

A Brain-Friendly Guide

Head First PHP & MySQL



Discover the secrets
behind dynamic,
database-driven sites

Avoid
embarrassing
mishaps with
web forms



Load all the key
syntax directly
into your brain



Hook up your
PHP and
MySQL code



Flex your scripting
knowledge with dozens
of exercises



O'REILLY®

Lynn Beighley & Michael Morrison

Head First PHP & MySQL

by Lynn Beighley and Michael Morrison

Copyright © 2009 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Printing History:

October 2008: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. The *Head First* series designations, *Head First PHP & MySQL*, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and the authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-0-596-00630-3

[M]

This excerpt is protected by copyright law. It is your responsibility to obtain permissions necessary for any proposed use of this material. Please direct your inquiries to permissions@oreilly.com.

5 working with data stored in files

When a database just isn't enough



Don't believe the hype...about databases, that is. Sure, they work wonders for storing all kinds of data involving text, but what about binary data? You know, stuff like JPEG images and PDF documents. Does it really make sense to store all those pictures of your rare guitar pick collection in a database table? Usually not. That kind of data is typically stored in files, and we'll leave it in files. But it's entirely possible to have your virtual cake and eat it too - this chapter reveals that you can use files and databases together to build PHP applications that are awash in binary data.

Virtual guitarists like to compete

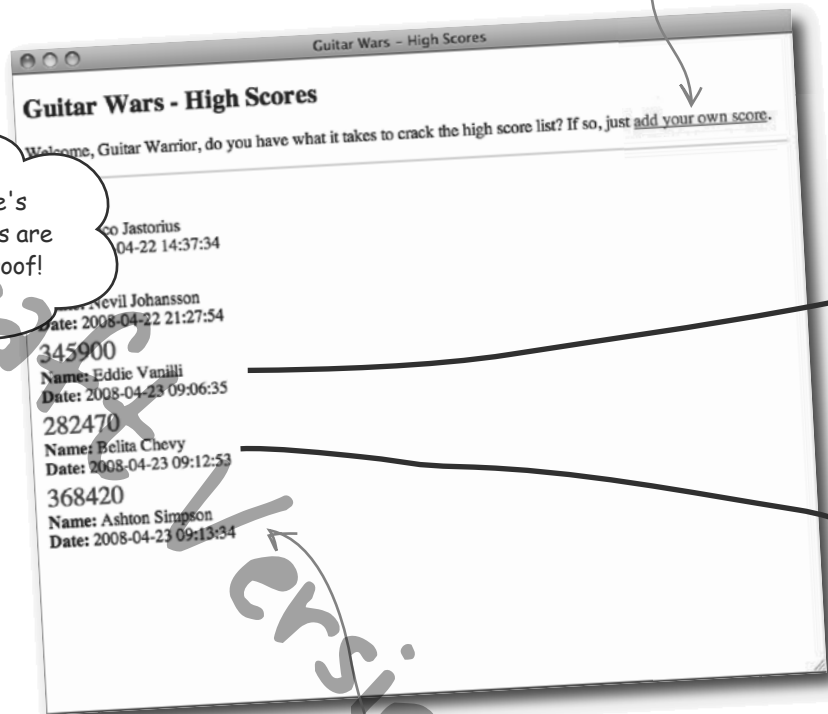
Apparently creating art for art's sake isn't always enough because players of the hot new game Guitar Wars are quite enamored with competitive virtual guitar playing. So much so that they regularly post their high scores at the Guitar Wars web site, which you are now in charge of maintaining. Problem is, there isn't currently a good way to verify the scores.

The Guitar Wars application allows users to add their own scores to the high score list.

Belita, skeptical Guitar Wars rocker.



This is so bogus. There's no way all those scores are real. I'd like to see proof!



Revision

With no way to verify, we can't know whose score is valid and whose isn't.

