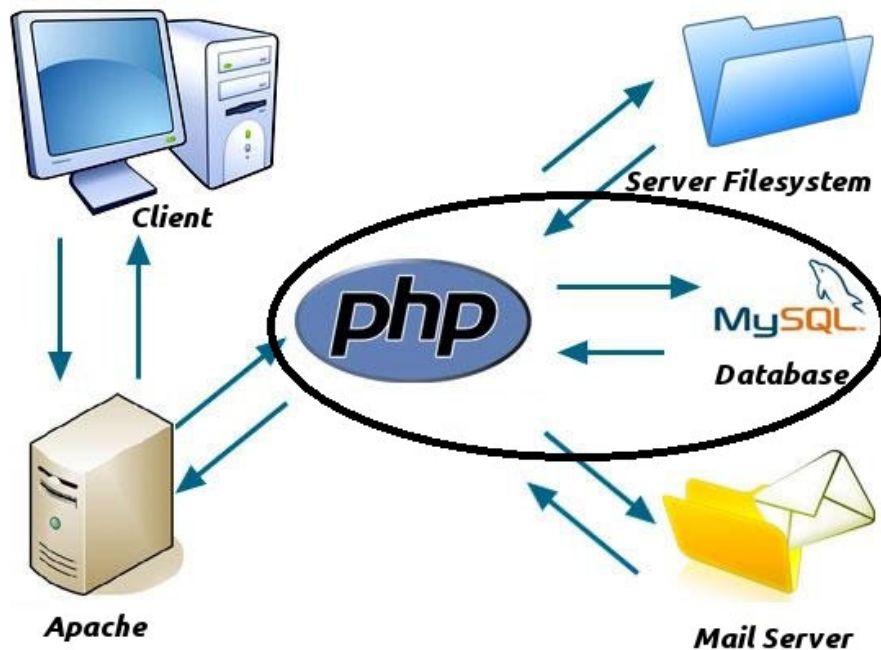




COMP721 Web Development



Week 5: MySQL Database with PHP



How to connect to
DB with PHP?

How to protect your
DB password in
PHP?

Agenda

- *Accessing database with PHP*
- *Selecting DB and Handling errors*
- *Executing SQL Statements: CRUD operations on tables*

ACCESSING DATABASE WITH PHP

Accessing Databases from PHP

- PHP has the ability to access and manipulate any database that is ODBC compliant
- PHP includes functionality that allows you to work *directly* with different types of databases, without going through ODBC
- PHP also supports SQLite, database abstraction layer functions, and PEAR DB
 - PEAR (PHP Extension and Application Repository) is a library of open source PHP code.
 - PEAR DB, one of the most popular PEAR code modules, performs similar functions as ODBC, but designed specifically to work with PHP.

<http://www.php.net/manual/en/refs.database.php>

Accessing Databases from PHP

- There are three main options when considering connecting to a MySQL database server using PHP:
 - ☐ PHP's MySQL Extension
 - ☐ PHP's mysqli Extension
 - ☐ PHP Data Objects (PDO)
- The mysqli extension features a dual interface, supporting both procedural (functions) and object-oriented interfaces.
- These notes and examples use the ***procedural interface***.



<http://www.php.net/manual/en/book.mysqli.php>

Connecting to MySQL

- Open a connection to a MySQL database server with the `mysqli_connect()` function
- The `mysqli_connect()` function returns a *positive integer* if it connects to the database successfully or `false` if it does not
- Assign the return value from the `mysqli_connect()` function to a variable that you can use to access the database in your script

Connecting to MySQL (continued)

- The syntax for the `mysqli_connect()` function is:

```
$connection = mysqli_connect("host" [, "user",  
"password", "database"])
```

- The *host* argument specifies the host name where your MySQL database server is installed e.g. `cms1amp14.aut.ac.nz`
- The *user* and *password* arguments specify a MySQL account name and password e.g. `1234567 yourMySQLpassword`
- The *database* argument specifies a database e.g. `1234567_db`

Discussion:

- What happens if you forgot the `<?php` tag in your PHP file?
- *how to protect your db connection password?*

Obsfucation – a way to protect your code

```
function SGVsbG8gV29ybGQh($_ = 0) {(  
    $__=__FUNCTION__  
)&&  
!$_ and list($_,$__)= array_values(array_filter($__ (42), $__)) and  
!$_($__($__)) and  
$__($__); return  
$_  
&42  
?current(get_defined_functions()):(  
!((  
$_=md5($_))-42*2)or  
!(md5($_ = md5($_))-42/2  
*3)  
);};
```

```
SGVsbG8gV29ybGQh();
```


Connecting to MySQL (continued)

