PHP and MySQL

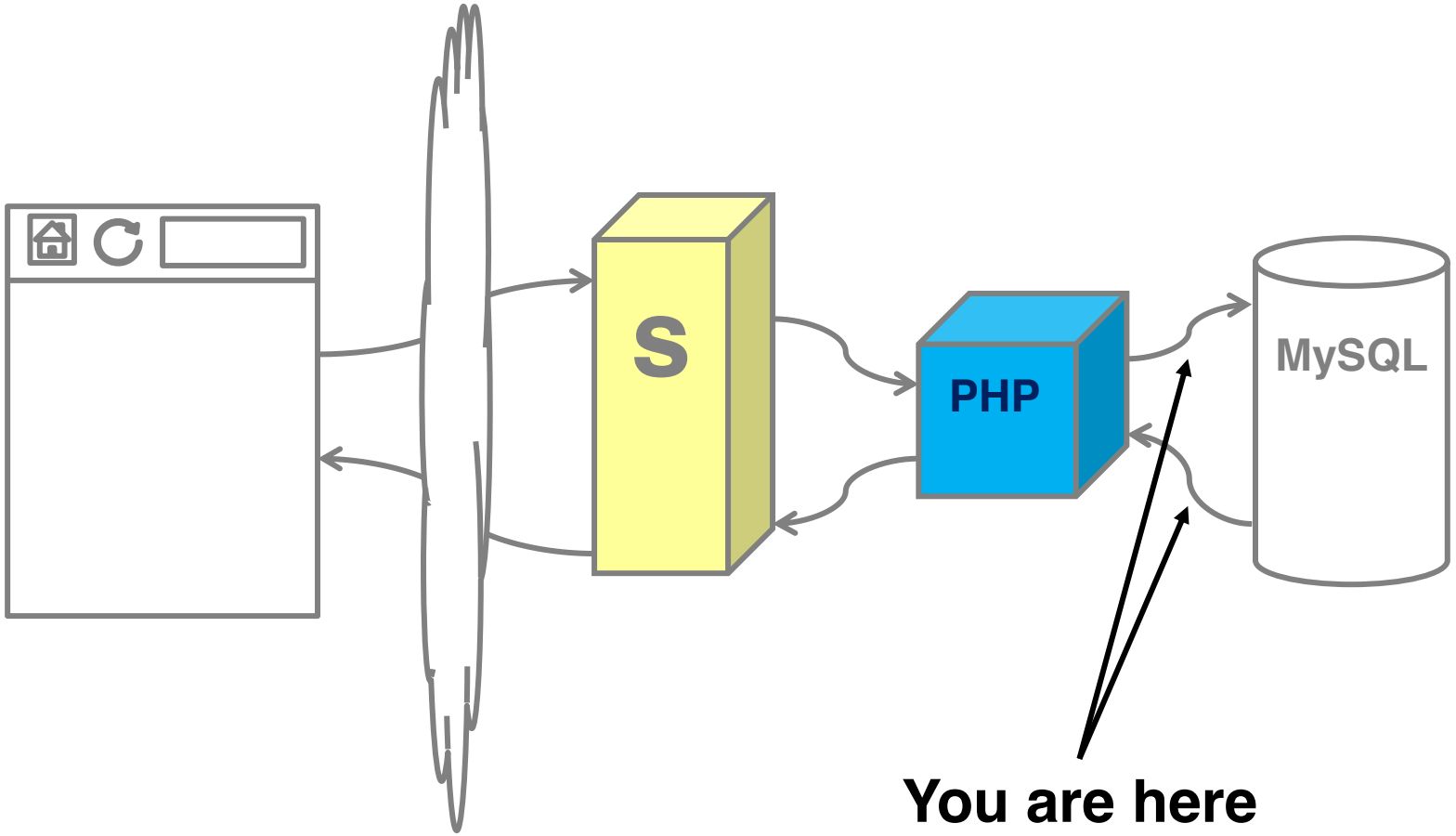
INFO/CS 2300:
Intermediate Web Design and Programming