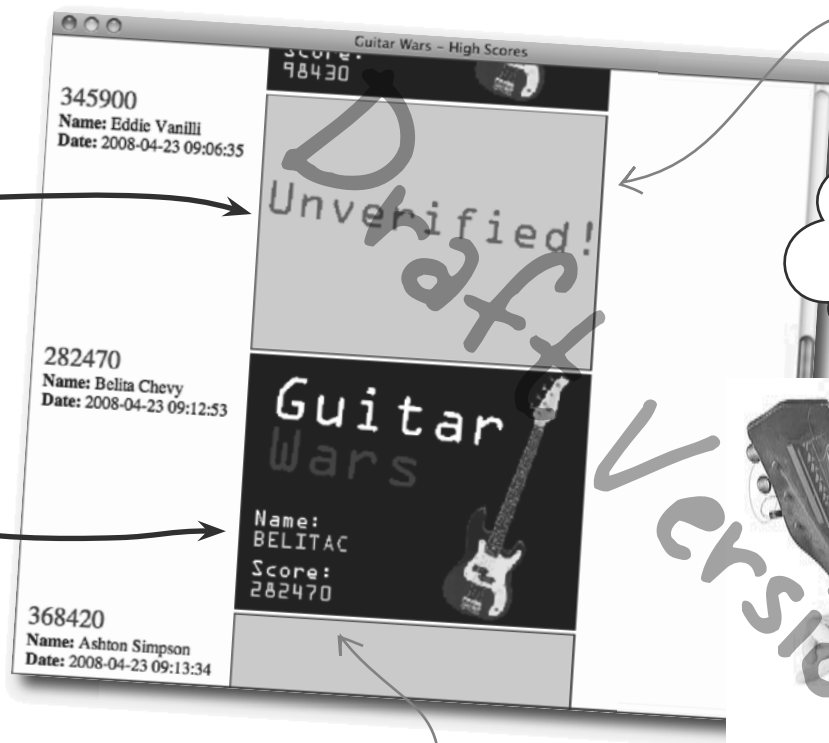
Text can't be trusted

Right now players just post up their high scores purely as text, and there have been lots of disputes over whose scores are valid and whose aren't. There's only one way to put an end to all this bickering and crown a legitimate Guitar Wars champion...

The proof is in the ^{picture} rockin'

Visual verification of a high score is what we need to determine who is for real and who isn't. So the Guitar Wars application needs to allow users to submit a screen shot of their high score when posting their score. This means the high score list will not only be a list of scores, names, and dates, but also a list of images (screen shots).

With photo verification, we find out that Eddie is a Guitar Wars fraud!



So you're saying I'm actually going to have to learn to play this thing? Bummer.

Belita's score is legit thanks to her submitted screen shot.

Eddie, rocker wannabe and Guitar Wars score faker.

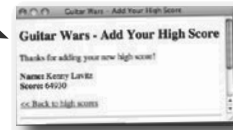
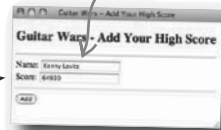


The application needs images

Currently, the Guitar Wars high score application keeps track of three pieces of information: the date and time of a new score, the name of the person submitting the score, and the score itself. This information is entered through a form as part of the application's user interface, after which it gets stored in a MySQL database table called guitarwars.

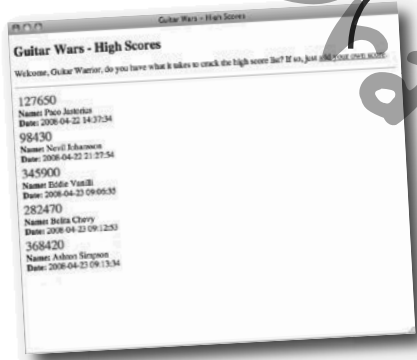
The Add Score page presents a form for entering the name and score (the date/time is automatically entered as the current date/time).

The "add your own score" link on the main Guitar Wars page leads to the Add Score page.