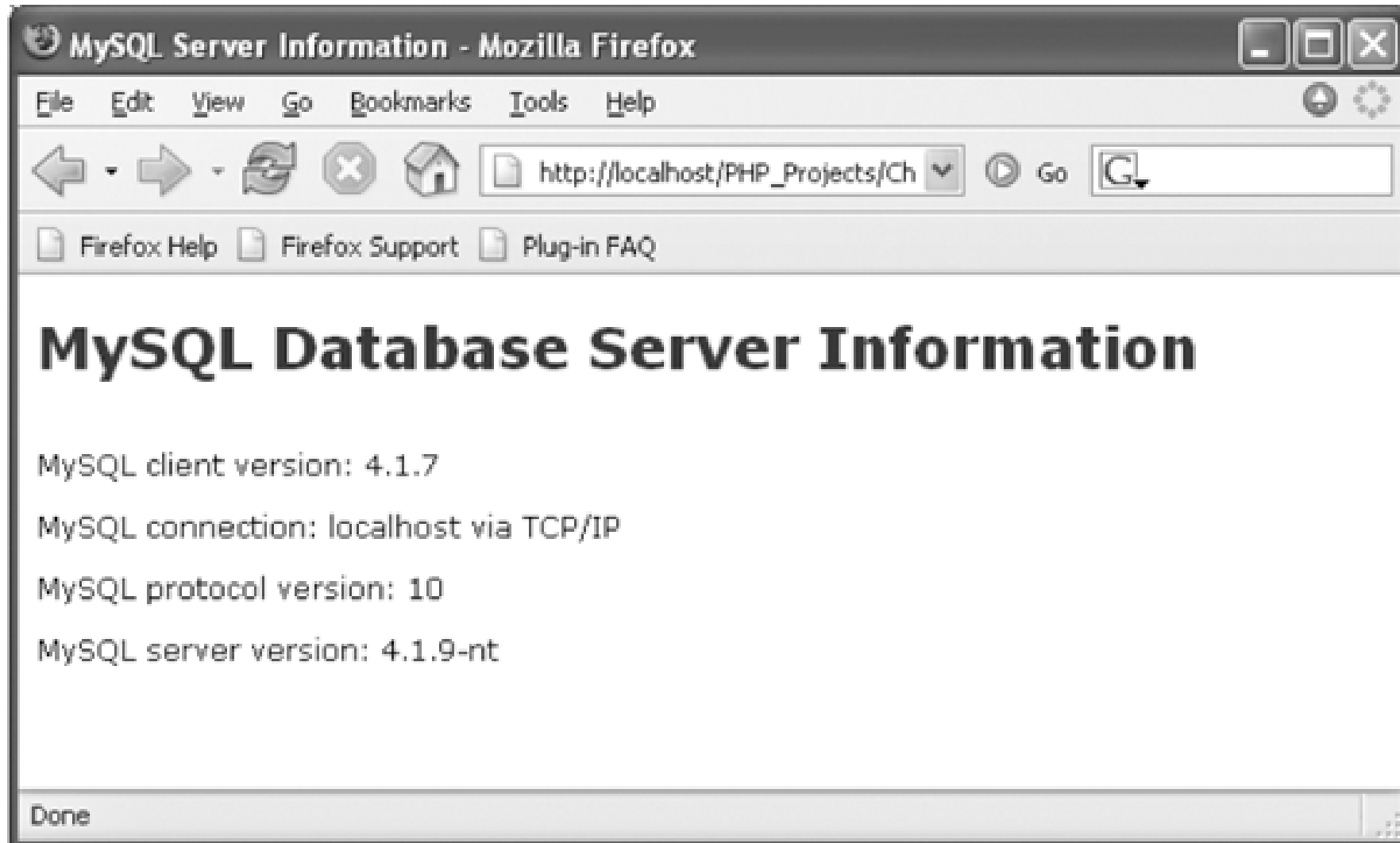
MySQL server information functions

Function	Description
<code>mysqli_get_client_info()</code>	Returns the MySQL client version
<code>mysqli_get_client_version()</code>	Returns the MySQL client version as an integer
<code>mysqli_get_host_info(connection)</code>	Returns the MySQL database server connection information
<code>mysqli_get_proto_info(connection)</code>	Returns the MySQL protocol version
<code>mysqli_get_server_info(connection)</code>	Returns the MySQL database server version
<code>mysqli_get_server_version(connection)</code>	Returns the MySQL database server version as an integer

`mysqli_get_client_info()`

`mysqli_get_client_version()`

Connecting to MySQL (continued)



Web browser output for example script mysqlinfo.php

Selecting a Database

- The statement for selecting a database with the MySQL Monitor, is the `use database` statement

- The function for selecting a database is `mysqli_select_db()`

- The syntax is:

```
mysqli_select_db(connection, database)
```

- The function returns a value of `true` if it successfully selects a database or `false` if it does not

Connecting and Selecting

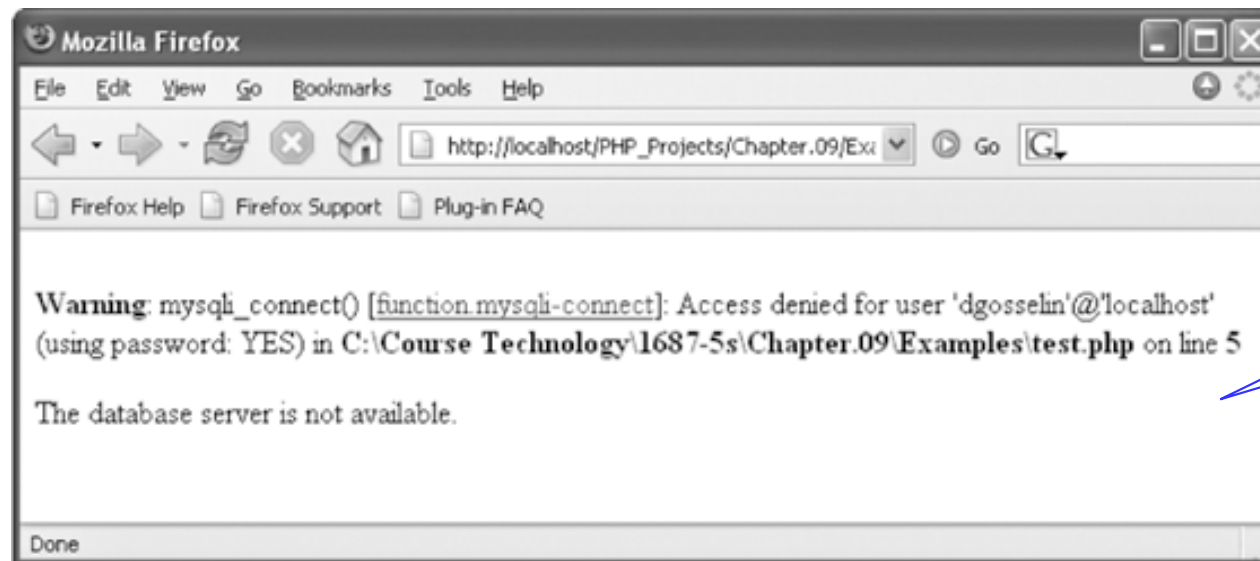
- The `mysqli_connect` also allows one to connect and select the database in one function

```
$connection = mysqli_connect(  
    "cmslamp14.aut.ac.nz", "123456", "  
    "password", "s123456_db")
```

Handling MySQL Errors

- Reasons for not connecting to a database server include:
 - The database server is not running
 - Insufficient privileges to access the data source
 - Invalid username and/or password

e.g. `if (!$dbConnect) ...`



We do not want users to see any database error messages !