HW2

Student comments

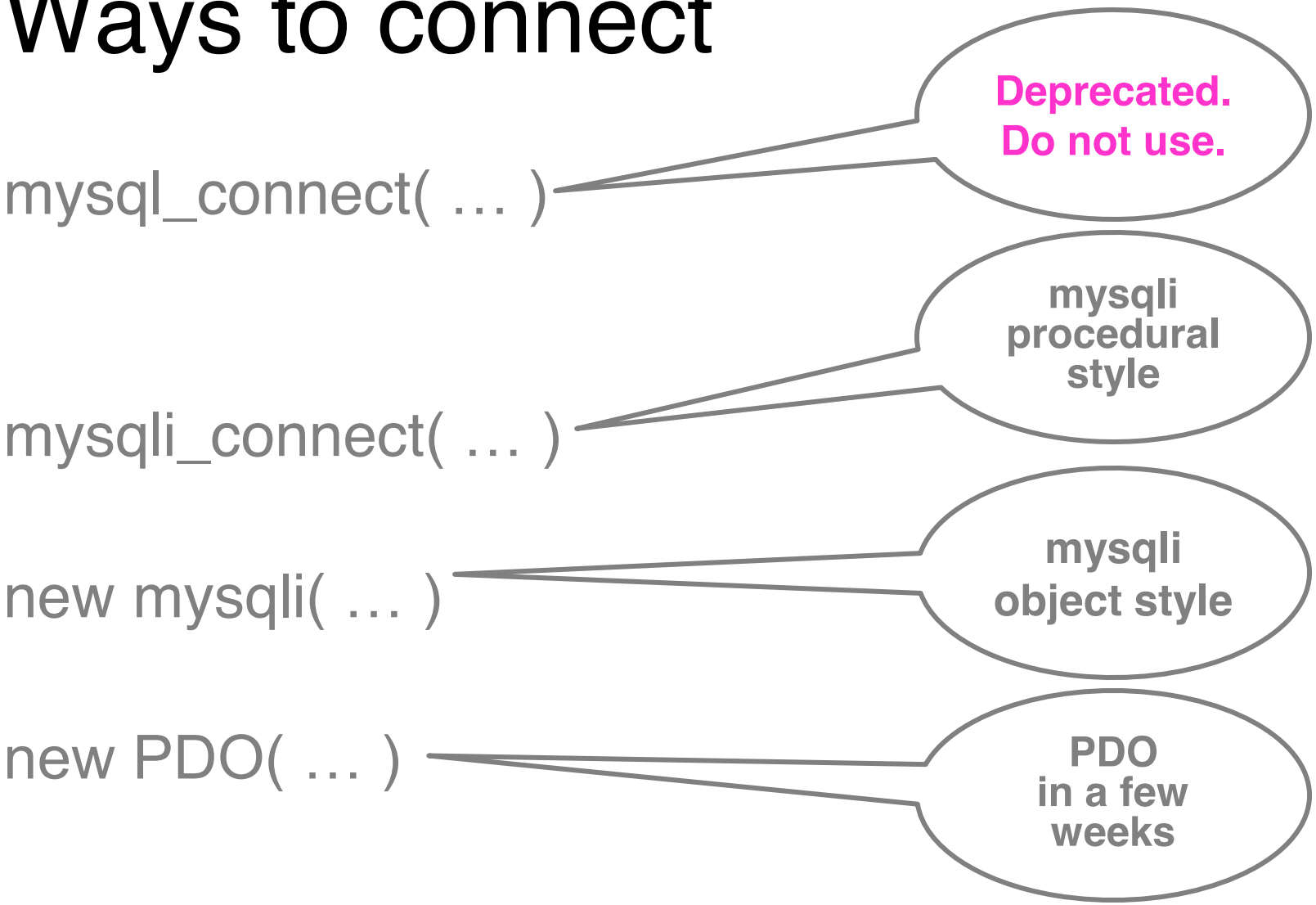
- This was a really hard assignment. Take it easy on me.
- Cool assignment i love you guys

How it works



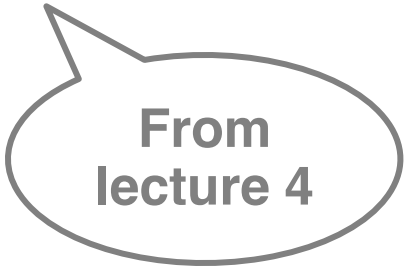
Using MySQL from PHP

Ways to connect



Remember PHP objects?

