalDate`) is likely to cause your code to fail when transferring a database from your local computer to a Linux server.

Deciding how many columns to create

How should you store each person's name? One column? Or one each for the family and personal names? A commercial contacts management program like Microsoft Outlook goes even further, splitting the name into five parts. In addition to first and last name, it stores a title (Mr., Mrs., and so on), a middle name, and a suffix (I, II, III, Jr., and Sr.). Addresses are best broken down into street, town, county, state, ZIP code, and so on. Think of all the possible alternatives, and add a column for each one. Things like company name, apartment number, and extra lines in an address can be made optional, but you need to make provision for them. This is an important principle of a relational database: *break down complex information into its component parts, and store each part separately*.

This makes searching, sorting, and filtering much easier. Breaking information into small chunks may seem a nuisance, but you can always join them together again. It's much easier than trying to separate complex information stored in a single field.

Choosing the right column type in MySQL

MySQL 5.0 has 28 different column types. Rather than confuse you by listing all of them, I'll explain just the most commonly used. You can find full details of all column types in the MySQL documentation at `http://dev.mysql.com/doc/refman/5.0/en/data-types.html`.

Storing text

The difference between the main text column types boils down to the maximum number of characters that can be stored in an individual field and whether you can set a default value.

- `CHAR`: A fixed-length width text column up to a maximum of 255 characters. You must specify the size when building the table, although this can be altered later. Shorter strings are OK. MySQL adds trailing space to store them and automatically removes it on retrieval. If you attempt to store a string that exceeds the specified size, excess characters are truncated. You can define a default value.
- `VARCHAR`: A variable-length character string. The maximum number must be specified when designing the table, but this can be altered later. Prior to MySQL 5.0, the limit is 255; this has been increased to 65,535 in MySQL 5.0. Another change in MySQL 5.0 affects the way trailing space is treated. Prior to MySQL 5.0, trailing space is stripped at the time of storing a record. Since MySQL 5.0, trailing space is retained for both storage and retrieval. You can define a default value.
- `TEXT`: Stores a maximum of 65,535 characters (approximately two thirds of this chapter). You cannot define a default value.

`TEXT` is convenient, because you don't need to specify a maximum size (in fact, you can't). Although the maximum length of `VARCHAR` is the same as `TEXT` in MySQL 5.0, other factors such as the number of columns in a table reduce this.

Prior to MySQL 5.0, you cannot use CHAR in a table that also contains VARCHAR, TEXT, or BLOB. When creating the table, MySQL silently converts any CHAR columns to VARCHAR.

Keep it simple: use VARCHAR *for short text items and* TEXT *for longer ones.*

Storing numbers

The most frequently used numeric column types are as follows:

- TINYINT: Any whole number (integer) between –128 and 127. If the column is declared as UNSIGNED, the range is from 0 to 255. This is particularly suitable for storing people's ages, number of children, and so on.
- INT: Any integer between –2,147,483,648 and 2,147,483,647. If the column is declared as UNSIGNED, the range is from 0 to 4,294,967,295.
- FLOAT: A floating-point number.
- DECIMAL: A floating-point number *stored as a string. This column type is best avoided*.

DECIMAL is intended for currencies, but you can't perform calculations with strings inside a database, so it's more practical to use INT. For dollars or euros, store currencies as cents; for pounds, use pence. Then use PHP to divide the result by 100, and format the currency as desired.

Don't use commas or spaces as the thousands-separator. Apart from numerals, the only characters permitted in numbers are the negative operator (-) and the decimal point (.*). Although some countries use a comma as the decimal point, MySQL accepts only a period.*

Storing dates and times

MySQL stores dates in the format YYYY-MM-DD. This may come as a shock, but it's the ISO (International Organization for Standardization) standard, and it avoids the ambiguity inherent in national conventions. The most important column types for dates and times are as follows:

- DATE: A date stored as YYYY-MM-DD. The supported range is 1000-01-01 to 9999-12-31.
- DATETIME: A combined date and time displayed in the format YYYY-MM-DD HH:MM:SS.
- TIMESTAMP: A timestamp (normally generated automatically by the computer). Legal values range from the beginning of 1970 to partway through 2037.

MySQL timestamps are based on a human-readable date and, since MySQL 4.1, use the same format as DATETIME. As a result, they are incompatible with Unix and PHP timestamps, which are based on the number of seconds elapsed since January 1, 1970. Don't mix them.

Attempting to insert a date in any format other than `YYYY-MM-DD` *results in the date being stored as* `0000-00-00`. *Handling dates in different formats is covered in Chapter 17.*

Storing predefined lists

MySQL lets you store two types of predefined lists that could be regarded as the database equivalents of radio button and checkbox states:

- `ENUM`: This column type stores a single choice from a predefined list, such as "yes, no, don't know" or "male, female." The maximum number of items that can be stored in the predefined list is a mind-boggling 65,535—some radio-button group!
- `SET`: This stores zero or more choices from a predefined list, up to a maximum of 64. Although this violates the principle of storing only one piece of information in a field, it's useful when the items form a coherent unit (for example, optional extras on a car).

The values stored in the `ENUM` and `SET` columns are stored as a comma-separated string. Individual values can include spaces and other characters but not commas.

Storing binary data

Binary data, such as images, bloat your tables and cannot be displayed directly from a database. However, the following column types are designed for binary data:

- `TINYBLOB`: Up to 255 bytes
- `BLOB`: Up to 64KB
- `MEDIUMBLOB`: Up to 16MB
- `LONGBLOB`: Up to 4GB

With such whimsical names, it's a bit of a letdown to discover that `BLOB` stands for **binary large object**.

Deciding whether a field can be empty

When defining a database table, specifying a column as `NOT NULL` is the equivalent of designating a required field. Since the phpMyAdmin default is `NOT NULL`, you need to manually override this to make a field optional. You can change a column definition from `NOT NULL` to `NULL`, and vice versa, at any time.

If you set a default value for a `NOT NULL` *column, MySQL automatically uses that value if nothing is entered in the field. Unfortunately, Dreamweaver doesn't support this useful feature.*

Creating a user registration system

After that essential introduction, it's now time to get down to business and create a simple application that registers a person's name, username, and password. This will become the basis of a user registration system that controls access to selected pages in your website. Once you have created the database table, Dreamweaver's Insert Record, Update Record, and Delete Record server behaviors make light work of creating the forms that add the user's details to the database. They are easy to use, and they protect you against a type of malicious attack known as **SQL injection**. An injection attack can be used to reveal sensitive information or even delete all your data by passing spurious values through form fields or URL query strings. That's the good news . . . .

The not-so-good news is that the server behaviors do nothing to ensure that user input meets your criteria for suitable data. So, you could end up with someone just pressing the spacebar a couple of times, rather than typing a username or a password. However, that's an issue that can wait until the next chapter. To begin with, I'll concentrate on getting your first database application up and running.

To register users for your site, you need the following elements:

- A database table to store user details, such as username and password
- A registration form
- A page to display a list of registered users
- A form to update user details
- A form to delete users

Defining the database table

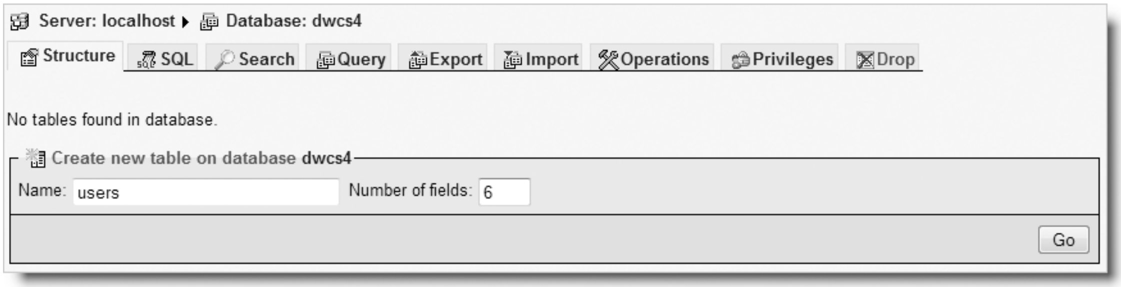
Let's start with creating the necessary table to store user details in the database. I plan to use the same table for both site administrators and ordinary visitors. So, it will also store the level of user privileges. This means the table needs a total of six columns to store the user's first name, family name, username, password, and privilege level. That's only five . . . . The missing column is needed for the primary key.

Creating the users table

These instructions show you how to define the `users` table in phpMyAdmin. If you're new to working with MySQL, I suggest you work through this section step-by-step to familiarize yourself with table definition. More experienced users might prefer to use the phpMyAdmin Import tab to build the table structure with `ch14_users.sql` in the `extras` folder of the download files (for MySQL 4.0 use `ch14_users40.sql`).

1. Launch phpMyAdmin, and select the `dwcs4` database from the list of databases in the left frame. Since the database doesn't yet have any tables, you should see a message that no tables were found and a form to create a new one. You want to

create new table called `users`. It needs to have six columns, so fill in the form as shown here, and click Go.


Server: localhost ▸ Database: dwcs4

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Drop

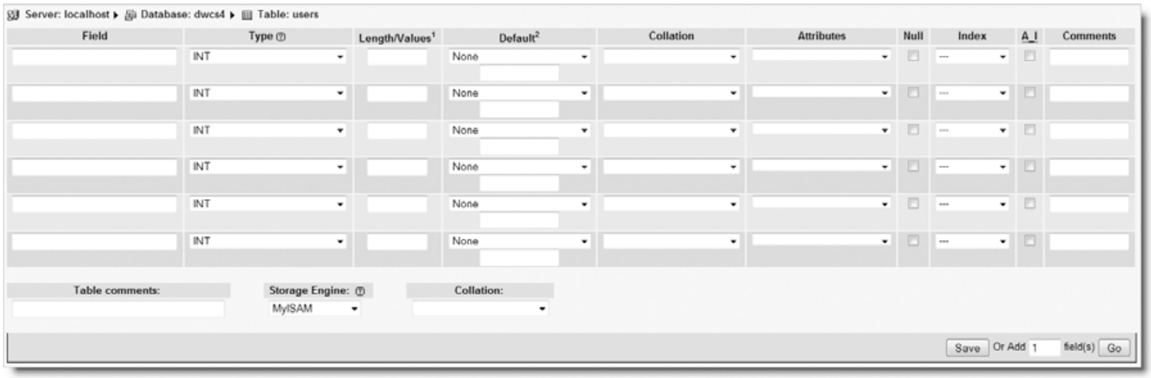
No tables found in database.

Create new table on database **dwcs4**

Name: users   Number of fields: 6

Go

2. This opens a huge matrix where you define the table. Although it looks intimidating at first glance, it's quite straightforward to fill in. The layout in phpMyAdmin 3 has changed since the previous version, so both versions are shown in Figures 14-8 and 14-9.

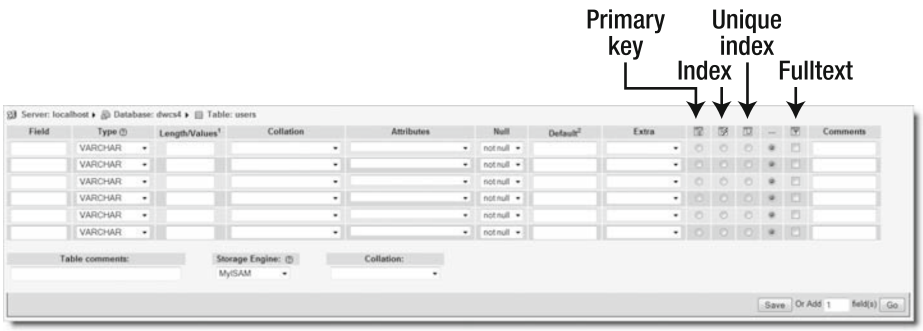

Server: localhost ▸ Database: dwcs4 ▸ Table: users

Field	Type	Length/Values[1]	Default[2]	Collation	Attributes	Null	Index	A_I	Comments
	INT		None			☐	---	☐	
	INT		None			☐	---	☐	
	INT		None			☐	---	☐	
	INT		None			☐	---	☐	
	INT		None			☐	---	☐	
	INT		None			☐	---	☐	

Table comments: Storage Engine: MyISAM   Collation:

Save | Or Add 1 field(s) | Go

Figure 14-8. The new table definition layout in phpMyAdmin 3



Server: localhost ▸ Database: dwcs4 ▸ Table: users

Field	Type	Length/Values[1]	Collation	Attributes	Null	Default[2]	Extra				—		Comments
	VARCHAR				not null			○	○	○	●	☐	
	VARCHAR				not null			○	○	○	●	☐	
	VARCHAR				not null			○	○	○	●	☐	
	VARCHAR				not null			○	○	○	●	☐	
	VARCHAR				not null			○	○	○	●	☐	
	VARCHAR				not null			○	○	○	●	☐	

Table comments: Storage Engine: MyISAM   Collation:

Save | Or Add 1 field(s) | Go

Figure 14-9. The phpMyAdmin 2 interface uses radio buttons to specify indexes.

