

INFO I-CE9224: Introduction to PHP Programming

Session 9
August 15, 2012

Resources

<http://davehauenstein.com/nyu/INFOI-CE9224-2012-Summer>

Username: nyuscps
Password: \$nyuscps\$

Class 9 Agenda

- Brief SQL Review
- Brief Introduction to Objects
- Connecting to a MySQL Database using PHP
- Querying the database using PHP
 - Retrieving data and looping over results
 - Inserting / Updating database rows
 - Escaping and Sanitizing data
- Importance of validation and error handling
- Intro to file includes.

SQL Review

SQL - SELECT

```
SELECT * FROM table_name;
```

```
SELECT * FROM table_name  
ORDER BY col ASC|DESC;
```

```
SELECT col_name1, col_name2 FROM table_name;
```

```
SELECT col_name1, col_name2 FROM table_name  
WHERE col_name3 = 'some_value';
```

```
SELECT col_name1, col_name2 FROM table_name  
WHERE col_name3 = 'some_value' LIMIT 5, 15;
```

SQL - SELECT Aliasing Columns and Tables

```
SELECT col_name as someCol  
FROM table_name;
```

```
SELECT t.col_name as someCol  
FROM table_name as t;
```

SQL - SELECT Limiting Number of Rows

Return the first 5 rows.

```
SELECT * FROM table_name LIMIT 5;
```

Return the first 5 rows.

```
SELECT * FROM table_name LIMIT 0, 5;
```

Return 5 rows starting at the 5th row.

```
SELECT * FROM table_name LIMIT 5, 5;
```

SQL - SELECT

Ordering the Results

Order Ascending (default)

```
SELECT * FROM table_name ORDER  
BY col_name / ASC;
```

Order Descending

```
SELECT * FROM table_name ORDER  
BY col_name / DESC;
```


SQL - WHERE Clause

Using AND and OR

```
SELECT col_name1, col_name2 FROM table_name  
WHERE col_name3 = 'some_value';
```

```
SELECT col_name1, col_name2 FROM table_name  
WHERE col_name3 = 'val' AND col_name4 = 'val';
```

```
SELECT col_name1, col_name2 FROM table_name  
WHERE col_name3 = 'val' OR col_name4 = 'val';
```

SQL - UPDATE

Modifying Rows

```
UPDATE table_name SET col_name = 'new_value'  
WHERE col_name2 = 'some_value';
```

- Columns will be updated in the Rows that match the WHERE clause.
- This can be one or many rows at a time.
- The WHERE portion of the update usually uses a primary key column when updating one row.

SQL - UPDATE

Modifying Rows

UPDATE *table_name*

SET *col1* = 'value', *col2* = 'value', *col3* = 'value'

WHERE *col4* = 'value';

- Multiple columns can be updated at one time by separating *col* = 'value' by a comma.
- This is more efficient than running multiple UPDATE statements on the rows needing updates.

SQL - DELETE

Removing Rows

```
DELETE FROM table_name  
WHERE col = 'value';
```

- Rows will be deleted that match the WHERE clause.
- This can be one or many rows at a time.
- The WHERE portion of the update usually uses a primary key column when updating one row.

Classes & Objects

Classes & Objects

- Object Oriented Programming (OOP) vs. Procedural Programming (what we've been doing).
- When modeling properties and behaviors we use Classes.
- An object is a specific instance of the class.

Classes & Objects

- Objects are created to model real world things and objects. Think: Cat or Person Object.
- Objects model real world properties and real world behaviors.
- Cat properties: color, weight, breed
- Cat behaviors: meow, purr, run, sleep

- OOP uses encapsulation: specific types of functionality are encapsulated in the objects in which they belong.
- Objects can inherit functionality from one another.
- OOP makes code easier to understand and maintain.
- Person Class > Employee Class
- Employee Class > Manager Class
- Employee Class > Programmer Class

Classes & Objects

- PHP has built in objects that can be used for general purpose functionality.
- For example, interacting with a database (PDO, mysqli).
- Interacting with Dates and Times (DateTime).
- etc...

Syntax

```
$datetime = new DateTime();  
print_r($datetime);
```

```
DateTime Object  
(  
    [date] => 2012-05-17 13:54:06  
    [timezone_type] => 3  
    [timezone] => America/New_York  
)
```

Syntax

variable
containing
DateTime
object

new Keyword

DateTime Class

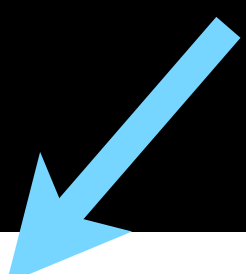
```
$datetime = new DateTime();  
print_r($datetime);
```

DateTime Object

```
(  
    [date] => 2012-05-17 13:54:06  
    [timezone_type] => 3  
    [timezone] => America/New_York  
)
```

Syntax

*Constructor
Argument*



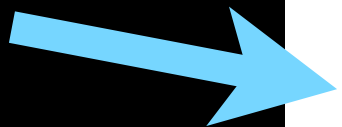
```
$datetime = new DateTime('tomorrow');  
print_r($datetime);
```

DateTime Object

```
(  
    [date] => 2012-05-18 00:00:00  
    [timezone_type] => 3  
    [timezone] => America/New_York  
)
```

Syntax

Calling a
Method



```
$date = new DateTime('now');  
echo $date->format('Y-m-d H:i:sP');
```

- Calling a method is like calling a function.
- Functions we've covered live in the “global namespace”
- Methods are functions that can only be called against an object.
- Notice the syntax: -> (hyphen, greater than) between the variable and the method.