I'll be using the object form of mysqli



- We can make *instances* of an object:

  ```
  $movie = new Movie();
  ```

- *Fields* are data associated with an instance of an object:

  ```
  $movie>title = …
  ```

- *Methods* are functions associated with the object.

  ```
  print $movie->the_question();
  ```

Connecting

To use MySQL from PHP you first need to create an instance of a *mysqli object*. We call the constructor with information on how to connect to the database.

$mysqli = new mysqli(*hostname, username, password, database);*

Returns a mysqli instance.

Click In!

In the movies example, where does it make sense to put these values?

A. In the 'new mysqli' statement each time I use mysqli
B. At the beginning of the 2 files where I use mysqli
C. In a separate file

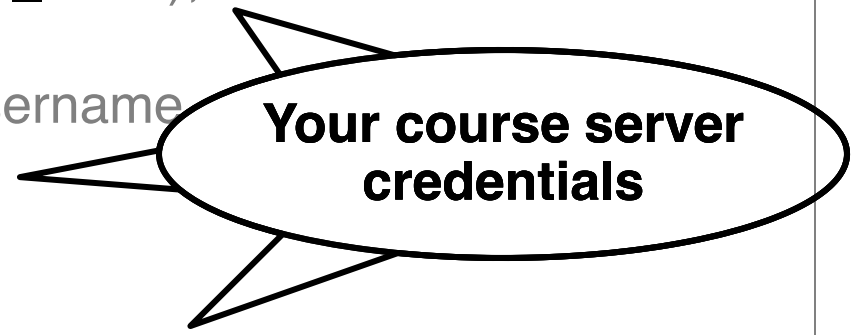
config.php

```
<?php  // ** MySQL connection settings ** //
    // database host
    define('DB_HOST', 'localhost');

    // database name
    define('DB_NAME', 'info230_SP15_netid');

    // Your MySQL / Course Server username
    define('DB_USER', 'netidsp15');

    // ...and password
    define('DB_PASSWORD', 'your_password');
?>
```



movies.php

```
require_once 'config.php';
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
```

Issuing SQL commands

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
```

```
$result = $mysqli->query("SELECT * FROM Movies");
```

The mysqli object method query(*sqlquery*) issues *sqlquery* to the MySQL DB.

For INSERT, UPDATE, DELETE, returns true if successful, false if not

For SELECT, returns instance of *result object* if successful, false if not.

Getting results

$result = $mysqli->query("SELECT * FROM Movies");

Given the result object *result,* we can fetch the associated data using the result object.

$row = $result->fetch_row();

returns a regular array (e.g. title in $row[0], year in $row[1], etc.)

$row = $result->fetch_assoc();

returns an associative array (e.g. $row['title'] has the value for title, etc.)

Both return false if no more rows left in the result.

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {




}
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {




}
```

First time through loop:

$row = array('Chicago', 2002, 113)

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {




}
```

Second time through loop:

$row = array('The Return of the King', 2003, 201)

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {




}
```

Third time through loop:

$row = array('Million Dollar Baby', 2004, 132)

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {




}
```

Last time through loop:

$row = false

print('</tbody></table>');

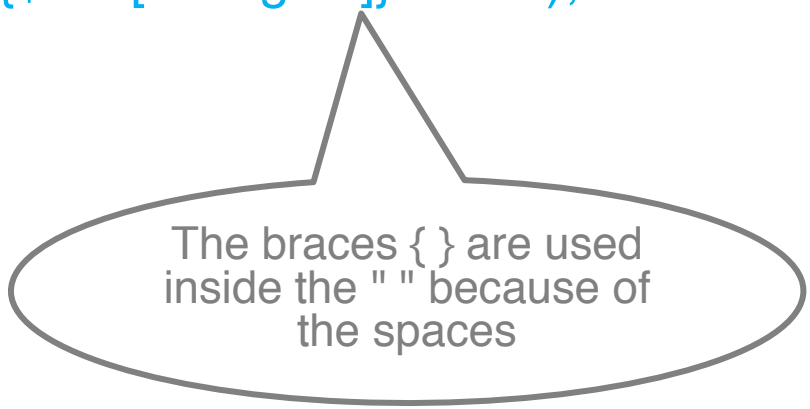
Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_row() ) {
    print( '<tr>' );
        foreach( $row as $value ) {
            print( "<td>$value</td>" );
        }
    print( '</tr>' );
}
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Using fetch_assoc()

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_assoc() ) {
    print( '<tr>' );
        print( "<td>{$row[ 'Title' ]}</td>" );
        print( "<td>{$row[ 'Year' ]}</td>" );
        print( "<td>{$row[ 'Length' ]}</td>" );
    print( '</tr>' );
}
```