For the sake of consistency, I will use screenshots of phpMyAdmin 3. If you are using phpMyAdmin 2 or need to switch between versions because your hosting company still uses phpMyAdmin 2, you should be aware of the following differences:

- When setting a default value in phpMyAdmin 3, you need to select a value from the drop-down menu. The options are None, As defined, NULL, and CURRENT_TIMESTAMP. If you select As defined, type the value in the field below. In phpMyAdmin 2, you simply enter a default value or leave the field blank.
- Both versions of phpMyAdmin set all columns to `NOT NULL`—in other words, required. To make a column optional in phpMyAdmin 3, select the Null checkbox; in phpMyAdmin 2, select null from the drop-down menu.
- phpMyAdmin 3 uses a drop-down menu to specify whether the column should have an index (this includes setting the table's primary key). In phpMyAdmin 2, use the radio buttons labeled in Figure 14-9.
- To create an auto incrementing column (normally used in conjunction with the primary key), select the A_I checkbox in phpMyAdmin 3. In phpMyAdmin 2, select auto_increment from the Extra drop-down menu.

The settings for each column are summarized in Table 14-1.

Table 14-1. Settings for the users table

Field	Type	Length/Values	Default	Attributes	Null	Index	A_I
`user_id`	INT		None	UNSIGNED	No	PRIMARY	Yes
`username`	VARCHAR	15	None		No	UNIQUE	No
`pwd`	VARCHAR	40	None		No		No
`first_name`	VARCHAR	30	None		No		No
`family_name`	VARCHAR	30	None		No		No
`admin_priv`	ENUM	'n', 'y'	n		No		No

The table's primary key is `user_id`. Setting the Attributes field to UNSIGNED restricts the column to use only positive numbers. By selecting A_I (auto_increment), the value will automatically increase by one each time a record is added to the table.

The next column, `username`, has Type set to VARCHAR with a length of 15, which should be long enough for a username. You don't want anyone to have the same username as anyone else, so a unique index is applied to the column ensuring that the same value can never be entered more than once.

The next three columns—pwd, first_name, and family_name—all have Type set to VARCHAR. I have set the length of pwd to 40, because the function used to encrypt the passwords always produces a hexadecimal string exactly 40 characters long.

Thirty characters each for first_name and family_name might seem a lot, but it's better to be overgenerous than to end up with truncated data.

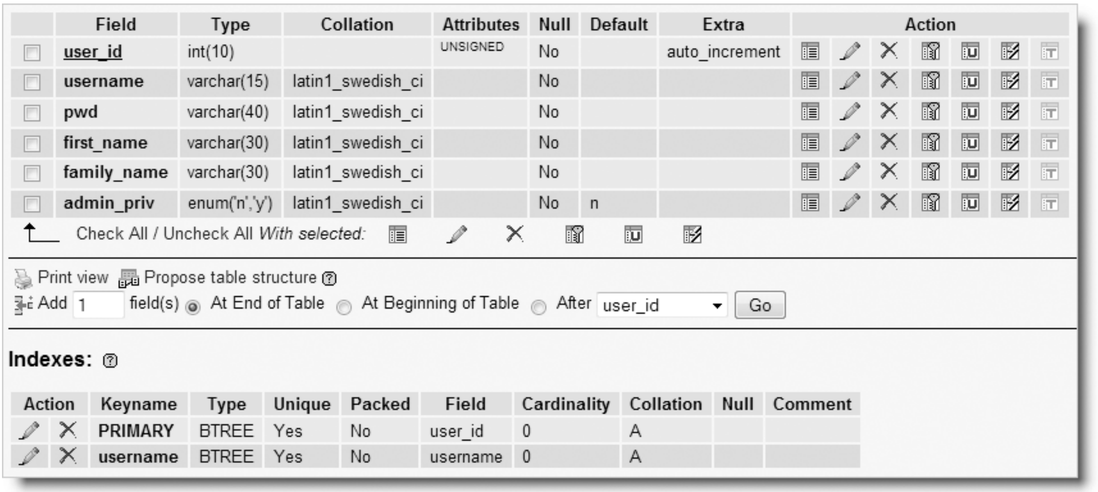
The final column, admin_priv, uses the ENUM column type. As explained earlier, this is typically used for "choose one of the following" situations. In this case, it's whether a user has administrative privileges. Type the permitted values in the Length/Values field as comma-separated strings like this:

'n', 'y'

In the Default column for admin_priv, enter n without any quotes (in phpMyAdmin 3, you also need to set the Default drop-down menu to As defined).

All columns have been set to not null. This is because I want all of them to be required fields. Click Save.

3. Check that the table structure displayed in phpMyAdmin looks like this:



	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	user_id	int(10)		UNSIGNED	No		auto_increment	
	username	varchar(15)	latin1_swedish_ci		No			
	pwd	varchar(40)	latin1_swedish_ci		No			
	first_name	varchar(30)	latin1_swedish_ci		No			
	family_name	varchar(30)	latin1_swedish_ci		No			
	admin_priv	enum('n','y')	latin1_swedish_ci		No	n		

Check All / Uncheck All *With selected:*

Print view Propose table structure

Add 1 field(s) ● At End of Table ○ At Beginning of Table ○ After user_id Go

Indexes:

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	user_id	0	A		
	username	BTREE	Yes	No	username	0	A		

Note that the Indexes table at the bottom left of the screenshot lists user_id as the primary key and both user_id and username are listed as unique indexes. The primary key is always unique.

If you need to make any changes, click the pencil icon in the row that needs amending. To change several rows, select the checkbox alongside the column names, and click the pencil icon at the bottom of the table structure. If you make a complete mess and need to start again, click the Drop tab at the top right of the screen and confirm that you want to delete the table.

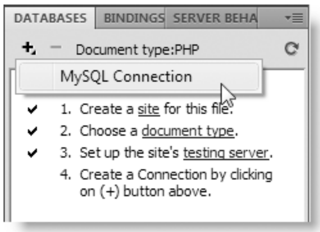
Telling Dreamweaver how to connect to the database

Before you can communicate with your database inside Dreamweaver, you need to create a MySQL connection. If you defined your site correctly in Chapter 2, it should take no more than a minute or two.

Creating a MySQL connection

A MySQL connection is simply a convenient way of storing the details needed to connect to MySQL: the server address, username, password, and database name. Dreamweaver stores them in an include file, which it automatically attaches to a web page whenever you select the connection in a server behavior.

1. Before you can create a MySQL connection, you need to have a PHP page open in the Document window. Create a blank PHP page, and save it as `register_user.php` in `workfiles/ch14`.
2. With `register_user.php` open in the Document window, open the Databases panel. If you can't see the panel, you can open it from the menu system (Window ➤ Databases) or use the keyboard shortcut Ctrl+Shift+F10/Shift+Cmd+F10.
3. Click the plus (+) button, and select MySQL Connection, as shown here:



DATABASES BINDINGS SERVER BEHA

Document type:PHP

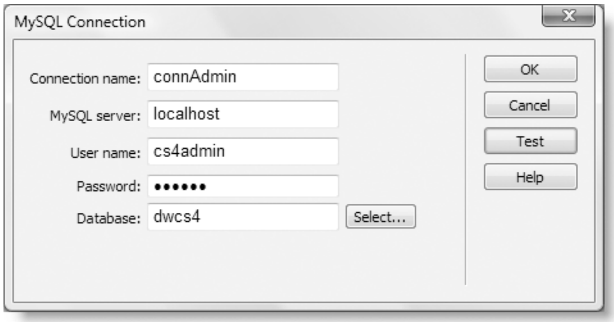
MySQL Connection

1. Create a site for this file.
2. Choose a document type.
3. Set up the site's testing server.
4. Create a Connection by clicking on (+) button above.

4. The dialog box that opens asks you for the following details:
   - Connection name: You can choose any name you like, but it must not contain any spaces or special characters. This connection will be used by the administrator user account, so I have entered connAdmin.
   - MySQL server: This is the address of the database server. If MySQL is on the same computer as Dreamweaver, you should enter localhost.
   - If you are running MySQL on a port other than the default 3306 (this happens with some of the all-in-one PHP packages, such as MAMP), add the port number after a colon (for example, localhost:8889).
   - If you are using your remote server as a testing server, use the address your hosting company gave you. In most cases, this is also `localhost`. Dreamweaver uploads hidden files to your remote server and creates a local connection there.
   - Some hosting companies locate the MySQL server on a different computer from your web files. If you are doing remote testing and have been given a server name other than `localhost`, enter that name now. If you are testing locally but know that your host doesn't use `localhost`, you will have to change this field when you finally upload your site to the remote server.
   - User name: Enter the name of the MySQL user account that you want to use. This connection will be used for administrative pages, so I have entered cs4admin.

- Password: Enter the password for the user account. This should be the password you registered when creating the MySQL account.
- Database: Enter the name of the database that you want to use. You can also use the Select button to get Dreamweaver to show you a list of databases that the named user has access to.

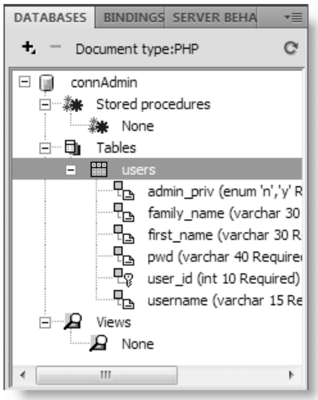
Fill in the necessary details. The completed dialog box should look something like the following screenshot. When you have finished, click the Test button. If all goes well, Dreamweaver will tell you that the connection was made successfully.



Connection name: connAdmin
MySQL server: localhost
User name: cs4admin
Password: ••••••
Database: dwcs4

5. If you got the thumbs-up from Dreamweaver, click OK to close both dialog boxes. If you failed to make the connection, cancel the connection setup, and check the points listed in step 4 before trying again. If that fails, see "Troubleshooting the connection."

6. In the Databases panel, you should see a database icon that has been created for connAdmin. Expand the tree menu by clicking the tiny plus button (it's a triangle on the Mac) to the left of connAdmin. It displays the database features available to the connection, including a brief description of every column in the users table. The columns are listed in alphabetical order, not the order they appear in the database. The little key icon alongside user_id indicates that it's the table's primary key. Both Stored procedures and Views are empty. Although MySQL 5.0 supports these features, support for them has not been implemented in Dreamweaver CS4.



```
connAdmin
  Stored procedures
    None
  Tables
    users
      admin_priv (enum 'n','y' R
      family_name (varchar 30
      first_name (varchar 30 R
      pwd (varchar 40 Require
      user_id (int 10 Required)
      username (varchar 15 Re
  Views
    None
```

If you ever need to change the connection details, double-click the database icon in the Databases panel to reopen the MySQL Connection dialog box, make your changes, and click OK. Alternatively, right-click the connection name, and choose Edit Connection from the context menu.

7. If you have created two user accounts for MySQL, create another MySQL connection called connUser for the second account that has only SELECT privileges.

Dreamweaver stores the MySQL connection details in a file with the same name as the connection. So, `connAdmin` *becomes* `connAdmin.php`*, which is stored in a folder called* `Connections` *that Dreamweaver creates in the site root. Don't forget to upload the contents of this folder to your remote server when deploying a PHP site on the Internet.*

Troubleshooting the connection

Hopefully, everything went OK, but this section should help identify what might have gone wrong if you get an error message. Normally, you get a message about there being no testing server or saying that the testing server doesn't map to a particular URL.