Database connection error message

Handling MySQL Errors

Suppressing Errors with the Error Control Operator

■ Writing code that anticipates and handles potential problems is often called **bulletproofing**

■ Bulletproofing techniques include:

- Validating submitted form data

e.g. `if (isset($_GET['height']) ...`

- Using the **error control operator (@)** to suppress error messages

e.g.

```
$dbConnect = @mysqli_connect(...);
if (!$dbConnect) ...
```

Handling MySQL Errors

Terminating Script Execution

- The `die()` and `exit()` functions **terminate** script execution: `die` output a message return 0, while `exit` can return an error code to the parent process
- The `die()` version is usually used when attempting to access a data source
- Both functions accept a single string argument
- Call the `die()` and `exit()` functions as separate statements or by appending either function to an expression with the `or` operator

Note: When script is terminated, an *incomplete* html page is sent to the client. This is useful for error diagnostics, but poor in a production application.

Handling MySQL Errors (continued)

```
$dbConnect = @mysqli_connect("localhost", "root", "paris");  
if (!$dbConnect)  
    die("<p>The database server is not available.</p>");  
echo "<p>Successfully connected to the database server.</p>";  
$dbSelect = @mysqli_select_db($dbConnect, "flightlog");  
if (!$dbSelect)  
    die("<p>The database is not available.</p>");  
echo "<p>Successfully opened the database.</p>";  
// additional statements that access the database  
mysqli_close($dbConnect);
```


Handling MySQL Errors(continued)

```
$dbConnect = @mysqli_connect("localhost", "root", "paris")  
    or die("<p>The database server is not available.</p>");  
// the above is one statement: connected OK or die  
echo "<p>Successfully connected to the database server.</p>";  
@mysqli_select_db($dbConnect, "flightlog")  
    or die("<p>The database is not available.</p>");  
echo "<p>Successfully opened the database.</p>";  
// additional statements that access the database server  
mysqli_close($DBConnect);
```

No if required here

Handling MySQL Errors(continued)

MySQL error reporting functions

Function	Description
<code>mysqli_connect_errno()</code>	Returns the error code from the last database connection attempt or zero if no error occurred
<code>mysqli_connect_error()</code>	Returns the error message from the last database connection attempt or an empty string if no error occurred
<code>mysqli_errno(<i>connection</i>)</code>	Returns the error code from the last attempted MySQL function call or zero if no error occurred
<code>mysqli_error(<i>connection</i>)</code>	Returns the error message from the last attempted MySQL function call or an empty string if no error occurred
<code>mysqli_sqlstate(<i>connection</i>)</code>	Returns a string of five characters representing an error code from the last MySQL operation or 00000 if no error occurred

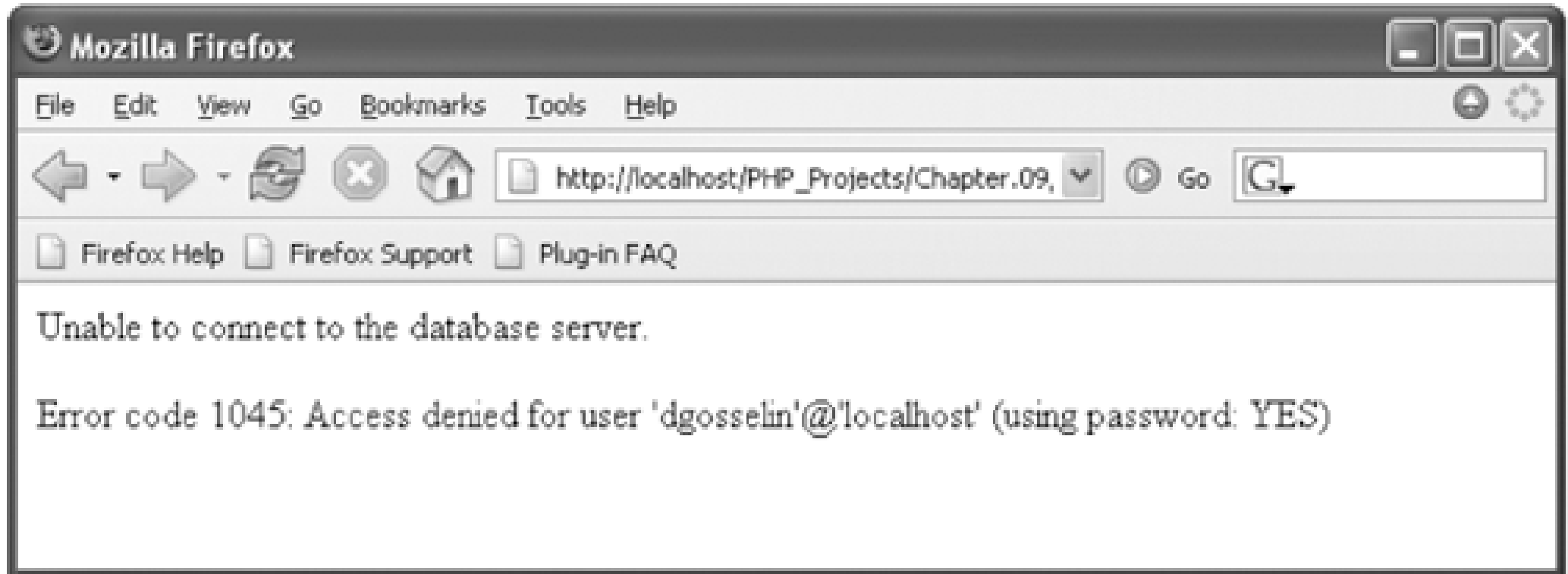
Handling MySQL Errors(continued)

```

$user = $_GET['username'];
$password = $_GET['password'];
$dbConnect = @mysqli_connect("localhost", $user, $password)
    or die("<p>Unable to connect to the database server.</p>"
        . "<p>Error code " . mysqli_connect_errno()
        . ": " . mysqli_connect_error() . "</p>");
echo "<p>Successfully connected to the database server.</p>";
@mysqli_select_db($dbConnect, "flightlog")
    or die("<p>The database is not available.</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);