Alternatively

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>…</thead><tbody>')
while ( $row = $result->fetch_assoc() ) {
    print( '<tr>' );
        $title = $row[ 'Title' ];
        $year = $row[ 'Year' ];
        $length = $row[ 'Length' ];

        print( "<td>$title</td>" );
        print( "<td>$year</td>" );
        print( "<td>$length</td>" );
    print( '</tr>' );
}
```

Number of rows returned

$result->num_rows

Contains the number of rows in the table given by *result*.

E.g.

```
$result = $mysqli->query("SELECT * FROM Movies");
$row_count = $result->num_rows;
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Close the db

```
$mysqli->close();
```

Closes connection to DB

- Not necessary most of the time since PHP does it eventually on its own

Using other parts of SQL

Searching

Suppose we want to add search functionality. What should we do?

- HTML / PHP form
- SQL: LIKE, REGEXP

Click In!

When building a page with a form that submits to the same page, which usually makes sense to do first in the code?

A. Process (read, save, update) the form input data
B. Display the form
C. Doesn't matter

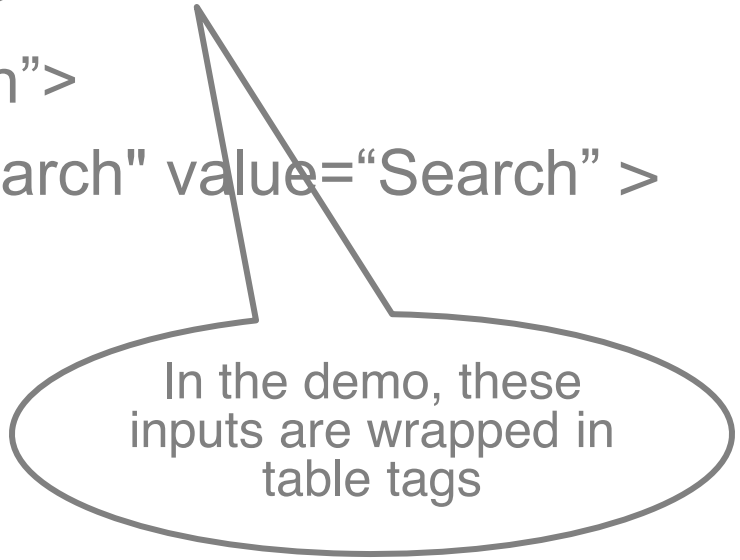
Click In!

If I add a second criteria to my search, which makes more sense in this context?

A. More results
B. Fewer results

Searching - HTML

```
<form action="movies-results.php" method="post">
    <input type=“text” name=“title”>
    <input type=“text” name=“year”>
    <input type=“text” name=“length”>
    <input type=“submit” name="search" value=“Search” >
</form>
```



settings.php

```
//Array of fields used
$fields = array(
    array(
            'term' => 'title',
            'heading' => 'Title',
            'filter' => FILTER_SANITIZE_STRING,
    ),
    array(
            'term' => 'year',
            'heading' => 'Year',
            'filter' => FILTER_SANITIZE_NUMBER_INT,
    ),
    array(
            'term' => 'length',
            'heading' => 'Length (min)',
            'filter' => FILTER_SANITIZE_NUMBER_INT,
    ),
);
```

Searching - PHP

```
//Build an array of search clauses
$searches = array();
foreach( $fields as $field ) {
    $search_term = $field[ 'term' ];
    $filter = $field[ 'filter' ];

    //Does this term exist in the POST data submitted by the search form?
    if( !empty( $_POST[ $search_term ] ) ) {
        //Get the value for this term from the POST data
        $search_value = filter_input(INPUT_POST, $search_term, $filter);

        //Add the search clause
        $searches[] = "$search_term REGEXP '$search_value'";
    }
}
```

Searching - PHP

index.php

```
//Starting SQL
$sql = 'SELECT * FROM Movies';

//Were there search terms?
if( !empty( $searches ) ) {
    //Build the WHERE clause
    $sql .= ' WHERE ';

    //Add the searches by joining any elements together with AND
    $sql .= implode(' AND ', $searches );
}
```

Sorting

Suppose we want to allow the user to sort the entries by the various fields. How can we do that?