All communication between Dreamweaver and MySQL is conducted through two files, `MMHTTPDB.php` and `mysql.php`, located in a hidden folder called `_mmServerScripts`. Dreamweaver automatically creates the hidden folder and files in the site root of your testing server. If you have defined the URL prefix incorrectly in your site definition, the folder will be in the wrong place. The solution is to use an Explorer window or Finder to see where the folder has been created. Then adjust the testing server site definition (see Chapter 2) so that both the testing server folder and URL prefix point to the site root.

If you're using your remote server as the testing server, Dreamweaver uploads the hidden folder and files to your remote server. Even if you have defined the URL prefix correctly, Dreamweaver might not be able to create the `_mmServerScripts` folder because of permission problems. Create the folder yourself, and make sure it has read and write permissions.

You may see a rather unhelpful message about an unidentified error. Things to check when this happens are that MySQL and your web server are running. Also check your username and password—both are case-sensitive and will fail if you use the wrong case (make sure Caps Lock isn't on by accident). A software firewall may also be blocking communication between Dreamweaver and MySQL. Try turning it off temporarily. If that solves the problem, adjust the firewall settings.

Inserting user details into the database

Although you can use phpMyAdmin or another graphical interface to insert records in a database table, it's more common to build a dedicated form to do so. Building dedicated forms gives you control over which parts of your database can be accessed by different people. The form you'll build over the next few pages is intended to be used by an administrator to control who has access to different parts of a website, but you could use a similar form in a public part of your site for users to register their details, such as email address.

There are two ways to create a form to insert records into a database table: either you can design your own form and use the Insert Record server behavior or you can use the Record Insertion Form Wizard to build the form and apply the server behavior in a single operation. Personally, I think the insert wizard creates ugly forms, but it offers a quick way to build a form to interact with a database. I'll use the wizard in this chapter, but in subsequent chapters, I'll show you how to adapt your own forms.

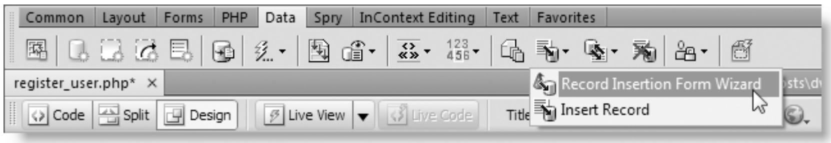
Using a wizard to build the registration form

These instructions step through the process of building a form to insert records in the `users` table. Continue working with `register_user.php` from the previous section.

1. Create another blank PHP page, and save it as `list_users.php` in `workfiles/ch14`. This will be used later to display a list of registered users. You don't need the page for the time being, so you can close it if you want.
2. Return to `register_user.php`. Give the page a title, such as Register User. Select Heading 1 from the Format menu in the HTML view of the Property inspector, and type the same heading at the top of the page. Then select the <h1> tag in the Tag selector at the bottom of the Document window, and press your right keyboard arrow to move the insertion point out of the heading. If you forget to do this, Dreamweaver embeds the entire form inside the `<h1>` tags.
3. Open the Data tab of the Insert bar, and locate the fifth icon from the right. It should display Insert Record as a tooltip. If this is the first time you have accessed this icon, clicking it opens a submenu, as shown in the following screenshot. Select Record Insertion Form Wizard from the submenu. On subsequent occasions, Dreamweaver remembers the option you used most recently, so you can just click the button.

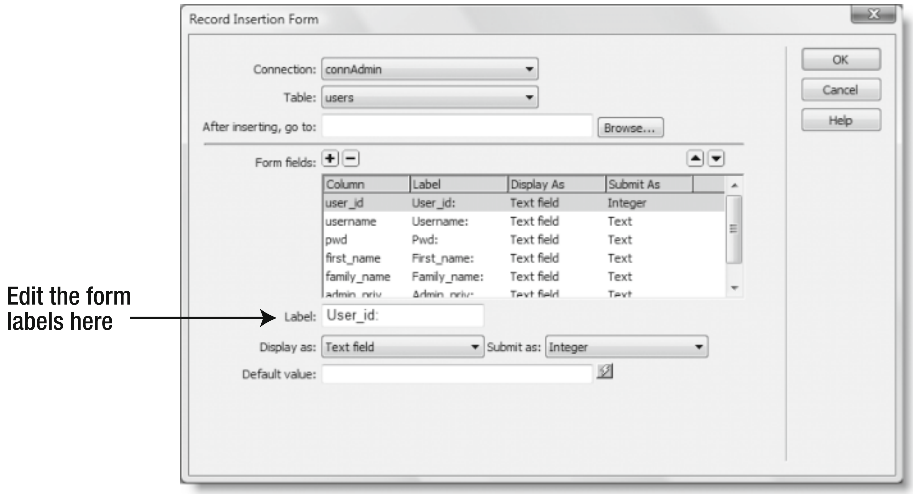
   If you're not sure whether you have used this option before, click the small down arrow alongside the icon to access the submenu directly.

   If you prefer working with the main menu system, use Insert ➤ Data Objects ➤ Insert Record ➤ Record Insertion Form Wizard.



4. This opens the Record Insertion Form dialog box (see Figure 14-10). When it first loads, you need to select a MySQL connection with `INSERT` privileges. If you created two user accounts for MySQL, use the administrator connection (`connAdmin`).

   This populates the Table drop-down menu with a list of tables in the database. They are listed in alphabetical order, so you need to select users. The dialog box then presents you with its suggested values for the record insertion form, as shown in Figure 14-10.



Connection: connAdmin
Table: users
After inserting, go to:
Form fields:

Column	Label	Display As	Submit As
user_id	User_id:	Text field	Integer
username	Username:	Text field	Text
pwd	Pwd:	Text field	Text
first_name	First_name:	Text field	Text
family_name	Family_name:	Text field	Text
admin_priv	Admin_priv:	Text field	Text

Label: User_id:
Display as: Text field Submit as: Integer
Default value:

Figure 14-10. The Record Insertion Form Wizard helps build the insertion form automatically.

5. Dreamweaver uses the table column names to suggest labels and appropriate types of input fields for the form. The columns are listed in the same order as they appear in the database, but you can use the up and down arrow buttons at the top right of the Form fields area to rearrange the order they will be displayed in the record insertion form. If you don't want to display a particular field, remove it by clicking the minus button. To restore a deleted item, click the plus button, and select it from the list.

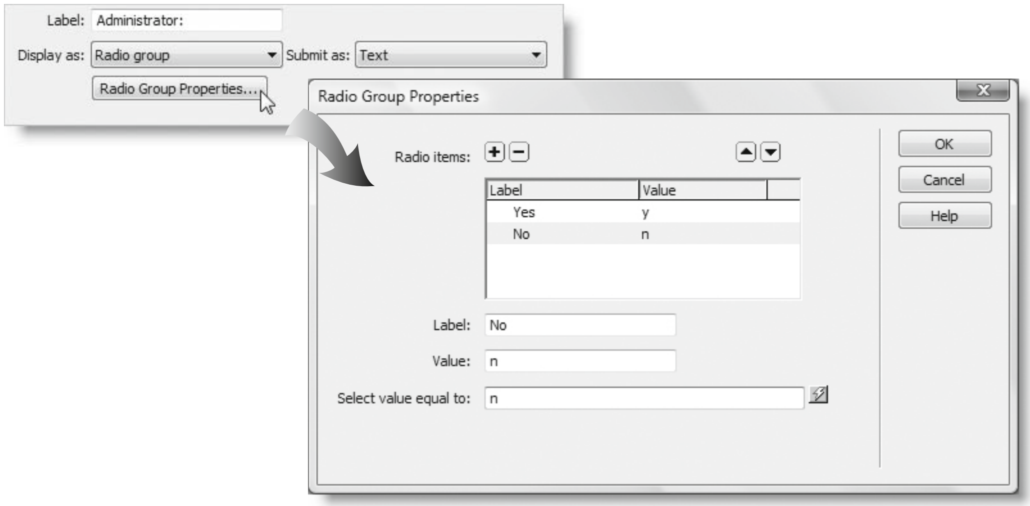
   You can also specify where you want to go to after the record has been inserted. If you leave the option blank, the same page will be redisplayed ready for another record.

6. The primary key is generated automatically, so you don't want a field for it in the form. Select user_id in the Form fields area, and click the minus button to delete it.

7. The suggested labels for the `pwd`, `first_name`, `family_name`, and `admin_priv` columns all need amending. Select each one in turn, and edit the value in the Label field (see Figure 14-10). Expand Pwd: to Password: and change the value of Display as to Password field; remove the underscore from First_name: and Family_name:, and change admin_priv to Administrator:.

8. The `admin_priv` column uses the `ENUM` column type, so you want to use a radio button group. With admin_priv selected in Form fields, change Display as to Radio group, and then click the Radio Group Properties button that appears. This opens the Radio Group Properties dialog box. Use the plus button to create two Radio items: Yes with a value of y, and No with a value of n, as shown in the following screenshot. These match the values defined in the `ENUM` column in the database

table. To make No the default value, enter n in the field labeled Select value equal to, and click OK.



Label: Administrator:
Display as: Radio group
Submit as: Text

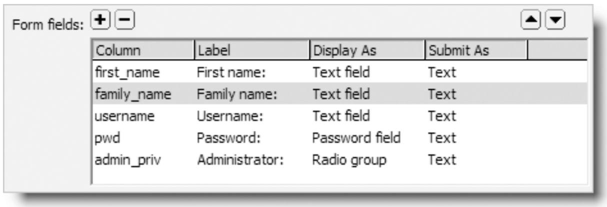
Radio Group Properties

Radio items:

Label	Value
Yes	y
No	n

Label: No
Value: n
Select value equal to: n

9. Reorder the items with the up and down arrows at the top right of the Form fields area so that they look like this:



Form fields:

Column	Label	Display As	Submit As
first_name	First name:	Text field	Text
family_name	Family name:	Text field	Text
username	Username:	Text field	Text
pwd	Password:	Password field	Text
admin_priv	Administrator:	Radio group	Text

10. Click the Browse button alongside the field labeled After inserting, go to. Navigate to `list_users.php`, and select it. This will redirect the user to `list_users.php` after a record has been inserted in the database table.
11. Click OK to create the form. In Design view, the page should now look like Figure 14-11. The form's light blue coloring indicates that it contains dynamic code.

Once you click OK *in the* Record Insertion Form *dialog box, you cannot reopen it to make any changes. All further changes to the form need to be made in the Document window. If you want to start afresh, use the minus button in the* Server Behaviors *panel to remove the Insert Record code before deleting the form. Otherwise, you'll end up with a tangle of impossible code.*

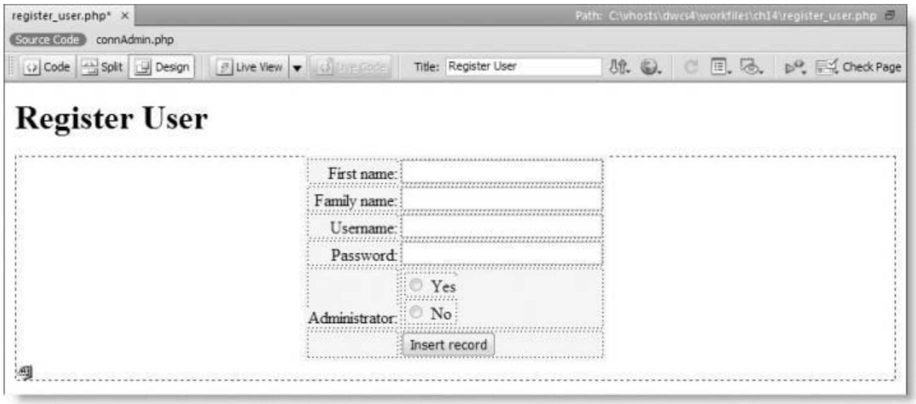
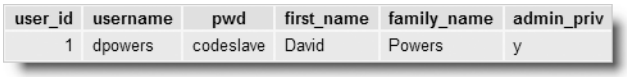


Figure 14-11. The Record Insertion Form Wizard creates the form and the necessary PHP code in a single operation.

12. Save `register_user.php`, and load it into a browser (don't use Preview in Browser with a temporary file). Enter some details in each field, and click Insert Record. Don't worry if you see a blank page; you should have been taken to `list_users.php`, which doesn't yet contain anything.