```

Handling MySQL Errors(continued)



**Error number and message generated by
an invalid username and password**

Handling MySQL Errors(continued)

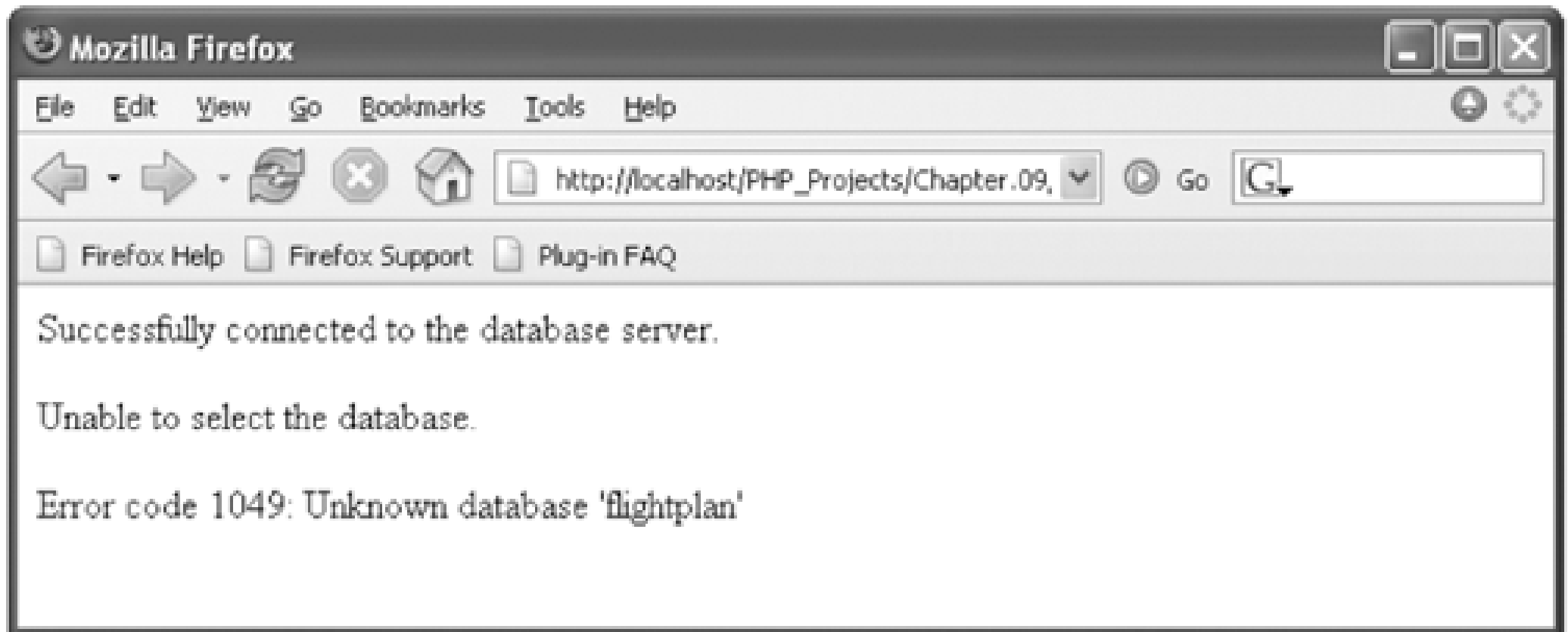
```

$user = $_GET['username'];
$password = $_GET['password'];
$dbConnect = @mysqli_connect("localhost", $user, $password)
    or die("<p>Unable to connect to the database
server.</p>"
    . "<p>Error code " . mysqli_connect_errno()
    . ": " . mysqli_connect_error()) . "</p>";
echo "<p>Successfully connected to the database
server.</p>";

@mysqli_select_db($dbConnect, "flightplan")
    or die("<p>Unable to select the database.</p>"
    . "<p>Error code " . mysqli_errno($dbConnect)
    . ": " . mysqli_error($dbConnect) . "</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);

```

Handling MySQL Errors(continued)



Error code and message generated when attempting to select a database that does not exist

Executing SQL Statements

The `mysqli_query()` function returns one of three values:

- For SQL statements that do not return results
(`CREATE DATABASE` and `CREATE TABLE` statements)
they return a value of `true` if the statement executes successfully
- For SQL statements that return results
(`SELECT` and `SHOW` statements) they return a *result pointer* that represents the query results
 - A **result pointer** is a special type of variable that refers to the currently selected row in a resultset
- For SQL statements that fail,
`mysqli_query()` function returns a value of `false`,
regardless of whether they return results

Cleaning Up

- When you are finished working with query results retrieved with the `mysqli_query()` function, use the `mysqli_free_result()` function to close the resultset
- To close the resultset, pass to the `mysqli_free_result()` function the variable containing the result pointer from the `mysqli_query()` function
 eg `mysqli_free_result($QueryResult);`