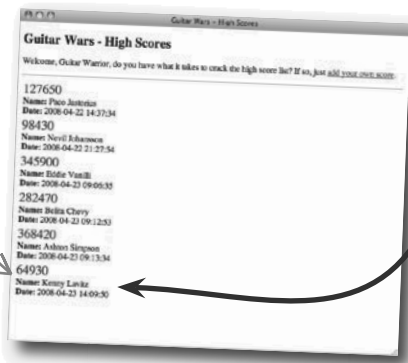


After entering a name and score and clicking Add, the new score is confirmed and added to the guitarwars table in the database.

This ID is the primary key for the database, and is automatically generated for each record.



The new score appears at the bottom of the high score list.



This is the exact date (and time) that a score was submitted to the Guitar Wars application.

id	date	name	score
1	2008-04-22 14:37:34	Paco Jastorius	127650
2	2008-04-22 21:27:54	Nevil Johansson	98430
3	2008-04-23 09:06:35	Eddie Vanilli	345900
4	2008-04-23 09:12:53	Belita Chevy	282470
5	2008-04-23 09:13:34	Ashton Simpson	368420
6	2008-04-23 14:09:50	Kenny Lavitz	64930

The newly added score immediately appears on the main Guitar Wars page.

The guitarwars table also stores the name and score for each high score entry in the database.



The Guitar Wars high score application will have to change to accommodate uploadable image files for high score screen shots. Circle and annotate the parts of the application that must change to support user-submitted images.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Guitar Wars - High Scores</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<h2>Guitar Wars - High Scores</h2>
<p>Welcome, Guitar Warrior, do you have what it takes to crack the
high score list? If so, just <a href="addscore.php">add your own
score</a>.</p>
<hr />

<?php
require_once('connectvars.php');

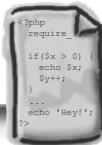
// Connect to the database
$dbc = @mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

// Retrieve the score data from MySQL
$query = "SELECT * FROM guitarwars";
$data = @mysqli_query($dbc, $query);

// Loop through the array of score data, formatting it
echo '<table>';
while($row = mysqli_fetch_array($data)) {
// Display the score data
echo '<tr><td class="scoreinfo">';
echo '<span class="score">' . $row['score'] . '</span>';
echo '<strong>Name:</strong>' . $row['name'] . '<br />';
echo '<strong>Date:</strong>' . $row['date'] . '</td>';
}
echo '</table>';

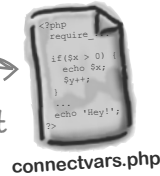
mysqli_close($dbc);
?>

</body>
</html>
```



index.php

These two files don't need to change, so you don't have to worry about them.



connectvars.php



style.css

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Guitar Wars - Add Your High Score</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<h2>Guitar Wars - Add Your High Score</h2>

<?php
require_once('connectvars.php');

if (isset($_POST['submit'])) {
// Grab the score data from the POST
$name = $_POST['name'];
$score = $_POST['score'];

if (!empty($name) && !empty($score)) {
// Connect to the database
$dbc = @mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

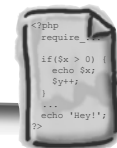
// Write the data to the database
$query = "INSERT INTO guitarwars VALUES (0, NOW(), '$name', '$score')";
mysqli_query($dbc, $query);

// Confirm success with the user
echo '<p>Thanks for adding your new high score!</p>';
echo '<p><strong>Name:</strong>' . $name . '<br />';
echo '<strong>Score:</strong>' . $score . '</p>';
echo '<p><a href="index.php">&lt;&lt; Back to high scores</a></p>';

// Clear the score data to clear the form
$name = "";
$score = "";

mysqli_close($dbc);
}
else {
echo '<p class="error">Please enter all of the information to add
your high score.</p>';
}
?>

<hr />
<form method="post" action="<?php echo $PHP_SELF; ?>">
<label for="name">Name: </label><input type="text" name="name"
value="<?php if(isset($name)) echo $name; ?>" /><br />
<label for="score">Score: </label><input type="text" name="score"
value="<?php if(isset($score)) echo $score; ?>" />
<hr />
<input type="submit" value="Add" name="submit" />
</form>
</body>
</html>
```