13. Launch phpMyAdmin, select the `dwcs4` database, and then select the `users` table. When you click the Browse tab, you should see the details of the record you just inserted listed like this:



user_id	username	pwd	first_name	family_name	admin_priv
1	dpowers	codeslave	David	Powers	y

Compare your code, if necessary, with `register_user.php` in `examples/ch14`.

It's as easy as that!

Before you start celebrating too soon, I should warn you that there are lots of things wrong with this registration form. The password is stored in the database in plain text, which is insecure. Also, the form is incapable of handling the database error if someone chooses the same username as another person. In fact, there's nothing to prevent someone from entering a single space in each field and registering that. This form is functional, but a lot of work still needs to be done on it.

We'll come back to server-side validation in the next chapter. The next stage is to create a page that displays a list of entries in the database table, but first a quick word about things that might have gone wrong:

- If you get a string of errors about `mysql_real_escape_string()` and the ODBC connection, it means you have used Preview in Browser with a temporary file. Load the actual page into a browser, and try again.

- If you get a fatal error about a call to undefined function `virtual()`, it means your site defaults to links relative to the site root and you're not using Apache as the web server. See the next section.

Using server behaviors with site-root-relative links

If you open `register_user.php` in Code view to see the PHP code that Dreamweaver has added to the page and you use document-relative links, the top section will look like this:

```
<?php require_once('../../Connections/connAdmin.php'); ?>
```

This code uses `require_once()` to include the MySQL connection details. However, if your site definition uses links relative to the site root, this will be replaced by the following:

```
<?php virtual('/Connections/connAdmin.php'); ?>
```

The `virtual()` function *works only on Apache*. If your code uses `virtual()`, make sure it is supported on both your testing and remote servers before going any further (see "Using site-root-relative links with includes" in Chapter 12 for details of how to do this).

All Dreamweaver server behaviors need to include the MySQL connection. If your server doesn't support `virtual()`, you have two options, namely:

- Change your site definition to use document-relative links, and manually override the default when creating links that you want to be relative to the site root. You do this in the Select File dialog box by changing the Relative to drop-down menu to Site Root, as described in "Including a text file" in Chapter 12.
- Manually replace `virtual()` with `require_once()` and a document-relative link in pages that use server behaviors. The `require_once()` command works on all servers.

Neither solution is ideal. I believe that Dreamweaver needs a platform-neutral way of connecting to MySQL when site-root-relative links are used, or it should use `require_once()` regardless of the default link type.

Retrieving information from the database

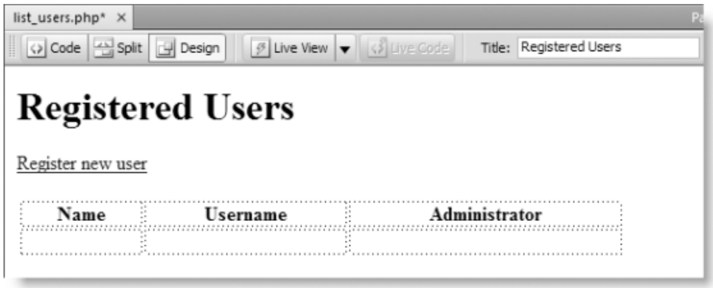
Inserting information into a database is fine, but there's not much point unless you can retrieve it and do something useful with it. Retrieving information from a database involves creating a SQL `SELECT` query. As the name suggests, it selects information from the database according to your criteria and returns the results. Dreamweaver calls this a **recordset**. Once you have created a recordset, you can use it to display the results of the query in a web page. Although there's currently only one record in the `users` table, let's build a page to display a list of all registered users, because this is an essential prerequisite to being able to update and delete information stored in the database.

Creating a recordset

These instructions show you how to use the Recordset dialog box in Simple mode to query the users table in preparation for displaying the results in a web page.

1. Open list_users.php, and give it a title and heading, such as Registered Users. Insert a link to register_user.php and a table with two rows and three columns. I made the table 500 pixels wide, with no border, cellpadding, or cellspacing. I also set Header to Top.

   Type Name, Username, and Administrator in the first row. The page should look like this:



Registered Users

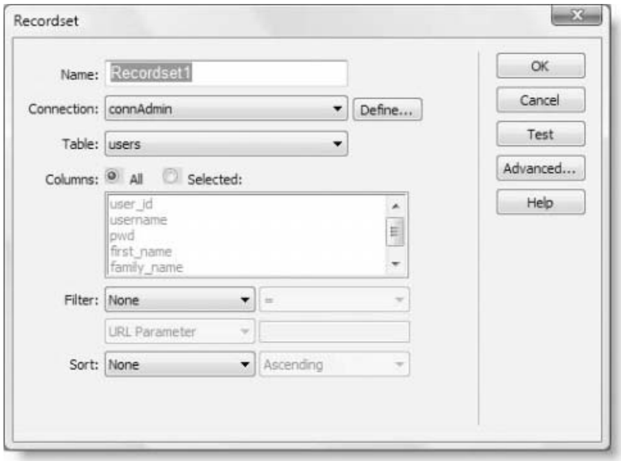
Register new user

Name	Username	Administrator

2. Open the Recordset dialog box by clicking the plus button in the Server Behaviors panel and selecting Recordset. The Server Behaviors panel is normally grouped with the Database and Bindings panels. If you can't see it, select Window ➤ Server Behaviors to open it, or press Ctrl+F9/Cmd+F9.

   You can also click the Recordset button on the Data tab of the Insert bar or select Insert ➤ Data Objects ➤ Recordset.

   The Recordset dialog box has two modes: Simple and Advanced. If this is the first time you have opened the dialog box, it will be in Simple mode, as shown in Figure 14-12.



Name: Recordset1
Connection: connAdmin
Table: users
Columns: All [✓] Selected:
user_id
username
pwd
first_name
family_name
Filter: None =
URL Parameter
Sort: None Ascending

Figure 14-12. The Recordset dialog box in Simple mode is used for basic SELECT queries.

You can tell which mode you're in by looking at the buttons on the right side of the dialog box. If you're in Simple mode, the fourth button is labeled Advanced; and if you're in Advanced mode, it's labeled Simple (because it switches to the opposite mode).

3. By default, Dreamweaver enters a generic value such as Recordset1, Recordset2, and so on, in the Name field. However, the name is used to create several PHP variables, so it's better to choose something that tells you what the recordset is for. Use only letters, numbers, and the underscore. Don't use any spaces. Some people use the convention of beginning recordset names with rs, but this isn't necessary. The name I have chosen is listUsers.

4. Dreamweaver CS4 now remembers the most recent connection you used, so the `connAdmin` connection is automatically selected. Although this recordset performs only a SELECT operation, you'll be editing the records later, so it's more consistent to use the administrator connection for all the pages.

5. There's only one table in the database at the moment, so the `users` table is also selected automatically.

6. The Columns field has two radio buttons: All and Selected. By default, the All radio button is selected, and the columns are grayed out. A lot of beginners select All every time, even if they need only one or two columns. It's easy, and it makes the SQL query a lot easier to read (we'll study SQL syntax in Chapter 16). However, it's a bad habit. Even if you need all columns, it's considered best practice to select them individually because it makes the meaning of your code much clearer.

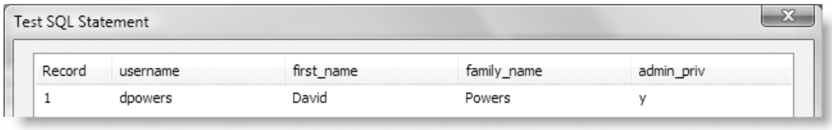
   Choose the Selected radio button, and Ctrl-click/Cmd-click username, first_name, family_name, and admin_priv.

7. You can ignore the Filter settings this time. I'll explain their use later.

8. Although there's only one record at the moment, it's a good idea to decide how the results should be sorted when there are more records in the table. Open the drop-down menu labeled Sort. It lists all the columns in the table. Choose family_name. This enables the drop-down menu to the right. It has two options: Ascending and Descending. Select Ascending. This will sort all results by the family name in alphabetical order.

   In Simple mode, you can sort by only one column. In Chapter 16, I'll show you how to use Advanced mode to sort by multiple columns.

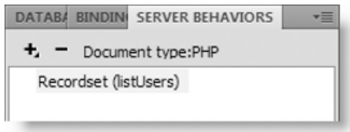
9. Click the Test button on the right of the Recordset dialog box. This opens the Test SQL Statement panel with the results of the query, as shown here:



Test SQL Statement

Record	username	first_name	family_name	admin_priv
1	dpowers	David	Powers	y

10. Click OK to close the test panel, and then click OK again to close the Recordset dialog box and create the recordset.
11. The `listUsers` recordset should now be listed at the top of the Server Behaviors panel, as shown in the following screenshot:



Recordset (listUsers)

12. Save `list_users.php`. Leave the page open ready to insert the code that will display the results of the recordset. If you want to check your page so far, compare it with `list_users_01.php` in `examples/ch14`.

Editing and removing server behaviors

Whenever you create a recordset or apply a server behavior, Dreamweaver adds it to the list in the Server Behaviors panel (some server behaviors add several items to the list). If you need to edit a server behavior, double-click its listing in the Server Behaviors panel to reopen its dialog box. To delete a server behavior, *always* select it from this list and click the minus button at the top of the panel. Failure to do so will result in code that is likely to behave erratically.

Dreamweaver creates a lot of PHP code behind the scenes when working with server behaviors. You'll examine a lot of it in coming chapters to get to know what it's for. Until you understand the code, it's dangerous to highlight PHP elements in Design view and press Delete. Using the minus button in the Server Behaviors *panel removes the code cleanly.*

Displaying the results of a recordset

Once you have created a recordset, Dreamweaver makes its results available through the Bindings panel, which displays a list of the database columns retrieved by each recordset you create. You use the panel to insert PHP code into your web page and display the results of the database query.

Creating the list of registered users

The following instructions show how to insert dynamic text from the Bindings panel and display the results of the `listUsers` recordset. Continue working with `list_users.php` from the previous section.

1. Open the Bindings panel by selecting its tab, selecting Window ➤ Bindings, or pressing the keyboard shortcut Ctrl+F10/Cmd+F10.

2. Expand the `listUsers` recordset as shown in Figure 14-13 by clicking to the left of the icon alongside Recordset (listUsers).

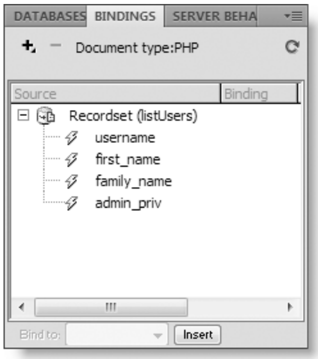
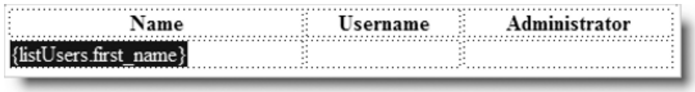


Figure 14-13. The Bindings panel gives access to the query result.

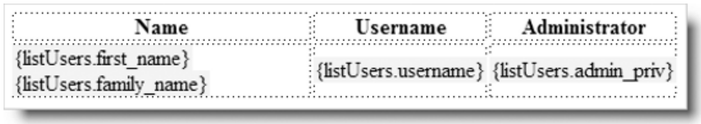
3. To insert the database results into a page, you can drag the column names from the Bindings panel into the Document window. Alternatively, position your cursor in the Document window where you want to display the result, select the column name in the Bindings panel, and click the Insert button at the bottom of the panel.