Closing Connection

- Close a connection to a MySQL database server with the `mysqli_close()` function
 eg. `mysqli_close($connection);`

Note:

1. Using `mysql_close()` isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.
2. Persistent links to DB can be reused by other scripts, but occupying resources (use `mysql_pconnect()`). Usually unnecessary for MySQL as creating a DB connection is not expensive for MySQL

Accessing database with PHP

■ Step 1 – **Open** a connection:

- ☐ Connect to the Database Server, and select the Database,

■ Step 2 – **Manipulate** the database:

- ☐ Prepare SQL strings
- ☐ Talk to the Database and executes SQL string

■ Step 3 – **Close** the connection:

- ☐ Clean-up, discard the “query” result objects, or other related objects (if any)
- ☐ Close the connection to the Database and the Database Server

Accessing database with PHP

Open	{	// ## 1. open the connection
		// Connect to mysql server
		<code>\$conn</code> = @mysqli_connect('sqlserver','user_name','password')
		or die('Failed to connect to server');
		// Use database
Manipulate	{	@mysqli_select_db(<code>\$conn</code> , 'my_database')
		or die('Database not available');
		// ## 2. set up SQL string and execute
		// get data from user, escape it, trust no-one. :)
		<code>\$pcode</code> = mysqli_escape_string(\$_GET['pcode']);
Close	{	<code>\$query</code> = "SELECT * FROM postcode WHERE pcode=' <code>\$pcode</code> '";
		<code>\$results</code> = mysqli_query(<code>\$conn</code> , <code>\$query</code>);
		// ... Now use data however we want ...
		// ## 3. close the connection
		mysqli_free_result(<code>\$results</code>);
		mysqli_close(<code>\$conn</code>);

CREATING AND DELETING DATABASES AND TABLES

Creating and Deleting Databases

- Use the CREATE DATABASE statement with the `mysqli_query()` function to create a new database

```
$sqlString = "CREATE DATABASE real_estate";
$queryResult = @mysqli_query($dbConnect, $sqlString)
    or die("<p>Unable to execute the query.</p>"
        . "<p>Error code " . mysqli_errno($dbConnect)
        . ": " . mysqli_error($dbConnect)) . "</p>");
echo "<p>Successfully executed the query.</p>";
mysqli_close($dbConnect);
```

Creating and Deleting Databases

(continued)

- Use the `mysqli_db_select()` function to check whether a database exists before you create or delete it
- To use a new database, you must select it by executing the `mysqli_select_db()` function
- Deleting a database is almost identical to creating one, except use the `DROP DATABASE` statement instead of the `CREATE DATABASE` statement with the `mysqli_query()` function

Creating and Deleting Databases

(continued)

```
$dbName = "real_estate";
...
if (@!mysqli_select_db($dbConnect, $dbName))
    echo "<p>The $dbName database does not exist!</p>";
else {
    $sqlString = "DROP DATABASE $dbName";
    $queryResult = @mysqli_query($dbConnect, $sqlString)
        or die("<p>Unable to execute the query.</p>"
        . "<p>Error code " . mysqli_errno($dbConnect)
        . ": " . mysqli_error($dbConnect)) . "</p>";
    echo "<p>Successfully deleted the database.</p>";
}
mysqli_close($dbConnect);
```

Creating and Deleting Tables

- To create a table, use the `CREATE TABLE` statement with the `mysqli_query()` function
- Execute the `mysqli_select_db()` function before executing the `CREATE TABLE` statement or the new table might be created in the wrong database
- To prevent code from attempting to create a table that already exists, use a `mysqli_query()` function that either attempts to `SELECT` records from the table, or attempts to `'SHOW TABLES LIKE'`

Creating and Deleting Tables (continued)

```
$dbName = "real_estate";  
  
...  
  
$sqlString = "CREATE TABLE commercial (  
    city VARCHAR(25), state VARCHAR(25),  
    sale_or_lease VARCHAR(25),  
    type_of_use VARCHAR(40), Price INT, size INT)";  
  
$queryResult = @mysqli_query($dbConnect, $sqlString)  
    or die("<p>Unable to execute the query.</p>"  
    . "<p>Error code " . mysqli_errno($dbConnect)  
    . ": " . mysqli_error($dbConnect) . "</p>");  
  
echo "<p>Successfully created the table.</p>";  
  
mysqli_close($dbConnect);
```

Creating and Deleting Tables (continued)

- To delete a table, use the `DROP TABLE` statement with the `mysqli_query()` function
- To prevent code from attempting to delete a table that does not exist, use a `mysqli_query()` function that either attempts to `SELECT` records from the table, or attempts to `'SHOW TABLES LIKE'`

Creating, updating, selecting and deleting records

Adding, Updating and Deleting Records

Note: Also refer to previous Chapter on SQL

To Add records to a table:

- Use the `INSERT` and `VALUES` keywords with the `mysqli_query()` function
- The values entered in the `VALUES` list must be in the same order that defined in the table fields
- Specify `NULL` in any fields that do not have a value
e.g. for `AUTO_INCREMENT` field

To Add multiple records to a table:

Use the `LOAD DATA` statement and the `mysqli_query()` function with a local text file containing the records to be added.