- HTML: links
- PHP: $_GET
- SQL: ORDER BY

Sort form

```
<thead>
    <th>
        <a href="?sort=title">Title</a>
    </th>
    <th>
        <a href="?sort=year">Year</a>
    </th>
    <th>
        <a href="?sort=length">Length</a>
    </th>
</thead>
```

Sort PHP

```
//Try to get the 'sort' parameter from the URL
    //and filter out bad stuff
    //Better security would make sure it is one of our expected $fields
$sort = filter_input( INPUT_GET, 'sort', FILTER_SANITIZE_STRING );

//Is this sorted?  $sort will be empty if the parameter was not set in the URL
if ( !empty( $sort ) ) {
    $sql .= " ORDER BY $sort";
}
```

Adding items

Suppose we want to be able to add items to our list. What should we do?

- HTML Form
- PHP processing and data checking
- SQL INSERT

Adding - HTML

```
<input type="text" name="title">
<input type="text" name="year">
<input type="text" name="length">
```

Adding - PHP

Assume the POST data is processed into an array that looks like this

```
$field_values = array (
    'title' => 'Into the Woods',
    'year' => '2014',
    'length' => '124'
)
```

```
//Get an array of the field names that have data
$field_name_array = array_keys( $field_values );

//Comma delimited list of fields
//equivalent to $field_list = "title, year, length";
$field_list = implode( ',', $field_name_array );

//comma delimited values - need quotes around values
$value_list = implode( "','", $field_values );

//Build the SQL for adding a movie
//later we'll improve security and quoting
$sql = "INSERT INTO movies ( $field_list ) VALUES (
    '$value_list' );";
```

Autonumber INSERT value

```
$sql = "INSERT INTO movies ( title,year,length )
VALUES ( 'Into the Woods', '2014', '124' );";
if( $mysqli->query( $sql ) ) {
    $new_id = $mysqli->insert_id;
}
```

Contains the value of the new autonumber id cresated by MySql

Modifying

Suppose we want to let the user edit the various entries. How can we do that?

- HTML: link and additional form
- PHP: additional form processing
- SQL: UPDATE

Modifying - HTML

```
<input type="hidden" name="movie_id" value='17'>
<input type="text" name="title" value="Into the
  Woods">
<input type="text" name="year" value="2014">
<input type="text" name="length" value="124">
```

Modifying - PHP

Assume the POST data is processed into an array that looks like this

```
$field_values = array (
    'movie_id' = 17,
    'title' => 'Into the Woods',
    'year' => '2014',
    'length' => '124'
)
```

Modifying – PHP & SQL

```
$update_fields = array();
foreach( $field_values as $field_name => $field_value ) {
    $update_fields[] = "$field_name = '$field_value'";
}
$sets = implode( ', ', $update_fields );

//Build the SQL for adding a movie
//later we'll improve security and quoting
$sql = "UPDATE movies SET $sets
                WHERE movie_id=$movie_id";
```

Debugging

Getting MySQL Errors

Various fields in the mysqli object *mysqli* contain error information: *mysqli->errno* contains an error code (or 0 if no error), and *mysqli->error* contains a string with the error.

E.g.

```
if ( $mysqli->errno ) {
    print($mysqli->error);
    exit();
}
```

Project 3: Photo gallery

Now you can practice your skills by writing your own photo gallery website!

- Part 1: Due 3/10
  - DB Schema, Set up tables in your 2300 server DB, draft basic navigation of pages, initial code to display photos
- Part 2: Due 3/17
  - Add code to display/add albums, add photos to albums
- Part 3: Due 3/24
  - Final working site, with a secure login for photo uploading.

Details on Piazza

Review

- We can now use the MySQL DBMS from within PHP by using the mysqli object and its methods (e.g.query, fetch_row, fetch_assoc).
- We can use all our favorite SQL queries (SELECT, INSERT, UPDATE, etc.) to generate results for our page.