   Use either method to insert first_name from the Bindings panel into the first cell of the second row of the table. This inserts a dynamic text placeholder in the page like this:



Name	Username	Administrator
{listUsers.first_name}		

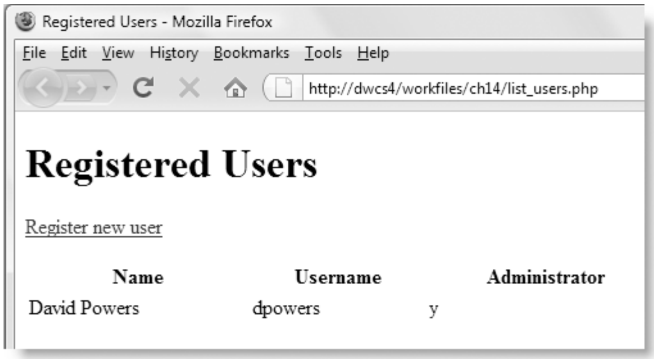
4. Click to the right of the dynamic text placeholder, and insert a space. Then insert family_name from the Bindings panel alongside. Insert username into the second cell of the second row, and insert admin_priv into the third cell. The table should now look like this:



Name	Username	Administrator
{listUsers.first_name} {listUsers.family_name}	{listUsers.username}	{listUsers.admin_priv}

   It doesn't matter if the dynamic text placeholders in the first cell stack on top of each other like this, because the placeholders are longer than the actual text that will be displayed.

5. Save list_users.php, and press F12/Opt+F12 to load it into a browser. You should see the details of the record you inserted into the users table displayed in the page like this:


Registered Users - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dwcs4/workfiles/ch14/list_users.php

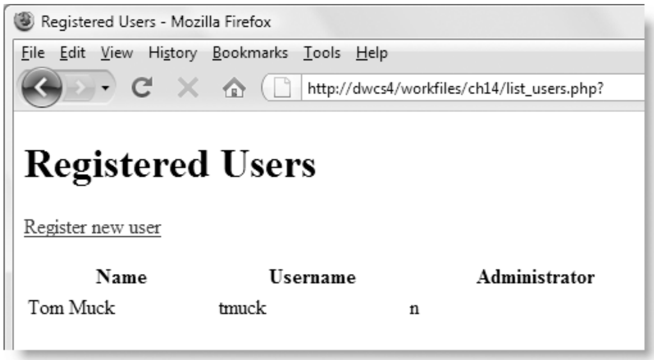
Registered Users

Register new user

Name	Username	Administrator
David Powers	dpowers	y

Don't worry about the way the page looks. That's cosmetic and can be easily fixed with CSS. At the moment, I just want to concentrate on working with the server behaviors. If anything went wrong, compare your code with list_users_02.php in examples/ch14.

6. Assuming everything went OK, click the Register new user link to go to register_user.php, and enter a new record in the database. Choose a family name that comes before the existing record in alphabetical order. When you click the Insert record button in register_user.php, list_users.php should automatically load and display the new name, as shown here:


Registered Users - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dwcs4/workfiles/ch14/list_users.php?

Registered Users

Register new user

Name	Username	Administrator
Tom Muck	tmuck	n

This is progress. The results have been sorted in alphabetical order, but only the first result is displayed. Keep list_users.php open in Dreamweaver, and we'll fix that next.

Displaying multiple results with a repeat region

Unless you explicitly limit the results of a database query (you'll learn how to do that later in the book), a recordset contains all the records in the database that match your search criteria. However, the dynamic text placeholders inserted from the Bindings panel display only the first result. To display the remaining ones, you need to apply the Repeat Region server behavior. Let's do that now so you can see both results.

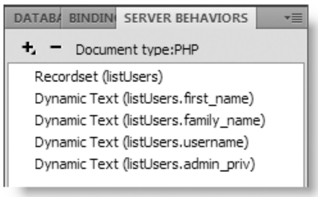
Adding a Repeat Region server behavior

These instructions show you how to display multiple results from a recordset by applying a Repeat Region server behavior to the table row that contains the dynamic text placeholders. Continue working with `list_users.php` from the preceding section.

1. Position your cursor inside the second table row in `list_users.php`, and click <tr> in the Tag selector at the bottom of the Document window to select the entire row.

It's important to select the entire row, including the opening and closing `<tr>` tags. If you simply drag across the table cells to select them in Design view, there's a danger that Dreamweaver will select only the `<td>` tags. Using the Tag selector ensures you get the correct selection every time.

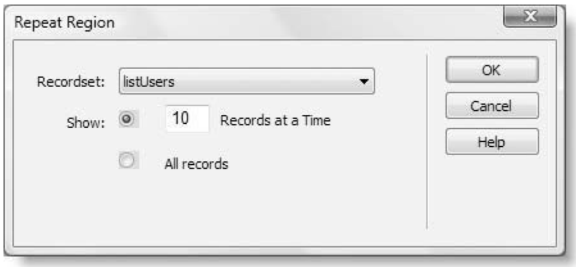
2. Open the Server Behaviors panel. Note that the four instances of dynamic text have been added to the list, as shown here:



Recordset (listUsers)
Dynamic Text (listUsers.first_name)
Dynamic Text (listUsers.family_name)
Dynamic Text (listUsers.username)
Dynamic Text (listUsers.admin_priv)

3. Click the plus button in the Server Behaviors panel, and select Repeat Region from the menu that appears. Alternatively, click the Repeat Region button on the Data tab of the Insert panel, or use the menu option, Insert ➤ Data Objects ➤ Repeat Region.

   This opens the Repeat Region dialog box shown in Figure 14-14.



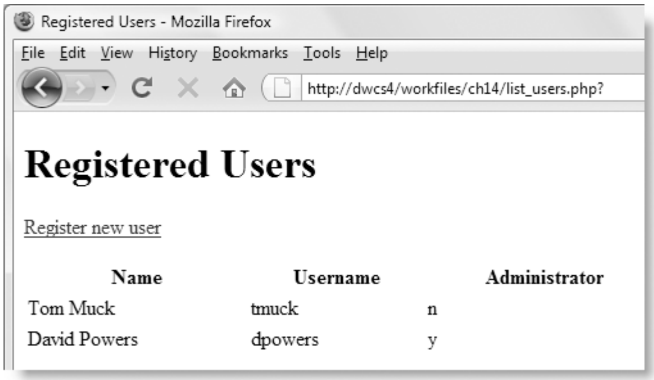
Recordset: listUsers
Show: (●) 10 Records at a Time
( ) All records

Figure 14-14. A repeat region can display a selected number of records or all of them.

4. There's only one recordset on the page, so the Repeat Region dialog box automatically selects `listUsers`. The Show option lets you choose whether to show a limited number of records or all of them. The default is to show a maximum of ten records, but you can change this by entering your own value in the text field.

You'll learn in Chapter 16 how to page through a long recordset several records at a time. On this occasion, though, select All records, and click OK to apply the repeat region.

5. Save `list_users.php`, and reload it in a browser. You should now see both records listed, as shown in Figure 14-15.



Registered Users - Mozilla Firefox

http://dwcs4/workfiles/ch14/list_users.php?

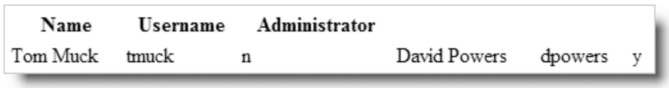
Registered Users

Register new user

Name	Username	Administrator
Tom Muck	tmuck	n
David Powers	dpowers	y

Figure 14-15.
The repeat region now displays all records in the table.

If the results end up being displayed across the page as shown in the following screenshot, it means that you failed to select the entire table row in step 1:



Name	Username	Administrator			
Tom Muck	tmuck	n	David Powers	dpowers	y

If this happened to you, select Repeat Region (listUsers) in the Server Behaviors panel, click the minus button to remove it cleanly, and start again from step 1.

Check your code, if necessary, against `list_users_03.php` in `examples/ch14`. You'll improve the page further in the next section, so keep it open in the Document window.

Updating and deleting records

To update a record in a database, you need to populate a form with the existing details so they can be edited and reinserted into the database. This is where a table's primary key plays a vital role. If you know the primary key of a record, you can easily retrieve it and populate the update form. Equally important, you can use the primary key to delete a record when it's no longer wanted. So, how do you find the primary key? Simple . . . Look it up in the database, and store it in a link that loads the update form or triggers the delete mechanism.

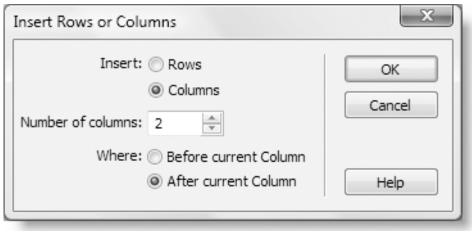
The `listUsers` recordset that you created earlier retrieves the `first_name`, `family_name`, `username`, and `admin_priv` columns. All you need to do is to edit the recordset to get it to retrieve the primary key, `user_id`, as well. You can then use that information to create edit and delete links in `list_users.php`.

Time to get back to work . . . .

Adding a record's primary key to a query string

These instructions show you how to edit the `listUsers` recordset to retrieve the primary key of each record and then incorporate the primary key into links that will be used to update and delete individual records.

1. Create two new blank PHP pages called `update_user.php` and `delete_user.php`. You don't need them for the time being, so you can close them if you want.
2. You need to add two columns on the right of the table in `list_users.php`. The easiest way to do this is to right-click in the last column and select Table ➤ Insert Row or Columns from the context menu. In the dialog box that opens, select the Columns radio button, set Number of columns to 2, and select After current Column, as shown here:



Insert Rows or Columns

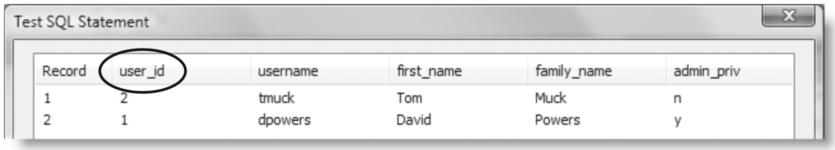
Insert: Rows [✓ Columns]

Number of columns: 2

Where: Before current Column [✓ After current Column]

3. When you click OK to insert the extra columns, you might find it difficult to insert your cursor in the new cells. To make it easier to work in the table, turn on Expanded Tables mode by pressing Alt+F6/Opt+F6 (you can also click the Expanded button on the Layout tab of the Insert bar or select View ➤ Table Mode ➤ Expanded Tables Mode.

   Type EDIT in the fourth cell of the second row, and type DELETE in the final cell. Once you have entered the text in the new cells, you can exit Expanded Tables mode by clicking Exit at the top of the Document window.
4. Open the Server Behaviors panel, and double-click Recordset (listUsers) to open the Recordset dialog box. Edit the settings by holding down the Ctrl/Cmd key and selecting user_id in the Columns field.
5. Click the Test button to make sure the query now includes the `user_id` primary key, as shown in the following screenshot:



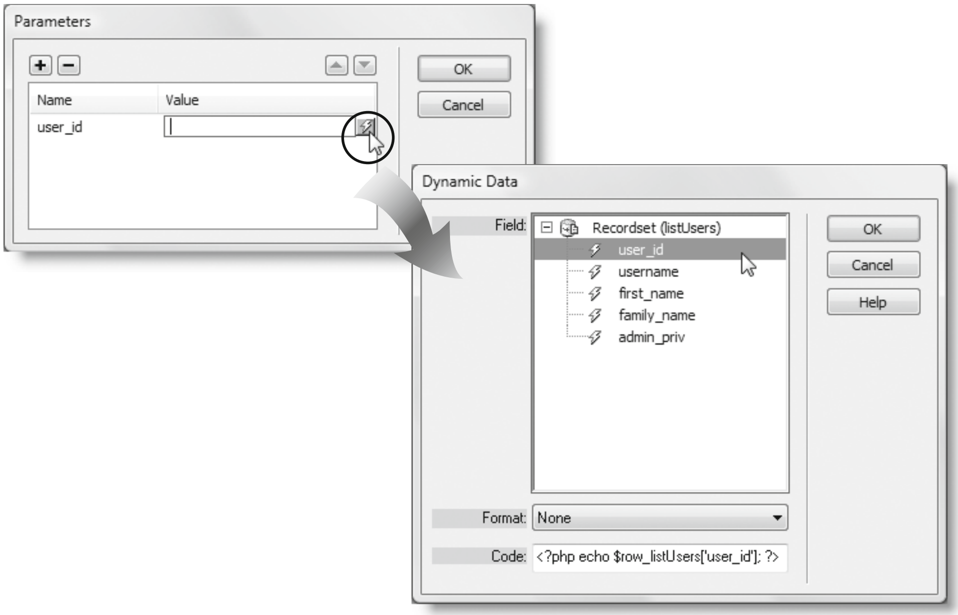
Test SQL Statement

Record	user_id	username	first_name	family_name	admin_priv
1	2	tmuck	Tom	Muck	n
2	1	dpowers	David	Powers	y

6. Close the test panel, and save the amended recordset.
7. Select the text in the fourth cell (EDIT). You need to turn it into a link to the update page and add the record's primary key to a query string at the end of the URL.

