UNIVERSITY OF MICHIGAN SCHOOL OF INFORMATION
Database Application Design

Assignment 6 – Building a CRUD Application with Redirect

Due Date: Monday October 13

Overview

In this assignment you will create a MySQL table, MySQL user (i.e using GRANT), and create a PHP application that performs a create, read, update and delete operation based on the sample code with redirects provided for Chapter 10 (i.e. pdo.zip). You can play with an implantation of the application here:

http://www.php-intro.com/assn/tracks/index.php

Warning – this is a public server – so be careful what you put up here.

Tasks

Your data model is about music tracks in a table named "tracks". You can put this table in any database (i.e. misc is OK). You have an auto increment field named "id", a title, play count, and rating information. The play count and rating should be integer values.

```
CREATE TABLE tracks (
  id INT UNSIGNED NOT NULL
     AUTO_INCREMENT KEY,
  title VARCHAR(128),
  plays MEDIUMINT,
  rating MEDIUMINT
);
```

You should build the application using the same file names as in the Chapter 10 sample 'users' code. You should name your form fields in your add and edit scripts title, plays, and rating to facilitate the auto grading.

You should handle the case where the user enters an empty title (string length<1) or non-numeric data or a value <= 0 into the plays or rating field during either the add or edit script. You should provide an error and redirect to **index.php** with an error message if there is invalid data in those fields. When producing the form for add and edit scripts use input fields with type="text" so that the auto grader can test your error handling (i.e. do not use the HTML5 type="number" fields for this assignment).

I recommend that you set the PDO error processing to throw an exception for any error in SQL. For example you could do the following in your **pdo.php** until you get it debugged:

```php
<?php
try
{
    $pdo = new PDO('mysql:host=sql192.byethost18.com;port=3306;dbname=b18_1331_misc',
        'b18_1331', 'blablah');
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}  catch(Exception $ex)  {
    die($ex->getMessage());
}
// Comment this line out when the database is working
echo("Success:\n"); var_dump($pdo);
?>
```

When you successfully complete the INSERT, DELETE, or UPDATE operation, instead of displaying a page with a message and a link to the main page, use the PHP session to store a message and then redirect to the **index.php** where the message is to be displayed.

Important: You must properly use **htmlentities()** *anytime* you are outputting user-entered data into HTML and you must use PDO prepared statements for all SQL that is handing user-entered data. The auto grader will try to exploit your application using both HTML and SQL injection. Points will be taken off if your application has "security holes" because of the lack of proper use of these two functions.

Note that the sample code from Chapter 10 *does* have some code that will fail this test so it is important to check all of the code for possible HTML injection errors. The auto grader will test HTML injection. The code running on **php-intro** no longer has any HTML injection vulnerabilities (or at least I hope so). To test your program, enter a title with some HTML markup. If your program is functioning properly you should see the < and >. If your program has an HTML injection problem you will see bold text as shown in in these screen shots:

Turn In

This assignment will be graded using the auto grader on CTools under "Lesson Builder"