HERIOT WATT UNIVERSITY

F27WD: Web Design & Databases

Databases Lecture 7: PHP and SQL

Fiona McNeill

5th March 2019

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

First, a bit more SQL …

- We will look at the DISTINCT command, which you may need in your assignment.
- This returns only *distinct* answers - i.e., it filters out duplicates.

DISTINCT example

In our running example,

SELECT actorName FROM ActorSkill;

will return:

actorName
Carl Pratt
Jemma Laurence
Anna Stone
Carl Pratt
Jemma Laurence
Rosie Ridley
Anna Stone
Jemma Laurence
Anna Stone
Jemma Laurence
Rosie Ridley

DISTINCT example

Whereas

SELECT **DISTINCT** actorName FROM ActorSkill;

Will return:

actorName
Anna Stone
Carl Pratt
Jemma Laurence
Rosie Ridley

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Start by declaring a table:

```
<table>

</table>
```

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Then add each row between <tr> tags

```
<table>
<tr>
This is row one
</tr>
<tr>
This is row two
</tr>
</table>
```

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Data in rows is added in **<td>** tags

```
<table>
<tr>
<td>This is row one</td>
</tr>
<tr>
<td>This is row two</td>
</tr>
</table>
```

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Add as much data as you like in each row

```
<table>
<tr>
<td>apples</td> <td>carrots</td> <td>mushrooms</td>
</tr>
<tr>
<td>pears</td> <td>peas</td>
</tr>
</table>
```

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Add a header row for a table using <th>

```
<table>
<tr><th>fruit</th><th>veg</th><th>other stuff</th></tr>
<tr>
<td>apples</td> <td>carrots</td> <td>mushrooms</td>
</tr>
<tr>
<td>pears</td> <td>peas</td>
</tr>
</table>
```

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Contents of rows automatically *left align*, except for headers, which *centre align.*

But you can alter this by using *align = 'left', align = 'center'* or *align = 'right'* inside the opening <th> or <td> for the object you want to change.

HERIOT WATT UNIVERSITY

HTML tables - a crash course

Add a header row for a table using <th>

```
<table>
<tr><th align=‘left’>fruit</th><th align=‘left’>veg</th><th align=‘left’>other stuff</th></tr>
<tr>
<td>apples</td> <td>carrots</td> <td>mushrooms</td>
</tr>
<tr>
<td>pears</td> <td>peas</td>
</tr>
</table>
```

HERIOT WATT UNIVERSITY

Database access using PHP and PDO

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

PHP and SQL

- PHP5 and later can connect to a MySQL database using
  - MySQLi (MySQL improved), or
  - PDO (PHP Data Objects)
- For MySQL, it doesn’t matter which you use.
- MySQLi only works with SQL, but PDO works with many different database systems.
- We’re just going to look at using PDO.

HERIOT WATT UNIVERSITY

PHP and SQL

- We will be working through an example file that forms part of your assignment.
- This will cover the HTML and PHP you need to know for this course.
- In your assignment, you won’t be expected to recreate this from scratch - you will be reusing and building on what we look at in this lecture.

HERIOT WATT UNIVERSITY

PHP and SQL

- You should refer to the files *pdo.php* and *sites.php* as we go through the lecture, and when you are looking back at it to help you with the assignment.
- I will put parts of the code on the slide, but not all at once as there is too much.

Connecting to the database

- The file *pdo* connects to the database.

First, within the <php> tags, you need to give it the credentials and location to connect:

```
$host = "mysql-server-1";
$user = "username";
$pass = "password";
$db = "username";
```

Obviously, you need to put your own password and username here.

HERIOT WATT UNIVERSITY

Connecting to the database

- The file *pdo* connects to the database.

Next the file set a variable *$dsn,* which is a concatenation of the host server and the database so that it is in the right format:

```
$dsn = "mysql:host=$host;dbname=$db";
```

HERIOT WATT UNIVERSITY

Connecting to the database

- The file *pdo* connects to the database.

Next comes pdo *options* that we are not going to worry about here.

HERIOT WATT UNIVERSITY

Connecting to the database

- The file *pdo* connects to the database.