Begin by clicking the Browse for File button to the right of the Link field in the HTML view of the Property inspector. In the Select File dialog box that opens, select `update_user.php`. Then click the Parameters button alongside the URL field.

8. In the Parameters dialog box, type user_id in the Name field. Click the lightning bolt icon on the right of the Value field. In the Dynamic Data dialog box, highlight user_id, as shown in Figure 14-16.



```
<?php echo $row_listUsers['user_id']; ?>
```

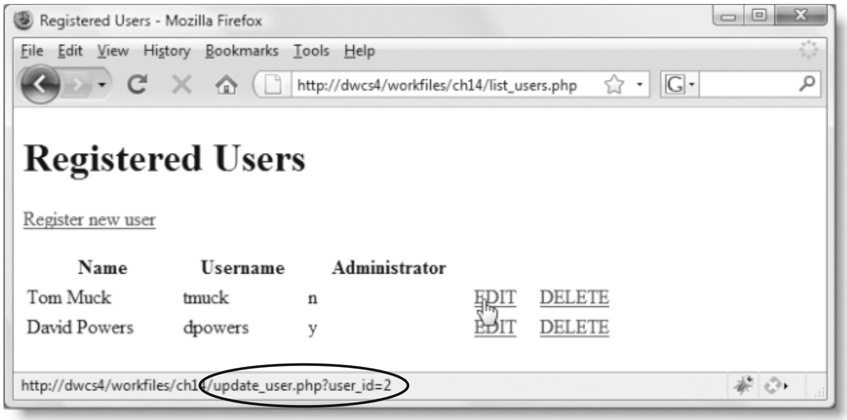
Figure 14-16. The Parameters and Dynamic Data dialog boxes build the query string.

This is where many people go wrong. The Name *field in the* Parameters *dialog box takes a* static *value, which you type in yourself. The* Value *field takes a* dynamic *value, which you insert by clicking the lightning bolt icon and selecting the primary key from the* Dynamic Data *dialog box.*

9. Click OK to close both the Dynamic Data and Parameters dialog boxes. Then click OK (Choose on the Mac) to close the Select File dialog box.

10. Repeat steps 7 through 9 with the text in the fifth cell (DELETE). In step 7, select `delete_user.php`.

11. Save `list_users.php`, and preview it in a browser. Mouse over the EDIT and DELETE links. The status bar of your browser should display links to `update_user.php` and `delete_user.php` and have a query string containing `user_id` and the user's primary key, as shown in Figure 14-17.


Registered Users - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dwcs4/workfiles/ch14/list_users.php

Registered Users

Register new user

Name	Username	Administrator		
Tom Muck	tmuck	n	EDIT	DELETE
David Powers	dpowers	y	EDIT	DELETE

http://dwcs4/workfiles/ch14/update_user.php?user_id=2

Figure 14-17. The query string has been added to the URL.

Make sure the query string is correctly formed at the end of the URL when you mouse over the links. This is very important; the update and delete pages won't work unless the query string displays user_id= followed by a number. Review steps 7–10 again if the URL doesn't look right. If necessary, check your code against `list_users_04.php` in `examples/ch14`.

Retrieving a database record using its primary key

The wizard that builds update forms is almost identical to the one that inserts new records into a database. The main difference is that you can't use it until you have already created a recordset to retrieve the existing details of the record you want to update. So, the process involves three basic steps, namely:

1. Create a recordset to retrieve a single record.
2. Display the results of the recordset in a form ready for editing.
3. Submit the edited information to update the existing record.

Since each record has a unique primary key, you can retrieve the details of a specific record by using its primary key as a filter. You don't need to know the actual primary key to create the recordset. All you need to know is the name of the variable containing the primary key and where it's coming from. The query string you added at the end of the URL in the EDIT link of `list_users.php` in the previous section contains the individual record's primary key as a variable called `user_id`. So, this is the value you use to filter the recordset.

Using the primary key to filter a recordset

The following instructions show you how to retrieve a record identified by its primary key:

1. Open `update_user.php`. Give the page a suitable title and a heading, such as Update User Record.

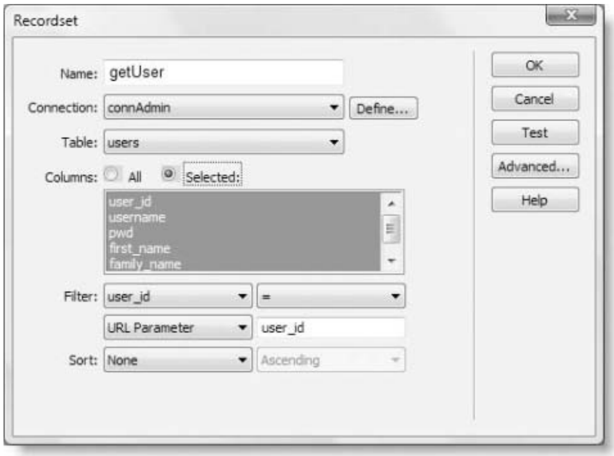
2. Click the plus button in the Server Behaviors panel, and select Recordset from the menu that appears. The Recordset dialog box should still be in Simple mode from the last time you used it.
3. Name the recordset getUser.
4. Check that the Connection field has been set to connAdmin (Dreamweaver should remember from the most recent time you used it).
5. Since you have only one table, users should be automatically selected. Click the Selected radio button, and Shift-click all the columns.
6. The Filter section consists of three drop-down menus and a text field. The first drop-down menu lists all the columns in the users table. You want to use the primary key to select the record, so choose user_id from the list.

   The drop-down menu on the right contains a range of comparison operators that determine how the filter is used. You want the variable passed through the query string to be equal to the user_id column, so the default, =, is fine.

   The drop-down menu in the second row determines where the value comes from. On this occasion, it's being passed in through a query string, so select URL Parameter.

   The text field on the right of the second row is where you enter the name of the variable whose value you want to match in the selected column. Dreamweaver assumes you want to use the same name as the column, which is why I used user_id in the query string. If you use a different variable name, you can type it in here, but it's not necessary on this occasion.

   The settings in the Recordset dialog box should now look like this:



Recordset

Name: getUser
Connection: connAdmin
Table: users
Columns: All [✓ Selected:]
user_id
username
pwd
first_name
family_name
Filter: user_id =
URL Parameter user_id
Sort: None Ascending

7. Click the Test button. Because you're filtering the recordset, Dreamweaver asks you to provide a test value for user_id. Assuming you haven't deleted any records in the table, enter 1 or 2, and click OK. You should see details of the record that has that primary key.
8. Click OK to dismiss the test panel, and then click OK again to save the recordset. You're now ready to create the update form. Leave update_user.php open to continue with the next section.

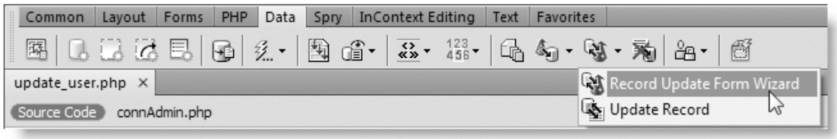
Using the Record Update Form Wizard

Now that you have a recordset that retrieves the details of the record you want to update, you can use the Record Update Form Wizard (if you attempt to use the wizard without first creating the recordset, Dreamweaver displays a message telling you to do so).

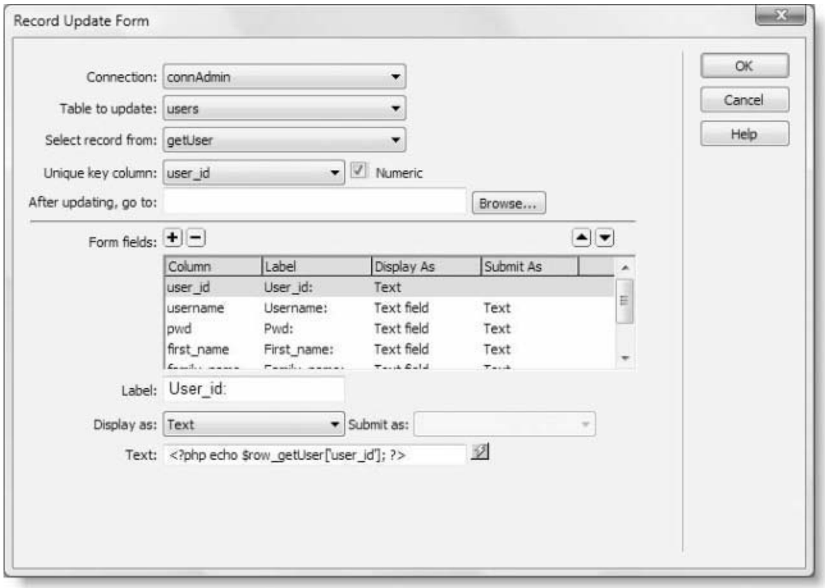
Building the update form

You use the wizard in almost the same way as the one that created the form to insert new records. Here's how . . . .

1. You need to make sure the wizard builds the update form outside the `<h1>` tags of the page heading in `update_user.php`. So, insert your cursor in the page heading, select <h1> in the Tag selector, and press your right arrow key once.
2. Select Record Update Form Wizard in the Data tab of the Insert bar (it's the fourth icon from the right), as shown in the following screenshot. Alternatively, use the menu option, Insert ➤ Data Objects ➤ Update Record ➤ Record Update Form Wizard.



3. This opens the Record Update Form dialog box shown in Figure 14-18. It's very similar to the dialog box used to create the form for new records, but there are some important differences. The wizard recognizes the `getUser` recordset you created in the previous section and automatically fills most fields to use it.



Connection: connAdmin
Table to update: users
Select record from: getUser
Unique key column: user_id ✓ Numeric
After updating, go to:
Form fields:

Column	Label	Display As	Submit As
user_id	User_id:	Text	
username	Username:	Text field	Text
pwd	Pwd:	Text field	Text
first_name	First_name:	Text field	Text

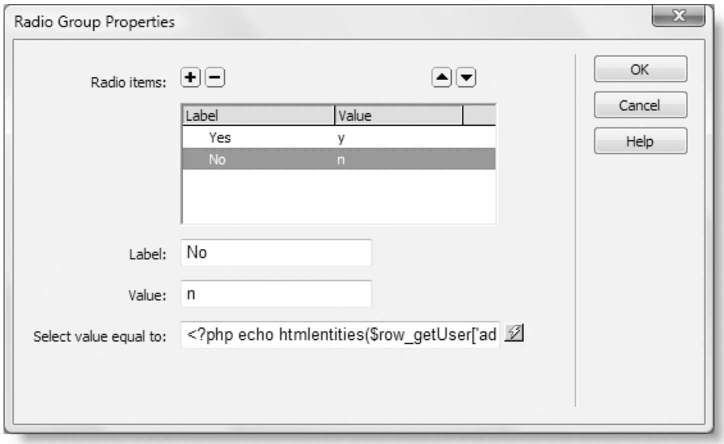
Label: User_id:
Display as: Text Submit as:
Text: <?php echo $row_getUser['user_id']; ?>

Figure 14-18. The update wizard automatically populates the dialog box with most options.

The first four options (Connection, Table to update, Select record from, and Unique key column) already have the correct details and don't need to be changed. The Unique key column refers to the primary key you're using to identify the correct record to update. The Numeric checkbox alongside is selected because the `user_id` column was defined as an `INT` type when you built the `users` table earlier in the chapter. This is an important security check that Dreamweaver makes to help protect your database from malicious attack.

4. The field labeled After updating, go to is where you want to redirect the user after the record has been updated. Click the Browse button, navigate to `list_users.php`, and select it.
5. Form fields lists all the columns retrieved in the `getUser` recordset. You use it in the same way as when you built the form to insert new records. You should never change the primary key of a record, so select user_id in Form fields, and click the minus button to remove it from the list.
6. Amend the labels for the `pwd`, `first_name`, `family_name`, and `admin_priv` columns in the same way as before by selecting each one and editing it in the Label field. Expand Pwd to Password, remove the underscores from First_name and Family_name, and change Admin_priv to Administrator. Use the up key at the top right of Form fields to move the first and family name items to the top of the list.