addscore.php

guitarwars

id	date	name	score
1	2008-04-22 14:37:34	Paco Jastorius	127650
2	2008-04-22 21:27:54	Nevil Johansson	98430
3	2008-04-23 09:06:35	Eddie Vanilli	345900
4	2008-04-23 09:12:53	Belita Chevy	282470
5	2008-04-23 09:13:34	Ashton Simpson	368420
6	2008-04-23 14:09:50	Kenny Lavitz	64930



Exercise Solution

The Guitar Wars high score application will have to change to accommodate uploadable image files for high score screen shots. Circle and annotate the parts of the application that must change to support user-submitted images.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Guitar Wars - High Scores</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<h2>Guitar Wars - High Scores</h2>
<p>Welcome, Guitar Warrior, do you have what it takes to crack the
high score list? If so, just <a href="addscore.php">add your own
score</a>.</p>
<hr />

<?php
require_once('connectvars.php');

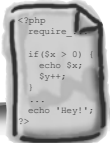
// Connect to the database
$dbc = @mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

// Retrieve the score data from MySQL
$query = "SELECT * FROM guitarwars";
$data = @mysqli_query($dbc, $query);

// Loop through the array of score data, formatting it
echo '<table>';
while($row = mysqli_fetch_array($data)) {
// Display the score data
echo '<tr><td class="scoreinfo">';
echo '<span class="score">' . $row['score'] . '</span>';
echo '<strong>Name:</strong>' . $row['name'] . '<br>';
echo '<strong>Date:</strong>' . $row['date'] . '</td>';
}
echo '</table>';

mysqli_close($dbc);
?>

</body>
</html>
```



index.php

The image should be displayed to the user to confirm success.

guitarwars			
id	date	name	score
1	2008-04-22 14:37:34	Paco Jastorius	127650
2	2008-04-22 21:27:54	Nevil Johansson	98430
3	2008-04-23 09:06:35	Eddie Vanilli	345900
4	2008-04-23 09:12:53	Belita Chevy	282470
5	2008-04-23 09:13:34	Ashton Simpson	368420
6	2008-04-23 14:09:50	Kenny Lavitz	64930

The table needs a new column to store the screen shot image filename for each score.

The form needs an <input> tag for the image file selection.

The screen shot image needs to be displayed on the main page.

The screen shot image file must be obtained from form POST data.

You should validate to make sure the image filename isn't empty.

The MySQL query must now insert the image filename into the guitarwars table.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Guitar Wars - Add Your High Score</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<h2>Guitar Wars - Add Your High Score</h2>

<?php
require_once('connectvars.php');

if (isset($_POST['submit'])) {
// Grab the score data from the POST
$name = $_POST['name'];
$score = $_POST['score'];

if (!empty($name) && !empty($score)) {
// Connect to the database
$dbc = @mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

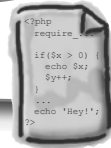
// Write the data to the database
$query = "INSERT INTO guitarwars VALUES (0, NOW(), '$name', '$score')";
mysqli_query($dbc, $query);

// Confirm success with the user
echo '<p>Thanks for adding your new high score!</p>';
echo '<p><strong>Name:</strong>' . $name . '<br />';
echo '<strong>Score:</strong>' . $score . '</p>';
echo '<p><a href="index.php">&lt;&lt; Back to high scores</a></p>';

// Clear the score data to clear the form
$name = "";
$score = "";

mysqli_close($dbc);
}
else {
echo '<p class="error">Please enter all of the information to add your high score.</p>';
}
?>

<hr />
<form method="post" action="<?php echo $PHP_SELF; ?>">
<label for="name">Name: </label><input type="text" name="name" value="<?php if(isset($name)) echo $name; ?>" /><br />
<label for="score">Score: </label><input type="text" name="score" value="<?php if(isset($score)) echo $score; ?>" />
<hr />
<input type="submit" value="Add" name="submit" />
</form>
</body>
</html>
```



addscore.php

Upon success, make sure the sticky image form field gets cleared.