Finally, the file tries to set a variable called *%pdo* to a pdo connection using the variables it has set earlier, returning an error message if that doesn't work:

```
try {
  $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
  echo $e->getMessage();
}
```

HERIOT WATT UNIVERSITY

The whole connecting file

```php
<?php

$host = "mysql-server-1";
$user = "username";
$pass = "password";
$db = "username";

$dsn = "mysql:host=$host;dbname=$db";
$options = [
  PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
  PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
  PDO::ATTR_EMULATE_PREPARES => false,
];

try {
  $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
  echo $e->getMessage();
}

?>
```

Connecting

- You can include this code in every file in which you want to connect to a database. But it's easier to create a separate file and then *include* it in any file which requires a database connection.

```
include "pdo.php";
```

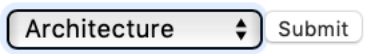
HERIOT WATT UNIVERSITY

Create a drop-down menu page

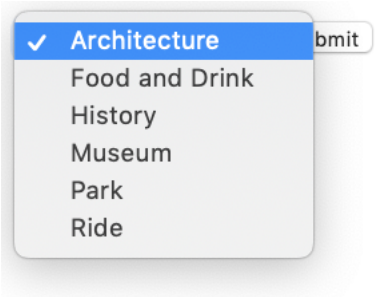
- Today, we are going to work through an example that creates a drop-down menu that is populated with data from a database.
- Users can select something from the drop-down menu.
- Their selection will cause relevant data to be returned.
- The code for this can be found in sites.php on Vision.

Create a drop-down menu page

- In this example, users can choose a particular type of site to visit:



What the user sees initially



What the user sees after clicking on the drop-down menu

Create a drop-down menu page

- In this example, users can choose a particular type of site to visit:

Architecture Submit

Site Name	City
Eiffel Tower	Paris
Golden Gate Bridge	San Francisco
Saint-Louis Cathedral	Versailles
St Giles Cathedral	Edinburgh
St Patricks Cathedral	Dublin
St Pauls Cathedral	London
Statue of Liberty	New York
Versailles Palace	Versailles

What the user sees after hitting ‘submit’

HERIOT WATT UNIVERSITY

Starting and connecting

First of all, you need to open php and connect to the database.

```
<?php
include “pdo.php";
?>
```

Remember that pdo.php is the file we have just been through.

HERIOT WATT UNIVERSITY

Checking for user input

Next, we are going to check whether a variable labelled site_type has been posted to the sites.php. If so, we will assign its value to a variable named selected_site_type.

```
$selected_site_type = $_POST["site_type"];
```

This will be set to whatever the user has selected, or to null if the user hasn't selected anything.

HERIOT WATT UNIVERSITY

Creating an HTML document

Everything else that we do will be contained within HTML code. We need to get PHP to write this code using the `echo` command:

```
echo "<html><body>";
```

```
echo "</body></html>";
```

Everything else that we do in sites.php will go between these lines.

Creating a drop-down menu

This is done via an HTML form. We need to set:

- the `action` - i.e., what file tells us how to process this data. *Note that in this case, this is the same file as the one we are defining the form in.*
- The method - post or get. In this case, we use post.

```
echo "<form action='sites.php' method='post'>";
```

```
echo “</form>";
```

Creating a drop-down menu

Within this, we need to add the code that creates the drop-down menu, and then the button that the user can press once they have selected.

We use `select` to create a drop-down list. Whatever is selected will be assigned a variable name `site_type` and this will be what is returned to sites.php.

```
echo "<form action='sites.php' method='post'>";
echo "<select name=‘site_type’>";

echo "</select>";

echo “</form>";
```

HERIOT WATT UNIVERSITY

Populating a drop-down menu

To populate this menu, we need to send a query to the database to find out what the options should be - what kinds of site types are there?

To do this, we need a query:

SELECT DISTINCT type FROM site ORDER BY type

Why ORDER BY type? This will return the list in alphabetical order, which looks nicer.

HERIOT WATT UNIVERSITY

Populating a drop-down menu

We need to send this query to the `pdo` variable, which is set in the pdo.php file, and we need to assign the result of this query to a variable, which we will name `$result`.

```
$result = $pdo->query("SELECT DISTINCT type FROM Site ORDER BY type");
```

Populating a drop-down menu

In our database, we have six type of sites: Ride, Architecture, History, Food & Drink, Museum, Park.

```
$result = $pdo->query("SELECT DISTINCT type FROM Site ORDER BY type");
```

Calling this query will give us:

HERIOT WATT UNIVERSITY

Populating a drop-down menu

In our database, we have six type of sites: Ride, Architecture, History, Food & Drink, Museum, Park.

```
$result = $pdo->query("SELECT DISTINCT type FROM Site ORDER BY type");
```

Calling this query will give us:

$result = [{ "type": "Architecture" }, { "type": "Food & Drink" }, { "type": "History" }, { "type": "Museum" }, { "type": "Park" }, { "type": "Ride" }];

HERIOT WATT UNIVERSITY

Populating a drop-down menu

Now we have extracted the data from the database, we need to display it. Every element of the list `$return` will be an option.