   Notice that the Text field at the bottom of the dialog box is automatically populated with PHP code. This uses the `getUser` recordset results to display the record's existing value in the update form.
7. Leave the Display as drop-down menu set to Text for all columns except `admin_priv`.
8. Select admin_priv in Form fields, and select Radio group from the Display as drop-down menu. Click the Radio Group Properties button to open the following dialog box:



Label	Value
Yes	y
No	n

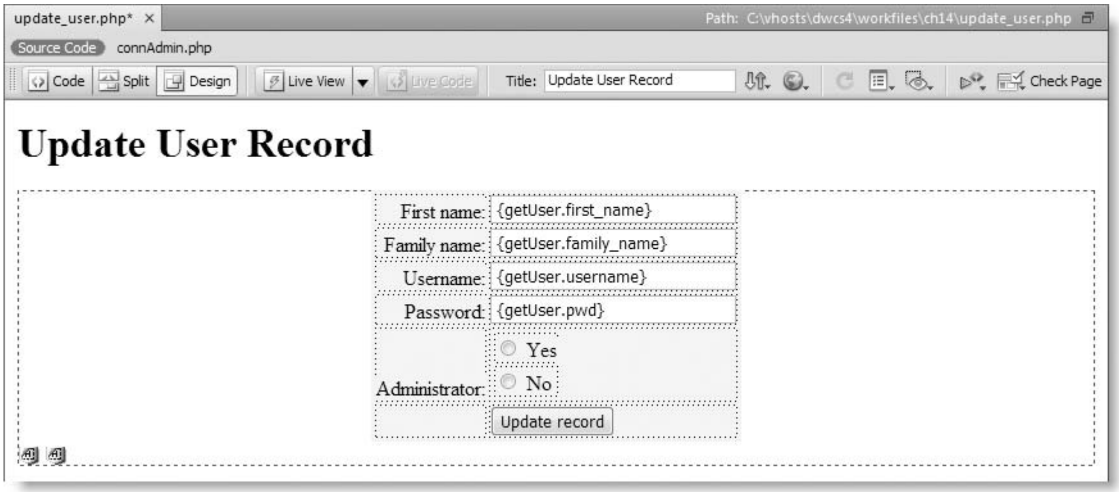
Label: No

Value: n

Select value equal to: <?php echo htmlentities($row_getUser['ad

Set the values for the two radio buttons as shown in the screenshot. Notice that, this time, you don't need to fill in the field labeled Select value equal to. Dreamweaver automatically populates this field with PHP code to select the correct radio button according to the value stored in the `getUser` recordset. Click OK to close the Radio Group Properties dialog box.

9. The update wizard is like the one you used earlier—once you click OK to close the Record Update Form dialog box, there's no way to reopen it to edit it. Check that you have made all the necessary changes, and click OK to create the update form, which should look like Figure 14-19. The form is basically the same as the one used to insert a new record, but each field contains a dynamic text placeholder that uses the `getUser` recordset to insert the existing value stored in the database. At the bottom left of the update form, you should see two gold shields. These are hidden form fields that Dreamweaver has created to send the record's primary key and details of the form to the update script.



Title: Update User Record

Update User Record

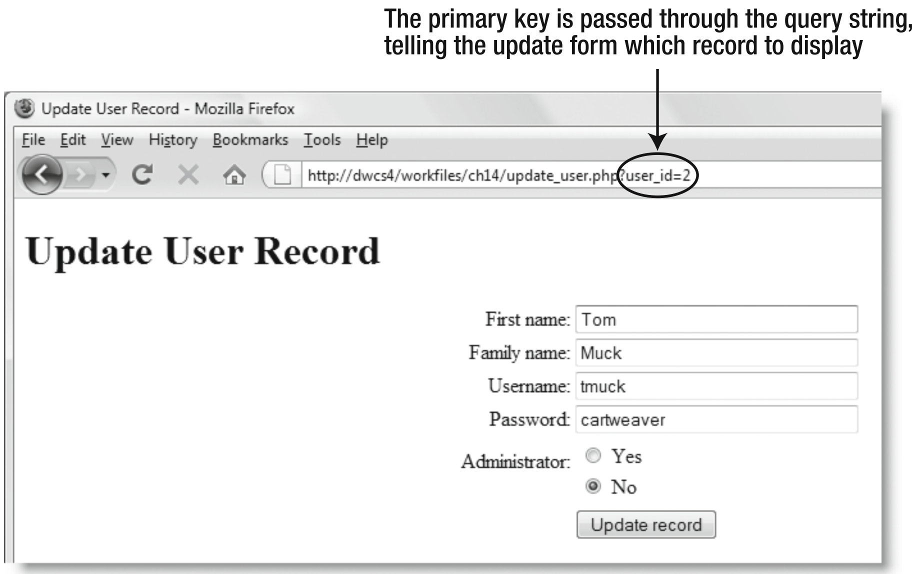
First name:	{getUser.first_name}
Family name:	{getUser.family_name}
Username:	{getUser.username}
Password:	{getUser.pwd}
Administrator:	○ Yes ○ No
	Update record

Figure 14-19. The update form contains dynamic text placeholders to display the record's details.

If you can't see the gold shields, make sure there's a check mark alongside Invisible Fields *in* View ➤ Visual Aids. *Also check the* Invisible Elements *category in the Dreamweaver* Preferences *panel (*Edit ➤ Preferences *or* Dreamweaver ➤ Preferences *on a Mac). Make sure that the* Hidden form fields *checkbox is selected.*

10. Save `update_user.php`. You now need to test the update form, but if you load the page directly into a browser, you'll get an empty form. Even if you fill it in, it won't create a new record because the underlying code uses the SQL `UPDATE` command, rather than `INSERT`. Without a primary key, there's nothing to update. So, to test the page, you need to load `list_users.php` into a browser.

11. Click the EDIT link alongside one of the records. The update form should load into the browser with the record's details ready for updating, as shown in Figure 14-20.


The primary key is passed through the query string, telling the update form which record to display
Update User Record - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dwcs4/workfiles/ch14/update_user.php?user_id=2
Update record

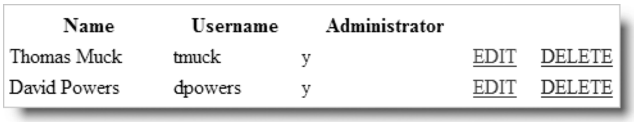
Update User Record

First name:	Tom
Family name:	Muck
Username:	tmuck
Password:	cartweaver
Administrator:	Yes [✓ No]

Figure 14-20. The update form displays the existing details ready for editing.

If the form is empty, check that the query string has been added correctly to the end of the URL. Also check the spelling of the variable in the query string. It must match exactly the way you spelled the primary key in the users table.

12. Make some changes to the record, such as changing the first name and administrator status, and click Update record. The amended details (apart from the password) are displayed immediately in list_users.php, as shown here:



Name	Username	Administrator		
Thomas Muck	tmuck	y	EDIT	DELETE
David Powers	dpowers	y	EDIT	DELETE

Check your code, if necessary, against update_users.php in examples/ch14.

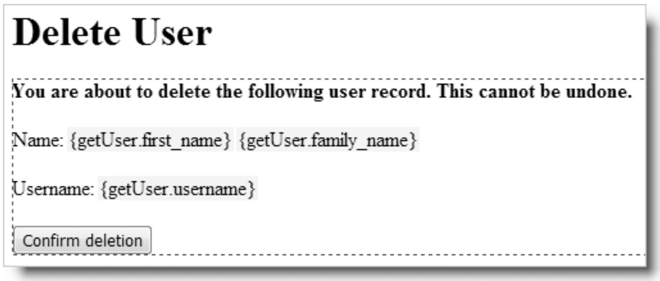
Deleting a record

Deleting a record is an irreversible action, so it's essential to get confirmation not only that the deletion should go ahead but also that the correct record is being deleted. There isn't a wizard for creating a delete page, but it's not difficult to build the page yourself. As when updating a record, you need a recordset to identify whether Dreamweaver can apply the necessary server behavior. The good news is that you can save time by copying the recordset from the update page.

Building the delete page

These instructions show you how to create a page that asks the user for confirmation before deleting a record from the database. They assume you have created `list_users.php`, `update_user.php`, and `delete_user.php` from the preceding sections. You should also be familiar with form building techniques (see Chapter 9 if you need to refresh your memory).

1. Open `delete_user.php` in the Document window.
2. Open `update_user.php`, or switch to it if it's still open.
3. In the Server Behaviors panel, highlight Recordset (getUser), right-click, and select Copy from the context menu.
4. Switch back to `delete_user.php`, right-click inside the Server Behaviors panel, and select Paste. Bingo, one quick, easy recordset.
5. Give the page a heading and title, and insert a form. Use the Bindings panel to insert some details that will identify the user (this is the same as displaying details in `list_users.php` earlier in the chapter), and add a submit button named `delete` with a suitable label. The screenshot shows a suggested layout:



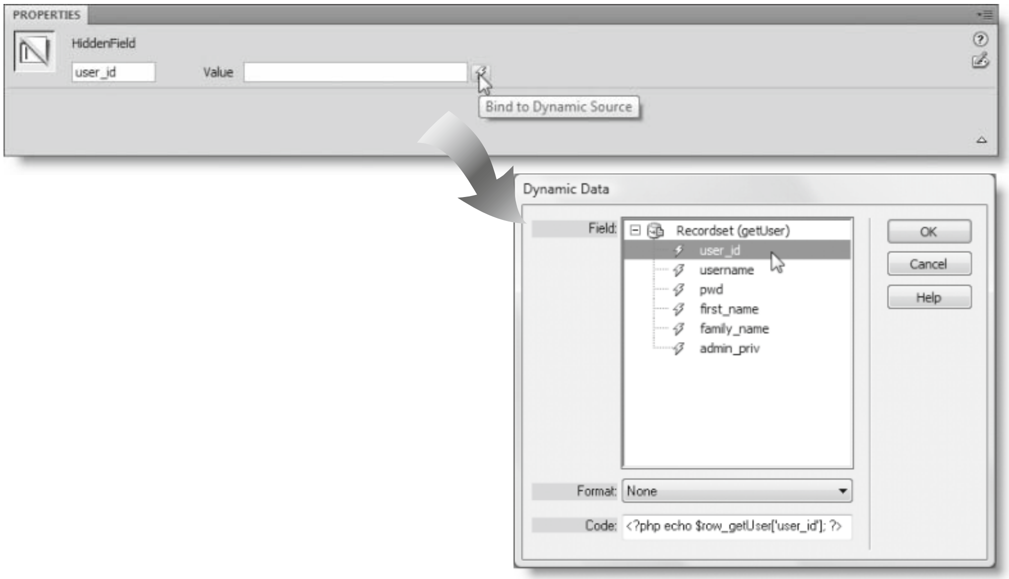
Delete User

You are about to delete the following user record. This cannot be undone.

Name: {getUser.first_name} {getUser.family_name}

Username: {getUser.username}

6. Insert a hidden field into the form. This will be used to pass the primary key to the `DELETE` command, so name the field user_id. You need to get its value from the `getUser` recordset, so click the lightning bolt icon to the right of the Value field to open the Dynamic Data dialog box, and select user_id from the getUser recordset, as shown in Figure 14-21.

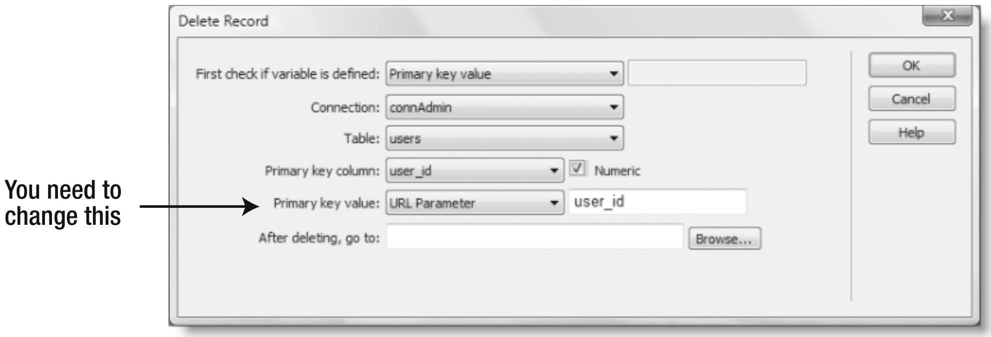


Code: `<?php echo $row_getUser['user_id']; ?>`

Figure 14-21. Bind the value of a form field to a recordset result by clicking the lightning bolt icon alongside the field.