Planning for image file uploads in Guitar Wars

Although it may not seem like a big deal to add support for uploadable screen shot images to Guitar Wars, the application must change in a variety of different ways. For this reason, it's a good idea to have a solid plan of attack before diving into any code. Let's work through the steps required to revamp the Guitar Wars high scores for screen shot images.

1 Use ALTER to add a screenshot column to the table.

First off is the database, which needs a new column for storing the name of each screen shot image file. Since we plan on putting all image files in the same folder, all we need to store in the database is the filename itself (no path).

screenshot

3 Write a query to INSERT the screenshot image filename into the screenshot column of the table.

The Add Score script that processes the form for adding scores must also take into consideration the new file input form field, and handle inserting a screen shot image filename into the screenshot column when inserting a new high score record into the guitarwars table.

screenshot
phizscore.gif

5 Change the main Guitar Wars page to show screen shot images for the high scores.

Last on the laundry list of changes involves the main index.php Guitar Wars page, which must be changed to actually show the screen shot image for each high score that is displayed.

2 Change the Add Score form so that it uses a file input field to allow image file uploads.

The Add Score page already has a form for adding scores, so we need to modify that form and give it a file input field. This input field works in conjunction with the web browser to present the user with a user interface for selecting a file to upload.

Screen shot: Choose File  phizscore.gif

4 Move the uploaded image file from a temporary upload folder to a permanent folder for images.

When a file gets uploaded through a form, it gets stored in a temporary folder somewhere on the web server. It's a good idea to move uploaded files from this temporary location to a permanent folder, such as a special folder you create just for storing application images.



The high score database must be ALTERed

In addition to a variety of PHP scripting tweaks, the image-powered Guitar Wars application needs a new column in the guitarwars table to store screen shot image filenames. Enter MySQL, which offers a statement called ALTER that is capable of modifying tables in all kinds of interesting ways, including adding new columns of data. You used the ALTER statement in the previous chapter to tweak Elmer's email_list table, but let's recap how the command works.

The ALTER statement is used to change the structure of a database.

```
ALTER TABLE guitarwars  
DROP COLUMN score
```

The DROP COLUMN statement drops an entire column from a table.

The ALTER statement is often followed by TABLE to indicate that you're going to alter a table. It's also possible to alter the structure of the entire database with ALTER DATABASE, but that's another story.

OK, maybe that's a dangerous example since it reveals how to drop an entire column from a table, data and all. Yet there certainly may be a situation where you need to remove a column of data from a table. It's more likely, however, that you need to add a column of data, as is the case with Guitar Wars. This is made possible by ADD COLUMN, which is one of several table alterations you can carry out with ALTER.

ADD COLUMN

Adds a new column to a table - just specify the name of the column and its type following ADD COLUMN.

```
ALTER TABLE guitarwars  
ADD COLUMN age TINYINT
```

CHANGE COLUMN

Changes the name and data type of a column - just specify the name of the old column, new column, and new data type following CHANGE COLUMN.

```
ALTER TABLE guitarwars  
CHANGE COLUMN score high_score INT
```

DROP COLUMN

Drops a column (and any data stored in it) from a table - just specify the name of the column following DROP COLUMN.

```
ALTER TABLE guitarwars  
DROP COLUMN age
```

MODIFY COLUMN

Changes the data type or position of a column within a table - just specify the name of the column and its new data type following MODIFY COLUMN. To change the position of a column, specify the name of the column and its exact position (FIRST, SECOND, etc.), or a relative position (BEFORE, AFTER) and another column name.

```
ALTER TABLE guitarwars  
MODIFY COLUMN date AFTER name
```

CHANGE COLUMN is used to change the name and data type of a column, while MODIFY COLUMN changes the data type or position of a column within a table.

