CSE 154

LECTURE 16: SQL AND PHP/HTML

Querying a Database in PHP with PDO

```
$name = new PDO("dbprogram:dbname=database;host=server",
                                        username, password);
$name->query("SQL query");                                  PHP
```

```
# connect to world database on local server
$db = new PDO("mysql:dbname=world;host=localhost", "traveler",
                                                   "packmybags");
$db->query("SELECT * FROM countries WHERE population > 100000000;"); PHP
```

- PDO database library allows you to connect to many different database programs
  - replaces older, less versatile functions like `mysql_connect`
- PDO object's `query` function returns rows that match a query

Result rows: query

```php
$db = new PDO("dbprogram:dbname=database;host=server", username,
                                                       password);
$rows = $db->query("SQL query");
foreach ($rows as $row) {
  do something with $row;
}
```

PHP

- `query` returns all result rows
  - each row is an associative array of [column name -> value]
  - example: `$row["population"]` gives the value of the `population` column

A complete example

```php
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$rows = $db->query("SELECT * FROM actors WHERE last_name LIKE 'Del%'");
foreach ($rows as $row) {
  ?>
  <li> First name: <?= $row["first_name"] ?>,
       Last name:  <?= $row["last_name"]  ?> </li>
  <?php
}
```

PHP

- First name: Benicio, Last name: Del Toro
- First name: Michael, Last name: Delano
- ...

output

PDO object methods

name	description
query	performs a SQL SELECT query on the database
exec	performs a SQL query that modifies the database (INSERT, DELETE, UPDATE, etc.)
getAttribute, setAttribute	get/set various DB connection properties
quote	encodes a value for use within a query

Including variables in a query

```
# get query parameter for name of movie
$title = $_GET["movietitle"];
$rows = $db->query("SELECT year FROM movies
                    WHERE name = '$title'");
```

- you should not directly include variables or query parameters in a query
- they might contain illegal characters or SQL syntax to mess up the query

Quoting variables

```php
# get query parameter for name of movie
$title = $_GET["movietitle"];
$title = $db->quote($title);
$rows = $db->query("SELECT year FROM movies
                    WHERE name = $title");
```

PHP

- call PDO's `quote` method on any variable to be inserted
- `quote` escapes any illegal chars and surrounds the value with `'` quotes
- prevents bugs and security problems in queries containing user input

Database/query errors

```php
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");
# FALSE
```

- database commands can often fail (invalid query; server not responding; etc.)
- normally, PDO commands fail silently by returning `FALSE` or `NULL`
- but this makes it hard to notice and handle problems

Exceptions for errors

```php
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");
# kaboom!
```

- using `setAttribute`, you can tell PDO to throw (generate) a `PDOException` when an error occurs
- the exceptions will appear as error messages on the page output
- you can **catch** the exception to gracefully handle the error

Catching an exception

```php
try {
  statement(s);
} catch (ExceptionType $name) {
  code to handle the error;
}
```

Example with error checking

```php
try {
  $db = new PDO("mysql:dbname=imdb_small", "jessica","guinness");
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");
  foreach ($rows as row) { ... }
} catch (PDOException $ex) {
  ?>
  <p>
    Sorry, a database error occurred. Please try again later.
  </p>
  <p>(Error details: <?= $ex->getMessage() ?>)</p>
  <?php
}
```

PHP

PDOStatement methods

The $rows variable returned by PDO's query method is technically not an array but an object of type PDOStatement. It can be foreach-ed over like an array, but it also has the following methods:

columnCount()	number of columns in the results
fetch()	return the next row from the results
fetchColumn(number)	return the next column from the results
rowCount()	number of rows returned by the query

```php
$rows = $db->query("...");    # query omitted
if ($rows->rowCount() > 0) {
  $first_row = $rows->fetch();
  ...
}
```

HTML tables: <table>, <tr>, <td>

A 2D table of rows and columns of data (block element)

```
<table>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```

HTML

1,1	1,2 okay
2,1 real wide	2,2

output

- `table` defines the overall table, `tr` each row, and `td` each cell's data
- tables are useful for displaying large row/column data sets
- NOTE: tables are sometimes used by novices for web page layout, but this is not proper semantic HTML and should be avoided

Table headers, captions: <th>, <caption>

```
<table>
  <caption>My important data</caption>
  <tr><th>Column 1</th><th>Column 2</th></tr>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```

HTML

My important data

Column 1	Column 2
1,1	1,2 okay
2,1 real wide	2,2

output

- `th` cells in a row are considered headers; by default, they appear bold
- a `caption` at the start of the table labels its meaning

Styling tables

```
table { border: 2px solid black; caption-side: bottom; }
tr { font-style: italic; }
td { background-color: yellow; text-align: center; width: 30%; }
```

Column 1	Column 2
1,1	1,2 okay
2,1 real wide	2,2

My important data

output

- all standard CSS styles can be applied to a table, row, or cell
- table specific CSS properties:
  - `border-collapse`, `border-spacing`, `caption-side`, `empty-cells`, `table-layout`

The border-collapse property

```
table, td, th { border: 2px solid black; }
table { border-collapse: collapse; }
```
CSS

Without `border-collapse`

Column 1	Column 2
1,1	1,2
2,1	2,2

With border-collapse

Column 1	Column 2
1,1	1,2
2,1	2,2

- by default, the overall table has a separate border from each cell inside
- the `border-collapse` property merges these borders into one

The rowspan and colspan attributes

```HTML
<table>
  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td colspan="2">1,1-1,2</td>
    <td rowspan="3">1,3-3,3</td></tr>
  <tr><td>2,1</td><td>2,2</td></tr>
  <tr><td>3,1</td><td>3,2</td></tr>
</table>
```

HTML

Column 1	Column 2	Column 3
1,1-1,2		1,3-3,3
2,1	2,2	
3,1	3,2	

HTML

- `colspan` makes a cell occupy multiple columns; `rowspan` multiple rows
- `text-align` and `vertical-align` control where the text appears within a cell

Column styles: <col>, <colgroup>

```
<table>
  <col class="urgent" />
  <colgroup class="highlight" span="2"></colgroup>

  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td>1,1</td><td>1,2</td><td>1,3</td></tr>
  <tr><td>2,1</td><td>2,2</td><td>2,3</td></tr>
</table>
```

HTML

Column 1	Column 2	Column 3
1,1	1,2	1,3
2,1	2,2	2,3

output

- `col` tag can be used to define styles that apply to an entire column (self-closing)
- `colgroup` tag applies a style to a group of columns (NOT self-closing

Don't use tables for layout!

- (borderless) tables appear to be an easy way to achieve grid-like page layouts
  - many "newbie" web pages do this (including many UW CSE web pages...)
- but, a `table` has semantics; it should be used only to represent an actual table of data
- instead of tables, use `div`s, widths/margins, floats, etc. to perform layout

- tables should not be used for layout!
- tables should not be used for layout!!
- TABLES SHOULD NOT BE USED FOR LAYOUT!!!
- TABLES SHOULD NOT BE USED FOR LAYOUT!!!!