7. You're now ready to apply the Delete Record server behavior. Use the plus button in the Server Behaviors panel, and select Delete Record from the menu that opens. Alternatively, use the Delete Record button on the Data tab of the Insert bar, or the menu option, Insert ➤ Data Objects ➤ Delete Record.

This opens the Delete Record dialog box, as shown in Figure 14-22. Most of the values are selected automatically by Dreamweaver, but you should always check them.



First check if variable is defined:	Primary key value
Connection:	connAdmin
Table:	users
Primary key column:	user_id ✓ Numeric
Primary key value:	URL Parameter user_id
After deleting, go to:	

Figure 14-22. The Delete Record server behavior needs to use the primary key submitted by the form.

Make sure the connection that has administrative privileges and the correct table are selected.

When you select the table from which the record is to be deleted, Dreamweaver should automatically select the correct value for the Primary key column. However, the server behavior uses the hidden field to identify the correct record to delete, so make sure you select Form Variable as the Primary key value and that the primary key's name (`user_id`) is entered in the text field alongside.

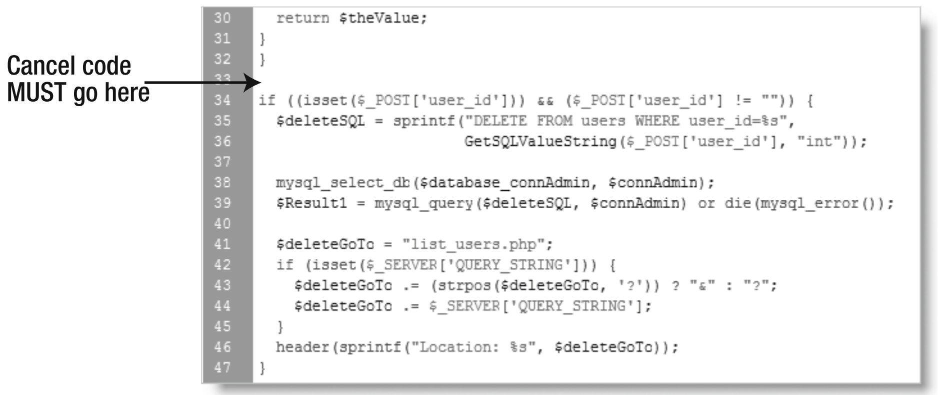
After the record has been deleted, it's a good idea to load the complete list, so enter list_users.php in the final field labeled After deleting, go to. Click OK to insert the server behavior.

8. The delete user page is now fully operational, but what happens if you have selected the wrong record or change your mind about deletion? The easy way is to use the browser back button or a text link to return to the list of registered users. However, it looks more professional to add a cancel button.

You'll notice that the Delete Record server behavior has inserted a hidden field icon alongside the Confirm deletion button in Design view. Position your cursor alongside the hidden field icon, and insert another submit button. In the Property inspector, enter cancel in the Button name field, and set the Value field to Cancel.

9. You need to be very careful where you put the code to cancel the delete operation.

Switch to Code view, and locate the code shown in the following screenshot:



```
30     return $theValue;
31   }
32   }
33
34   if ((isset($_POST['user_id'])) && ($_POST['user_id'] != "")) {
35     $deleteSQL = sprintf("DELETE FROM users WHERE user_id=%s",
36                          GetSQLValueString($_POST['user_id'], "int"));
37
38     mysql_select_db($database_connAdmin, $connAdmin);
39     $Result1 = mysql_query($deleteSQL, $connAdmin) or die(mysql_error());
40
41     $deleteGoTo = "list_users.php";
42     if (isset($_SERVER['QUERY_STRING'])) {
43       $deleteGoTo .= (strpos($deleteGoTo, '?')) ? "&" : "?";
44       $deleteGoTo .= $_SERVER['QUERY_STRING'];
45     }
46     header(sprintf("Location: %s", $deleteGoTo));
47   }
```

The code shown on lines 34–47 is the Delete Record server behavior. The important thing to notice is the conditional statement on line 34. It simply checks whether `$_POST['user_id']` is set and that it's not an empty string. Because the hidden field inserted by the server behavior sets `$_POST['user_id']`, you *must* cancel the delete operation before the script gets to this line.

10. Insert the following code at the point indicated in the preceding screenshot:

```
if (array_key_exists('cancel', $_POST)) {
  header('Location: http://dwcs4/workfiles/ch14/list_users.php');
  exit;
}
```

The first argument to `array_key_exists()` must be the name you give to the cancel button. It's case-sensitive, so make sure you spell it correctly. The code inside the braces uses `header()` to redirect the user back to `list_users.php`. Change the URL to the page you want users to be sent to when cancelling a delete operation. Calling `exit` after `header()` terminates the PHP script, ensuring that the form is not displayed again. Technically speaking, `exit` is not a function, so it doesn't need to be followed by a pair of parentheses. However, `exit;` and `exit();` are both equally correct.

11. Load `list_users.php` into a browser, and click the DELETE link alongside one of the names. Make sure that `delete_user.php` displays the details of the record you selected for deletion. If the details aren't there, check the query string at the end of the URL. Make sure it's correctly formed with the variable and primary key number. Also check the spelling, paying careful attention to uppercase and lowercase.

If the details display correctly, test the Cancel button first. When you're taken back to the list of users, the record should still be listed. If it isn't, check the location of the code inserted in step 10.

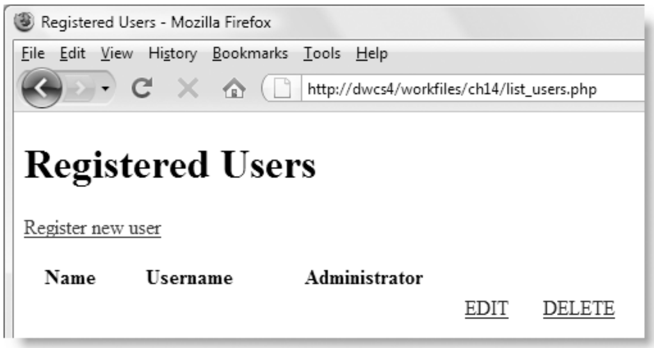
Finally, test the Confirm deletion button. This time, the record should be deleted. Don't worry that you're losing data that has already been saved. Testing is essential. You should always be prepared to sacrifice data while making sure everything works as expected.

That's all there is to it. You can check your code against `delete_user.php` in `examples/ch14`.

You now have a basic but nevertheless fully functional user registration system. In the next chapter, you'll improve it considerably, but to round out this chapter, I want to show you how to control what is displayed onscreen when a recordset produces no results.

Displaying different content when a recordset is empty

Use `list_users.php` to delete all records in the `users` table. When the final record has been deleted, the page looks rather odd, as shown in Figure 14-23.


Registered Users - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dwcs4/workfiles/ch14/list_users.php

Registered Users

Register new user

Name	Username	Administrator		
			EDIT	DELETE

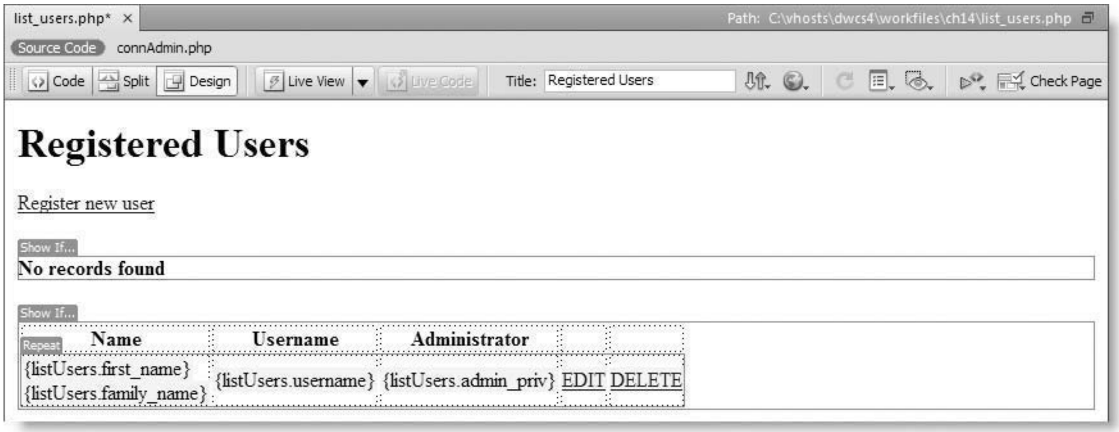
Figure 14-23. The list of users looks untidy when no records are left in the table.

Dreamweaver has a convenient set of server behaviors that can display different content depending on whether a recordset is empty. Let's put them into action in `list_users.php`.

Applying the Show Region server behavior

These instructions show you how to hide the table and display a different message onscreen when the `listUsers` recordset is empty.

1. Open `list_users.php` in the Document window.
2. Insert a new paragraph between the Register new user link and the table that displays the `listUsers` recordset. Type No records found.
3. Select the paragraph you have just created by clicking <p> in the Tag selector at the bottom of the Document window.
4. Click the plus button in the Server Behaviors panel, and select Show Region ➤ Show If Recordset Is Empty from the menu that appears. The same options are available on the Data tab of the Insert bar and the Insert ➤ Show Region menu.
5. This opens a dialog box that asks you to select the recordset. Since there's only one on the page, `listUsers` is chosen automatically, so just click OK to apply the server behavior.
6. Select the table that displays the results of the `listUsers` recordset, and repeat steps 4 and 5. However, this time select Show If Recordset Is Not Empty. The paragraph and table should now be surrounded by Show If tabs, as shown in Figure 14-24.



Registered Users

Register new user

No records found

Name	Username	Administrator		
{listUsers.first_name} {listUsers.family_name}	{listUsers.username}	{listUsers.admin_priv}	EDIT	DELETE

Figure 14-24. Dreamweaver surrounds optional regions with Show If tabs.

7. Save `list_users.php`, and load it into a browser. This time, you should see No records found. The empty table is hidden.
8. Click the Register new user link, and add a new record to the `users` table. When `list_users.php` reloads, you should see the details in the table, and the No records found message has disappeared. A simple but effective solution.

   You can check your code, if necessary, against `list_users_05.php` in `examples/ch14`.

When you mouse over the EDIT and DELETE links in `list_users.php`, you'll see that the number used for the primary key has *not* been reset to 1. MySQL continues assigning new numbers as the primary key. As I wrote earlier in this chapter, *don't even think about renumbering*. The primary key is intended as a unique identifier and should not be reused even when a record is deleted. If you need to find out how many records there are in a database table, it's easy to do with the following SQL:

```
SELECT COUNT(*) AS total FROM tableName
```

I'll show you in Chapter 17 how to build your own SQL queries like this.

Chapter review

This chapter has taken you through all the basic commands in SQL: INSERT, SELECT, UPDATE, and DELETE. With the exception of activating the Cancel button on the delete page, I have deliberately avoided diving into the code that Dreamweaver has created on your behalf. What you have built is only a simple table, but the principle behind creating a more complex table to store much more information is identical. Most of your work with a database involves these four commands.

Dreamweaver has taken virtually all the hard work out of creating this user registration system, but if you're hoping to leave all the coding to Dreamweaver, you'll rapidly discover that you're very limited in what you can do with a database. The Adobe development team says it regards the server behaviors as serving two main purposes: rapid prototype development and as a learning tool. Rapid prototype development lets you build a database-driven site as a proof of concept to demonstrate how the site will work. Once the plan has been approved, it's necessary to add server-side validation to the basic code generated by Dreamweaver. As a learning tool, Dreamweaver takes a lot of the tedium and uncertainty out of connecting with a database and building the basic SQL queries to manage and display database content.

The next chapter begins the learning process by examining the code created by Dreamweaver for the user registration system and then makes it more secure by adding server-side validation. You'll also use the details stored in the users table to control access to different parts of your website.