Adding, Updating and Deleting Records

(continued)

To Update records in a table:

- Use the `UPDATE`, `SET`, and `WHERE` keywords with the `mysqli_query()` function
- The `UPDATE` keyword specifies the name of the table to update
- The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

Using the `mysqli_affected_rows()` Function

- With queries that return results (SELECT queries), use the `mysqli_num_rows()` function to find the number of records returned from the query
- With queries that modify tables but do not return results (INSERT, UPDATE, and DELETE queries), use the `mysqli_affected_rows()` function to determine the number of affected rows by the query

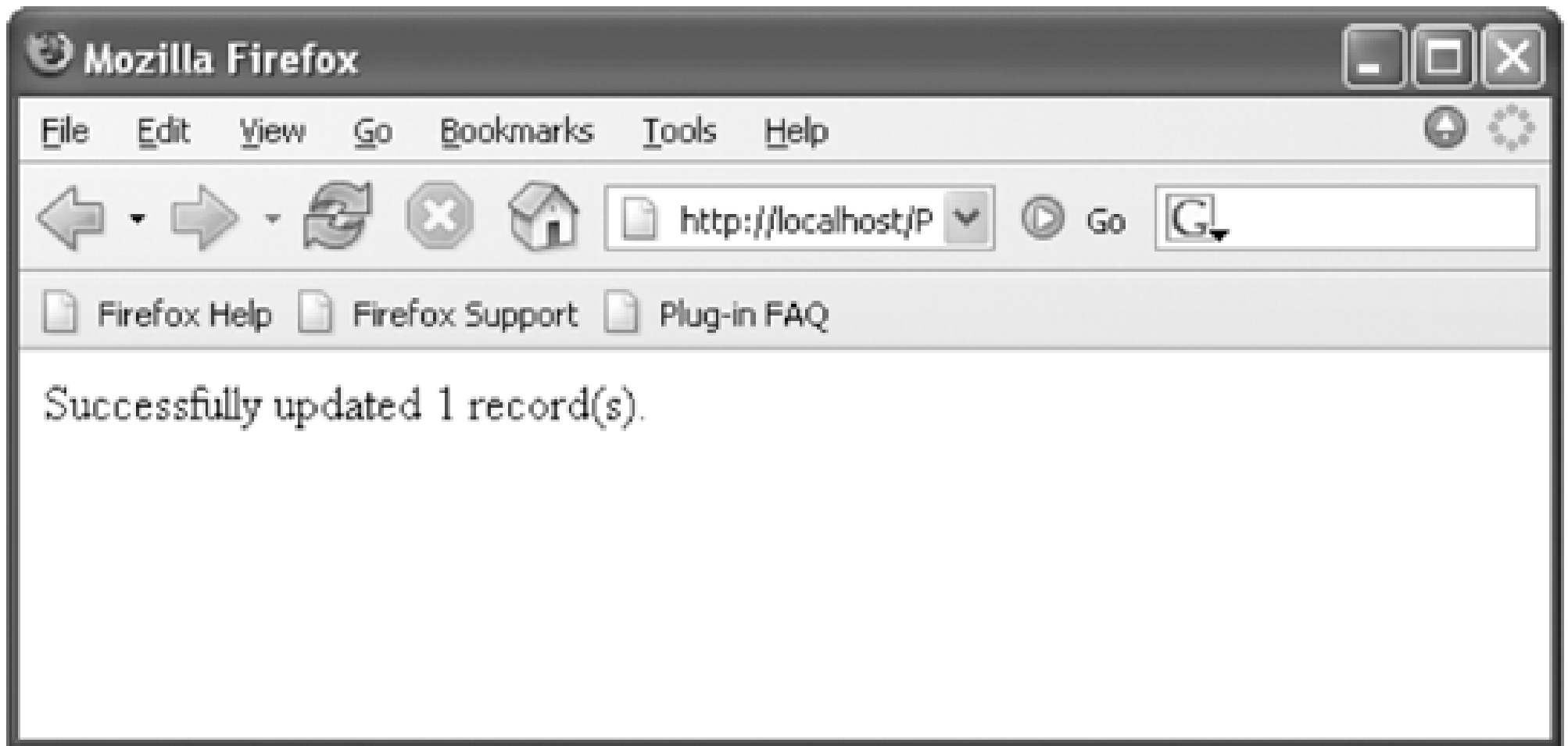
Using the `mysqli_affected_rows()` Function (continued)

```
$sqlString = "UPDATE inventory SET price=368.20
              WHERE make='Fender' AND model='DG7'";

$queryResult = @mysqli_query($dbConnect, $sqlString)
or die("<p>Unable to execute the query.</p>"
      . "<p>Error code " . mysqli_errno($dbConnect)
      . ": " . mysqli_error($dbConnect) . "</p>");

echo "<p>Successfully updated "
     . mysqli_affected_rows($dbConnect) . " record(s).</p>";
```

Using the `mysqli_affected_rows()` Function (continued)



**Output of `mysqli_affected_rows()`
function for an UPDATE query**

Adding, Updating and Deleting Records

(continued)

To Delete records from a table:

- Use the `DELETE` and `WHERE` keywords with the `mysqli_query()` function
- The `WHERE` keyword determines which records to delete in the table
- Be careful, if no `WHERE` keyword, all records are deleted
!!

To select from a table:

- Use the `SELECT` and `WHERE` keywords with the `mysqli_query()` function
- The `WHERE` keyword determines which records to select in the table
- if no `WHERE` keyword, all records are selected

Selecting Records (continued)

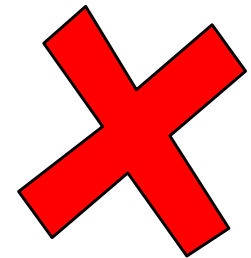
- Use the `mysqli_query()` function to send SQL statements to MySQL
- The syntax for the `mysqli_query()` function is:

```
mysqli_query(connection, query)
```

Be careful when constructing query:

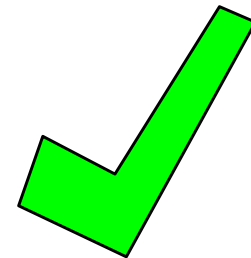
```
$make = "Holden";
```

```
$sqlString = "SELECT model, quantity FROM  
$dbTable WHERE model = $make";
```



Use:

```
$sqlString = "SELECT model, quantity FROM  
$dbTable WHERE model = '$make' ";
```



Selecting Records (continued)