Really, the high score database must be ALTERed!

The ALTER statement is powerful and all but we really do need to add a column to the guitarwars table to hold screen shot image filenames. So let's take a closer look at ADD COLUMN specifically.

**ALTER TABLE guitarwars
ADD COLUMN age TINYINT**

ADD COLUMN indicates that we want to alter the table by adding a new column of data.

The name of the table to be altered follows ALTER TABLE.

The name and data type of the new column is specified last in the MySQL query - a TINYINT can hold an integer between -128 and 128.

guitarwars

id	date	name	score	age
1	2008-04-22 14:37:34	Paco Jastorius	127650	
2	2008-04-22 21:27:54	Nevil Johansson	98430	
3	2008-04-23 09:06:35	Eddie Vanilli	345900	
4	2008-04-23 09:12:53	Belita Chevy	282470	
5	2008-04-23 09:13:34	Ashton Simpson	368420	
6	2008-04-23 14:09:50	Kenny Laviitz	64930	

The ALTER statement doesn't affect any of the other table data.

After adding a new column with ALTER TABLE and ADD COLUMN, you end up with an empty column ready to hold data the next time you perform an INSERT.

The new column of data starts out empty for pre-existing records.



Sharpen your pencil

Write a MySQL statement that uses the ALTER command to add a new column named screenshot to the guitarwars table. Make sure to give the new column a MySQL data type.

.....

.....



Sharpen your pencil Solution

Write a MySQL statement that uses the ALTER command to add a new column named screenshot to the guitarwars table. Make sure to give the new column a MySQL data type.

After running this MySQL statement on the guitarwars table, the new column is added to the table.

```
ALTER TABLE guitarwars
```

```
ADD COLUMN screenshot varchar(32);
```

guitarwars

id	date	name	score	screenshot
1	2008-04-22 14:37:34	Paco Jastorius	127650	
2	2008-04-22 21:27:54	Nevil Johansson	98430	
3	2008-04-23 09:06:35	Eddie Vanilli	345900	
4	2008-04-23 09:12:53	Belifa Chevy	282470	
5	2008-04-23 09:13:34	Ashton Simpson	368420	
6	2008-04-23 14:09:50	Kenny Lavitz	64930	

32 characters are enough to accommodate most image filenames, although you can make the column even longer if you want to be extra safe.

Since the column is new, it starts out empty for existing records in the table.

1 Use ALTER to add a screenshot column to the table.

there are no Dumb Questions

Q: Do new columns added with ALTER have to be added to the end of a database table?

A: No, they can be added anywhere. But keep in mind that the order of the columns in a table isn't terribly important. In other words, you can structure query results so that data is organized in any order you want. But maybe you like the sense of structural order brought about by a specific ordering of columns, in which case you may want to add a column in an exact location. You can do this by tacking on the keyword FIRST or LAST to the ALTER query. LAST is the default, so you don't really need to use it.

For even more control, use BEFORE column or AFTER column, like this:

```
ALTER TABLE guitarwars  
ADD COLUMN age TINYINT AFTER name
```

Q: What happens to the existing high score database records after adding the new screenshot column?

A: Since the ALTER statement purely affects the structure of a database, the new screenshot column is empty for all pre-existing high score records. While it's possible to populate the screenshot field of future records, pre-existing records all have an empty screenshot field.

Q: Can screen shot filenames still be added to the pre-existing records?

A: Yes, they definitely can, and you would use the UPDATE MySQL statement to do so. There is nothing stopping you from manually uploading image files to the web server and then using UPDATE to fill in the screen shot filenames for existing scores. But remember that the whole idea here is user-submitted image files, so it makes sense to allow users to upload their own screen shot images. And they can do exactly this by using the improved image-powered Add Score script you're about to build...