Lecture 20

A Fuller Example:

PHP PDO SQLite CSV Bootstrap

* Course logo spider web photograph from Morguefile openstock photograph by Gabor Karpati, Hungary.
This example walks through using a SQLite database and PHP. Use it to study the relationship between PHP objects and elements in a database along with common operations such as sorting, searching, updating and deleting.

**About the list**

Rolling Stone is an American magazine devoted to music and popular culture. It was founded in San Francisco in 1967. In 2003, Rolling Stone published an article describing what it considered to be the top 500 music albums of all time. In 2012, this list was updated and released in book form.
Create Revisited

Whole page body fits on one screen.

Use PHP as a programming language.

And don’t try to read each line, just notice structure.

```
<ol>
  <li>Start a connection</li>
  <?php if (!$dbh = setupMusicConnection()) { die; } ?>
  <li>Delete old tables. Here we want to use `exec()` because the drop command does not return a query set but instead number of rows effected. (Note: Always returns 0 when deleting or creating tables.)</li>
  <li>Delete artist:</li>
  <?php dropTableByName ( "artist" );
       dropTableByName ( "album" ); ?>
  </li>
  <li>Create Tables. The SQL here is the same as if you were doing it via command line</li>
  <li>Artist Table then Album Table</li>
  <?php createTableArtist ();
       createTableAlbum (); ?>
  </li>
  <li>Load Values Into Database</li>
  <?php loadMusicIntoEmptyDatabase (); ?>
  </li>
  <li>Check Values (should be 100 albums but less artist since The Beatles and Led Zeppelin Rock).
      <p class="bg-success" /></p>
      <?php
       $artist_num = $dbh->query ( "SELECT count(*) FROM artist" );
       $album_num = $dbh->query ( "SELECT count(*) FROM album" );
       echo "Number of artists: ", $artist_num->fetchColumn () . '<br />';
       echo "Number of albums: ", $album_num->fetchColumn () . '';
      ?></li>
</ol>
```
```php
function createTableArtist() {
    $sql = "CREATE TABLE artist (
    artist_id INTEGER PRIMARY KEY ASC,
    artist_name varchar(50)"
createTableGeneric ( $sql );
}

function createTableAlbum() {
    $sql = "CREATE TABLE album ( 
    album_id INTEGER PRIMARY KEY ASC,
    artist_id int(5),
    title varchar(255),
    year int(4),
    rank int(5),
    FOREIGN KEY (artist_id) REFERENCES artist(artist_id))";
createTableGeneric ( $sql );
}

function createTableGeneric($sql) {
    global $dbh;
    $status = $dbh->exec ( $sql );
    if ($status !== FALSE) {
        echo '<pre class="bg-danger">'
        print_r ( $dbh->errorInfo () );
        echo '</pre>,'
    } else {
        echo '<pre class="bg-success">'
        echo 'Number of rows effected: ' . $status;
        echo '</pre>','
    }
}
```