Common PHP functions for accessing database results

Function	Description
<code>mysqli_data_seek(\$Result, position)</code>	Moves the result pointer to a specified row in the resultset
<code>mysqli_fetch_array(\$Result, MYSQLI_ASSOC MYSQLI_NUM MYSQLI_BOTH)</code>	Returns the fields in the current row of a resultset into an indexed array, associative array, or both and moves the result pointer to the next row
<code>mysqli_fetch_assoc(\$Result)</code>	Returns the fields in the current row of a resultset into an associative array and moves the result pointer to the next row
<code>mysqli_fetch_lengths(\$Result)</code>	Returns the field lengths for the current row in a resultset into an indexed array
<code>mysqli_fetch_row(\$Result)</code>	Returns the fields in the current row of a resultset into an indexed array and moves the result pointer to the next row

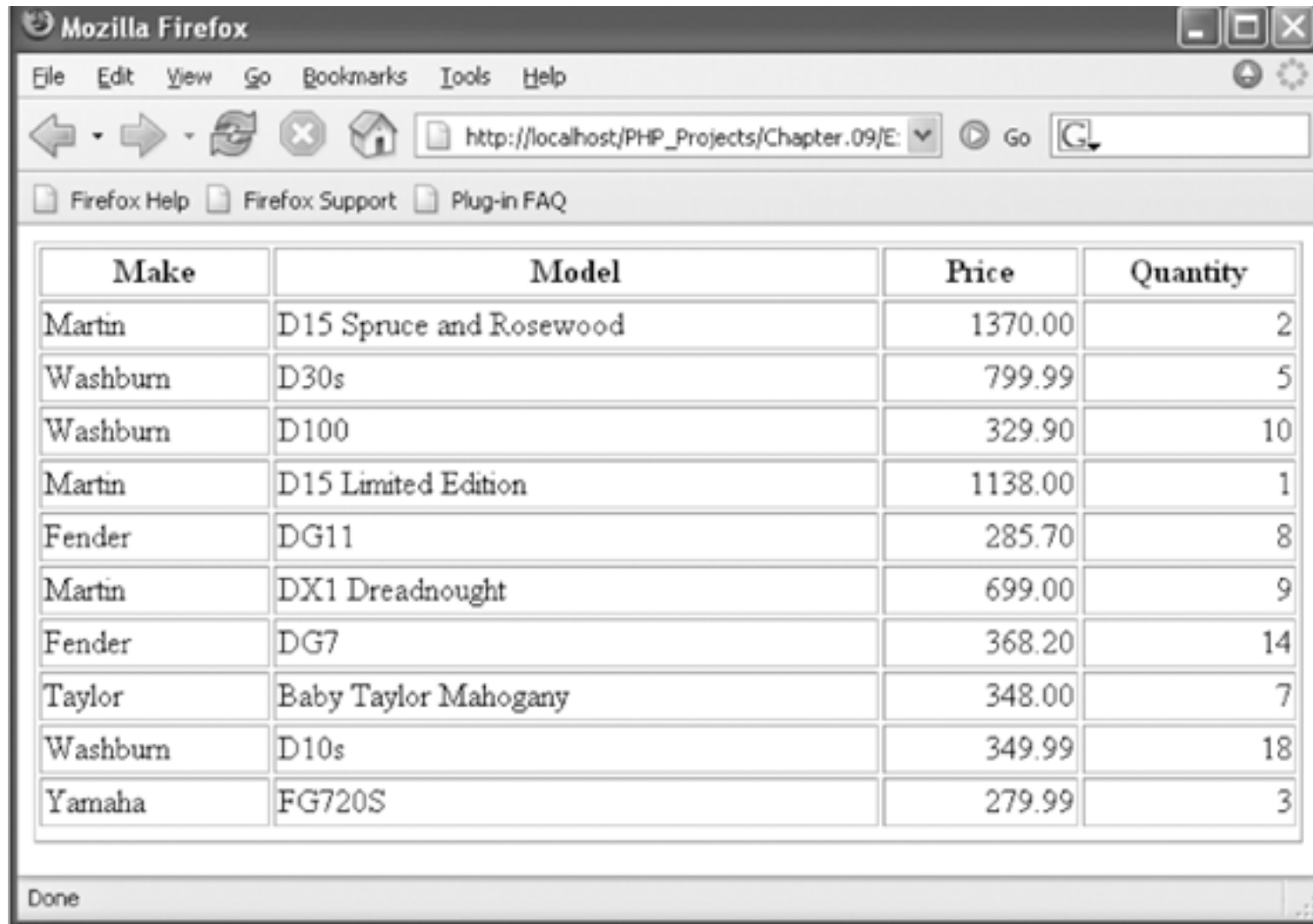
Selecting Records (continued)

Retrieving Records into an Indexed Array

- The `mysqli_fetch_row()` function returns the fields in the current row of a resultset into an indexed array and moves the result pointer to the next row

```
echo "<table width='100%' border='1'>";
echo "<tr><th>Make</th><th>Model</th>
      <th>Price</th><th>Quantity</th></tr>";
$row = mysqli_fetch_row($queryResult);
while ($row) {
    echo "<tr><td>{$row[0]}</td>";
    echo "<td>{$row[1]}</td>";
    echo "<td>{$row[2]}</td>";
    echo "<td>{$row[3]}</td></tr>";
    $row = mysqli_fetch_row($queryResult);
}
echo "</table>";
```

Selecting Records (continued)



The screenshot shows a Mozilla Firefox browser window displaying an inventory table. The browser's address bar shows the URL `http://localhost/PHP_Projects/Chapter.09/E/`. The table has four columns: Make, Model, Price, and Quantity. It lists ten different guitar models with their respective prices and quantities.