For this, we use `<option></option>`

If the user selects an option, we want that option to remain selected when the results are returned. For this, we use `<option selected></option>`

Populating a drop-down menu

So we work through the $results list until it is empty, putting each element into <option> brackets except the selected one, for which we add a selected.

```
while ($row = $result->fetch()) {
  $type = $row["type"];
  if ($type == $selected_site_type) {
    $option = "<option selected>";
  } else {
    $option = "<option>";
  }
  echo $option . $type . "</option>";
}
```

HERIOT WATT UNIVERSITY

Full code for creating the menu

```
echo "<html><body>";
echo "<form action='sites.php' method='post'>";
echo "<select name='site_type'>";

$result = $pdo->query("SELECT DISTINCT type FROM Site ORDER BY type");

while ($row = $result->fetch()) {
  $type = $row["type"];
  if ($type == $selected_site_type) {
    $option = "<option selected>";
  } else {
    $option = "<option>";
  }
  echo $option . $type . "</option>";
}

echo "</select>";
echo "<input type='submit' value='Submit'>";
echo "</form>";
```

HERIOT WATT UNIVERSITY

Finding and returning the response

Now we have created the options for the user to choose from and allowed them to select what they want, we need to return the relevant data.

In this case, we need to find the name and cities of sites of that type and return them appropriately.

SELECT name, city FROM site WHERE type=?

HERIOT WATT UNIVERSITY

Prepared queries

To do this, we will create a prepared query. This allows us to set up queries that have values that we don't yet know. We can instantiate the ? on the previous slide during run-time, but at the moment we don't know how to, as we don't know what the user will choose.

HERIOT WATT UNIVERSITY

Prepared queries

```
$stmt = $pdo->prepare("SELECT name, city FROM Site WHERE type = ?");
```

Here, we are creating a variable named `$stmt` and assigning it to whatever is returned by the variable `$pdo` when the given query is run.

But we can only execute this when we know what ? should be.

HERIOT WATT UNIVERSITY

Prepared queries

```
$stmt->execute([$selected_site_type]);
```

To execute a prepared query, simply set the variable that the prepared query is assigned to (`$stmt`) to execute, with a list of the values that any ? should be instantiated to. If there is more than one, just put them in in the same order as in the query.

HERIOT WATT UNIVERSITY

Returning the results

Now we know how to do that, we can go back to returning all relevant results. First, we see if `$selected_site_type` has been set - i.e., whether the user has made a selection.

If so, we return the relevant results. If not, there is nothing to do.

```
if ($selected_site_type) {
}
```

Returning the results

If we do have a value, first we need to start making the table. Let's make a give the table a border to make it look nice, so instead of <table></table>, we start with <table border = 1>. Then we add the header line with the names of the columns, and left align so that it looks nice.

```
echo "<table border = 1>";
  echo "<tr><th align='left'>Site Name</th><th align='left'>City</th></tr>";
```

Returning the results

Now we have to populate the rest of the table with the results we extract from the database, so we can put our prepared query in here and execute it:

```
$stmt = $pdo->prepare("SELECT name, city FROM Site WHERE type = ?");
  $stmt->execute([$selected_site_type]);
```

HERIOT WATT UNIVERSITY

Returning the results

Finally, as long as `$stmt` has anything left in it, we print a row of it's first element, printing first the name of the site and then its city.

```
while ($row = $stmt->fetch()) {
    echo "<tr><td>" . $row["name"] . "</td><td>" . $row["city"] . "</td></tr>";
  }
```

Full code for returning the result

```
if ($selected_site_type) {
  echo "<table border = 1>";
  echo "<tr><th align='left'>Site Name</th><th align='left'>City</th></tr>";

  $stmt = $pdo->prepare("SELECT name, city FROM Site WHERE type = ?");
  $stmt->execute([$selected_site_type]);

  while ($row = $stmt->fetch()) {
    echo "<tr><td>" . $row["name"] . "</td><td>" . $row["city"] . "</td></tr>";
  }
}

echo "</table>";
echo "</body></html>";
```

HERIOT WATT UNIVERSITY

Full code for the whole process

- This is too long to see easily on a slide - take a look at sites.php

HERIOT WATT UNIVERSITY

Summary

- Today, we looked at HTML tables and then walked through code to create a drop-down menu populated from a database and return information based on that choice.
- You will be doing similar things in your assignment. You won’t have to do it from scratch, but can take inspiration from what we have done today.