- SQL Command laid out in readable format.
- Relation from album to artist through FOREIGN KEY.
- Important detail, notice 'global' declaration.
'prepare’ for repetition

```php
function loadMusicIntoEmptyDatabase() {
    global $dbh;
    require "source_file/list.php";
    $albums = getAlbumsFromFile();
    $artists_ids = array(); // Stores artists and SQL index
    $sql_artist = "INSERT INTO artist (artist_name) VALUES (?)";
    $sql_album = "INSERT INTO album (artist_id, title, year, rank) VALUES (:artist_id, :title, :year, :rank)"
    // This allows caching of statements and optimized queries
    $artist_stm = $dbh->prepare ( $sql_artist );
    $album_stm = $dbh->prepare ( $sql_album );
    foreach ( $albums as $current_album ) {
        // 1. Check to make sure artist hasn't already been added
        if ( key_exists ( $current_album ['Band'], $artists_ids ) ) {
            $artist_id = $artists_ids [$current_album ['Band']];
        } else {
            testedInsertArtistName ( $current_album ['Band'], $artist_stm );
            $artist_id = $dbh->lastInsertId ( 'artist_id' );
            $artists_ids [$current_album ['Band']] = $artist_id;
        }
        testedInsertAlbum ( $current_album, $artist_id, $album_stm );
    }
}
```
function loadMusicIntoEmptyDatabase()
{
    global $dbh;
    require "source_file/list.php";
    $albums = getAlbumsFromFile();
    $artists_ids = array(); // Stores artists and SQL index
    $sql_artist = "INSERT INTO artist (artist_name) VALUES (?)";
    $sql_album = "INSERT INTO album (artist_id, title, year, rank)
                      VALUES (:artist_id, :title, :year, :rank)";
    // This allows caching of statements and optimized queries
    $artist_stmt = $dbh->prepare ( $sql_artist );
    $album_stmt = $dbh->prepare ( $sql_album );
    foreach ( $albums as $current_album ) {
        // 1. Check to make sure artist hasn't already been added
        if (key_exists ( $current_album ['Band'], $artists_ids )) {
            $artist_id = $artists_ids [ $current_album ['Band'] ];
        } else {
            testedInsertArtistName ( $current_album ['Band'], $artist_stmt );
            $artist_id = $dbh->lastInsertId ( 'artist_id' );
            $artists_ids [ $current_album ['Band'] ] = $artist_id;
        }
        testedInsertAlbum ( $current_album, $artist_id, $album_stmt );
    }
}
Execute Prepared Stmt.

```php
function testedInsertArtistName($name, $stmt) {
    global $dbh;
    if (! $stmt->execute ( array ($name ) )) {
        echo '<pre class="bg-danger">';
        print_r ( $dbh->errorInfo () );
        echo '</pre>,'
    }
}

function testedInsertAlbum($album, $aid, $stmt) {
    global $dbh;
    if (! $stmt->execute ( array (
        ':artist_id' => $aid,
        ':title' => $album ['Title'],
        ':year' => $album ['Year'],
        ':rank' => $album ['Rank']
    ) )) {
        echo '<pre class="bg-danger">';
        print_r ( $dbh->errorInfo () );
        echo '</pre>,'
    }
}
```

Single element array

Key-value pairs array
Tabs alter order.  

Pagination.
A Building Block

Track and highlight a selection.

From W3Schools, recall..

```php
<?php
if(!isset($page_name)){
    $page_name = "home";
}

function getIsActive($page_name, $nav_name){
    if($page_name == $nav_name){
        return ' class="active"';
    }
    return "";
}

?>

<nav>
    <ul class="nav nav-tabs">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">Menu 1</a></li>
        <li><a href="#">Menu 2</a></li>
        <li><a href="#">Menu 3</a></li>
    </ul>
    <ul class="nav nav-pills nav-stacked">
        <li><a href="index.php">Start Page</a></li>
        <li><a href="create.php">Create Music DB</a></li>
        <li><a href="select.php">Review Music</a></li>
        <li><a href="search.php">Search Music</a></li>
        <li><a href="update.php">Update Album</a></li>
        <li><a href="delete.php">Delete Album</a></li>
        <li><a href="lec20.zip">Download Code</a></li>
    </ul>
</nav>
```
Maintain State Information

Think through what this particular HTML anchor is actually doing!

```php
<?php
/*
 * select.php
 * Intro to PHP and SQLite - Simple SELECT Statements
 * Jaime Ruiz initial March 2, 2015
 */
require_once "inc/page_setup.php";
$title = "Databases with PDO and SQLite";
$page_name = "select";
include 'inc/header.php';
$current_tab = "rank";
$current_page = 1;
$num_per_page = 25;

/* By design on this page the current tab may only have one of three values:
 * they are: "rank", "artist" and "year". These correspond to field names
 * in the table returned by the SELECT statement.
 */
if (isset ( $_GET ["b"] )) {
    $current_tab = strip_tags ( $_GET ["b"] );
}
if (isset ( $_GET ["p"] )) {
    $current_page = intval ( $_GET ["p"] );
}
```
Study and experiment on your own time, a bit more than a simple lecture example.
Pagination Inwards: SQL