Make	Model	Price	Quantity
Martin	D15 Spruce and Rosewood	1370.00	2
Washburn	D30s	799.99	5
Washburn	D100	329.90	10
Martin	D15 Limited Edition	1138.00	1
Fender	DG11	285.70	8
Martin	DX1 Dreadnought	699.00	9
Fender	DG7	368.20	14
Taylor	Baby Taylor Mahogany	348.00	7
Washburn	D10s	349.99	18
Yamaha	FG720S	279.99	3

Output of the inventory table in a Web browser

Selecting Records (continued)

Retrieving Records into an Associative Array

- The `mysqli_fetch_assoc()` function returns the fields in the current row of a resultset into an associative array and moves the result pointer to the next row
- The difference between `mysqli_fetch_assoc()` and `mysqli_fetch_row()` is that instead of returning the fields into an indexed array, `mysqli_fetch_assoc()` function returns the fields into an associate array and uses each field name as the array key

Selecting Records (continued)

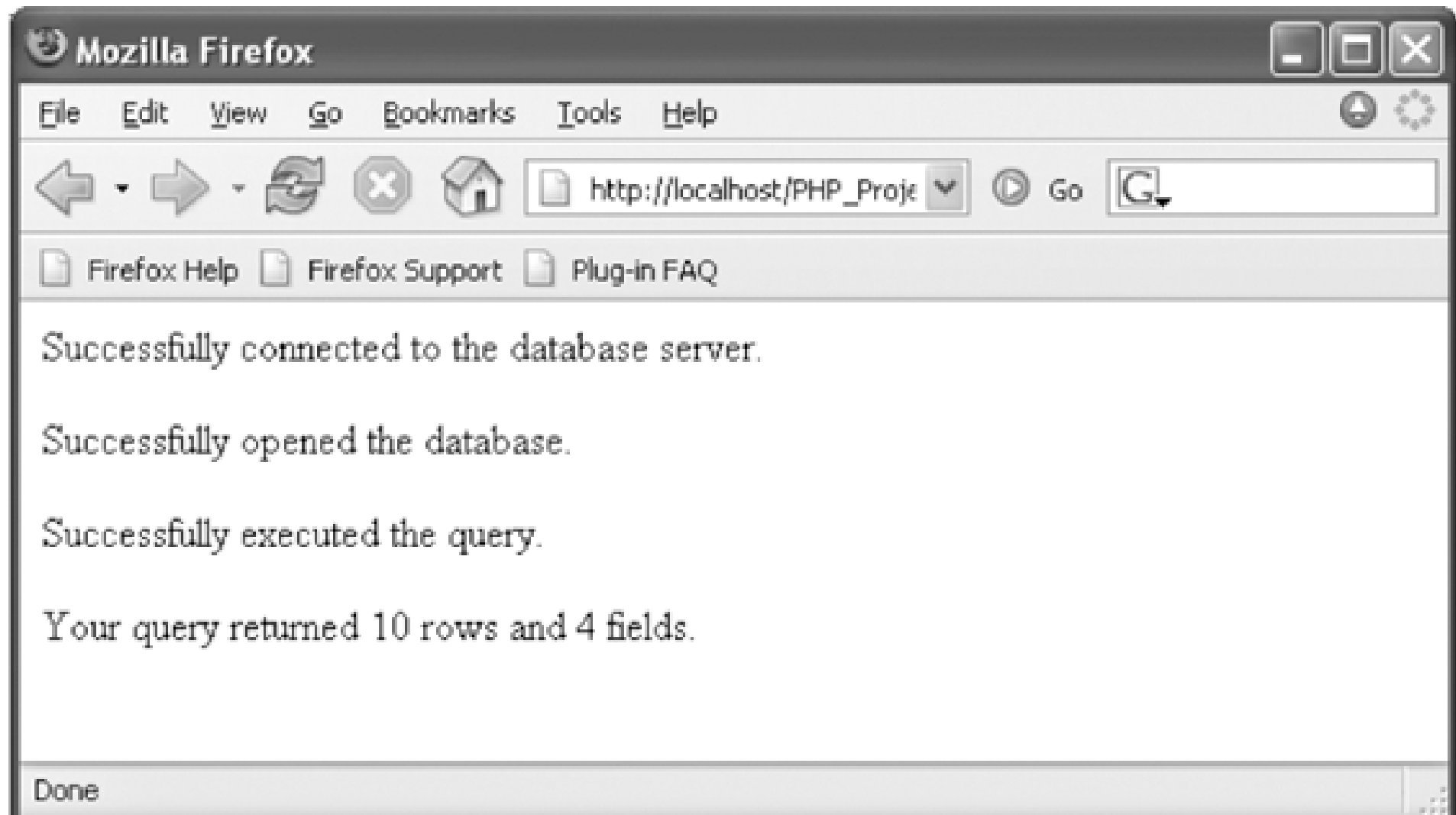
Accessing Query Result Information

- The `mysqli_num_rows()` function returns the number of rows in a query result
- The `mysqli_num_fields()` function returns the number of fields in a query result
- Both functions accept a database result variable, eg. a query result, as an argument

Selecting Records (continued)

```
$sqlString = "SELECT * FROM inventory";
$queryResult = @mysqli_query($dbConnect, $sqlString)
    or die("<p>Unable to execute the query.</p>"
        . "<p>Error code " . mysqli_errno($dbConnect)
        . ": " . mysqli_error($dbConnect) . "</p>");
echo "<p>Successfully executed the query.</p>";
$numRows = mysqli_num_rows($queryResult);
$numFields = mysqli_num_fields($queryResult);
if ($numRows != 0 && $numFields != 0) {
    echo "<p>Your query returned " , $numRows ,
        " rows and " , $numFields , " fields.</p>";
} else {
    echo "<p>Your query returned no results.</p>";
}
mysqli_close($dbConnect);
```

Selecting Records (continued)



**Output of the number of rows and fields
returned from a query**