Pay particular attention to the query string construction.

```php
function getAlbumsByField($field, $num_returned = 25, $offset = 0) {
    $sql = "SELECT album_id, artist_id, title, year, rank, artist_name AS artist
            FROM album NATURAL JOIN artist
            ORDER BY $field ASC LIMIT $num_returned OFFSET $offset";
    $result = $this->query ( $sql );
    if ($result === FALSE) {
        // Only doing this for class. Would never do this in real life
        echo '<pre class="bg-danger">';
        print_r ( $this->errorInfo () );
        echo '</pre>'; return array ( );
    }
    $albums = array ( );
    foreach ( $result as $row ) {
        $albums [] = Album::getAlbumFromRow ( $row );
    }
    return $albums;
}
```
# Search for Music

## CT 310: PDO & SQLite Example

### Search the Top Albums of All Time

Search the Top Albums of All Time

<table>
<thead>
<tr>
<th>Rank</th>
<th>Title</th>
<th>Year</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sgt. Pepper's Lonely Hearts Club Band</td>
<td>1967</td>
<td>The Beatles</td>
</tr>
<tr>
<td>3</td>
<td>Revolver</td>
<td>1966</td>
<td>The Beatles</td>
</tr>
<tr>
<td>5</td>
<td>Rubber Soul</td>
<td>1965</td>
<td>The Beatles</td>
</tr>
<tr>
<td>10</td>
<td>The Beatles (The White Album)</td>
<td>1968</td>
<td>The Beatles</td>
</tr>
<tr>
<td>14</td>
<td>Abbey Road</td>
<td>1969</td>
<td>The Beatles</td>
</tr>
<tr>
<td>39</td>
<td>Please Please Me</td>
<td>1963</td>
<td>The Beatles</td>
</tr>
<tr>
<td>53</td>
<td>Meet The Beatles!</td>
<td>1964</td>
<td>The Beatles</td>
</tr>
</tbody>
</table>
Search Internals: Form

```php
<div class="container">
  <div class="row">
    <div class="col-sm-3">
      <?php include "inc/sidenav.php"; ?>
    </div>
    <div class="col-sm-9">
      <h2>Search the Top Albums of All Time</h2>
      <form class="form-inline" method="get">
        <div class="form-group">
          <label class="sr-only" for="q">Search</label>
          <input type="text" class="form-control" name="q" id="q" placeholder="Search" />
          <input type="hidden" name="b" value="<?php echo $current_tab; ?>" />
        </div>
        <button type="submit" class="btn btn-default">
          <span class="glyphicon glyphicon-search" aria-label="Submit"/>
        </button>
      </form>
    </div>
  </div>
</div>

<?php
  if(isset($_GET["q"])):
    $tab_urls = Utility::removeParameterFromUrl("b");
    $tab_urls = Utility::makeSureURLIsQueryString($tab_urls);
  ?>
  <nav>
    <ul class="nav nav-tabs">
      <li><?php echo isActive($current_tab, "rank"); ?> <a href="<?php echo $tab_urls; ?>&b=rank">By Rank</a></li>
      <li><?php echo isActive($current_tab, "artist"); ?> <a href="<?php echo $tab_urls; ?>&b=artist">By Artist</a></li>
      <li><?php echo isActive($current_tab, "year"); ?> <a href="<?php echo $tab_urls; ?>&b=year">By Year</a></li>
    </ul>
  </nav>
```
Pay particular attention to the query string construction.

```php
function searchForResultsAndSort($query_term, $sort_col = "rank", $num_returned = 25, $offset = 0) {
    $query_term = SQLite3::escapeString($query_term);
    switch ($sort_col) {
        case "rank":
            $sort_col = "rank";
            $sec_sort = "title";
            break;
        case "year":
            $sort_col = "year";
            $sec_sort = "rank";
            break;
        case "artist":
            $sort_col = "artist";
            $sec_sort = "rank";
            break;
    }

    $sql = "SELECT album_id, artist_id, title, year, rank, artist_name AS artist
            FROM album NATURAL JOIN artist
            WHERE (title LIKE '%$query_term%' OR artist LIKE '%$query_term%' OR year = '$query_term')
            ORDER BY $sort_col ASC, $sec_sort ASC
            LIMIT $num_returned OFFSET $offset";
    $result = $this->query($sql);
    if ($result === FALSE) {
        echo $sql;
        echo '<pre class="bg-danger">
        print_r($this->errorInfo());
        echo '</pre>';
        return array();
    }

    $albums = array();
    foreach ($result as $row) {
        $albums[] = Album::getAlbumFromRow($row);
    }
    return $albums;
}
```