More on Objects and
Classes next class.

MySQL and PHP

Using PDO

PHP and MySQL

- PHP has several built in mechanisms for interacting to MySQL.
- Two examples are mysqli (MySQL improved) and PDO (PHP Data Objects).
- PDO is what we'll use.
 - What is PDO?
 - How do we use it?

PDO

- Provides a set of classes and methods (functions called on objects) for interacting with a database.
- A layer of abstraction for talking to different database engines.
- Layer of abstraction? Can talk to MySQL and other databases using the same objects and classes.
- <http://www.php.net/manual/en/class.pdo.php>

Establishing a MySQL Connection Using PDO

- The PDO constructor takes 3 arguments:
 - Database Source Name (DSN)
 - Username
 - Password
- The DSN format looks like...
- `mysql:host=localhost;dbname=mydatabase`

Establishing a MySQL Connection Using PDO

```
$dsn = 'mysql:host=localhost;dbname=homework-db';  
$username = 'daveh';  
$password = 'davespw';  
$connection = new PDO($dsn, $username, $password);
```

\$connection now contains a PDO Object
We can now call methods on **\$connection**

Another Example...

```
$username = 'daveh';  
$password = 'davespw';  
$driver    = 'mysql';  
$host      = 'localhost';  
$db        = 'homework-db';  
  
$dsn = sprintf(  
    '%s:host=%s;dbname=%s',  
    $driver,  
    $host,  
    $db  
);  
  
$connection = new PDO($dsn, $username, $password);
```

PDO

- Once we have a connection, PDO allows you to do several things with the database.
- Most importantly, it allows you to run whatever query you'd like.
- `$connection->query($sql);`
- `$connection->exec($sql);`

PDO - *query()*

- `$data = $connection->query($sql);`
- Returns a *PDOStatement* object which can be iterated over.
- *PDOStatement* is similar to an array, in that you use loops to access elements (rows) stored within it.
- Returns false on failure.

PDO - *query()*

```
$connection = new PDO($dsn, $username, $password);

$sql = 'SELECT * FROM products LIMIT 2';
$data = $connection->query($sql);

foreach ($data as $row) {
    echo $row['product_name'] . "\n";
    echo $row['price'] . "\n\n";
}
```

```
Independence Day
19.99
```

```
Good Will Hunting
13.49
```

PDO - *query()*

PDOStatement

Iterate over PDOStatement

```
$connection = new PDO($dsn, $username, $password);  
  
$sql = 'SELECT * FROM products LIMIT 2';  
$data = $connection->query($sql);  
  
foreach ($data as $row) {  
    echo $row['product_name'] . "\n";  
    echo $row['price'] . "\n\n";  
}
```

Rows are associative arrays
keys are column names

PDO - `exec()`

- `$numRows = $connection->exec($sql);`
- Returns a *integer* value of how many rows were effected.
- This method should be used for performing update, delete, insert, and anything else that you don't expect to return a set of rows.
- Returns false on failure, should use `===` when determining whether 0 rows were effected, or there was an error.

PDO - `exec()`

```
$connection = new PDO($dsn, $username, $password);

$sql        = 'DELETE FROM orders';
$effected   = $connection->exec($sql);

if($effected !== false) {
    echo $effected . ' rows were deleted';
}
```

- Uses `exec()` rather than `query`.
- Returns the number of rows that were effected by the query.

PDO - Safe Queries

- Most of the time when inserting data into a database, what you're inserting comes from a form a user submitted.
- Sometimes when querying for data, what you're querying for is using user submitted data.
- Never trust user submitted data!
- We must make sure the data is sanitized, to prevent SQL Injection attacks.

SQL Injection attacks

- Malicious users may purposefully enter data that gives them access to your database.
- With access to the database, they can corrupt your data, delete it, and/or steal it.
- It's important to prevent this using PDO's prepared statements whenever using user input in queries.

Prepared Statements

- Rather than place user inputted data directly into a query, use placeholders.
- Give PDO a map of the placeholders to the user inputted data.
- PDO will take care of making sure the data is sanitized for you.
- New PDO functions: `execute` and `prepare`.

PDO - Safe Queries

```
$connection = new PDO($dsn, $username, $password);

$sql = 'SELECT name, color, calories FROM fruit
       WHERE calories < :calories AND colour = :colour';

$stmt = $connection->prepare($sql);
$stmt->execute(array(
    ':calories' => 150,
    ':colour'   => 'red'
));
$data = $stmt->fetchAll();
```

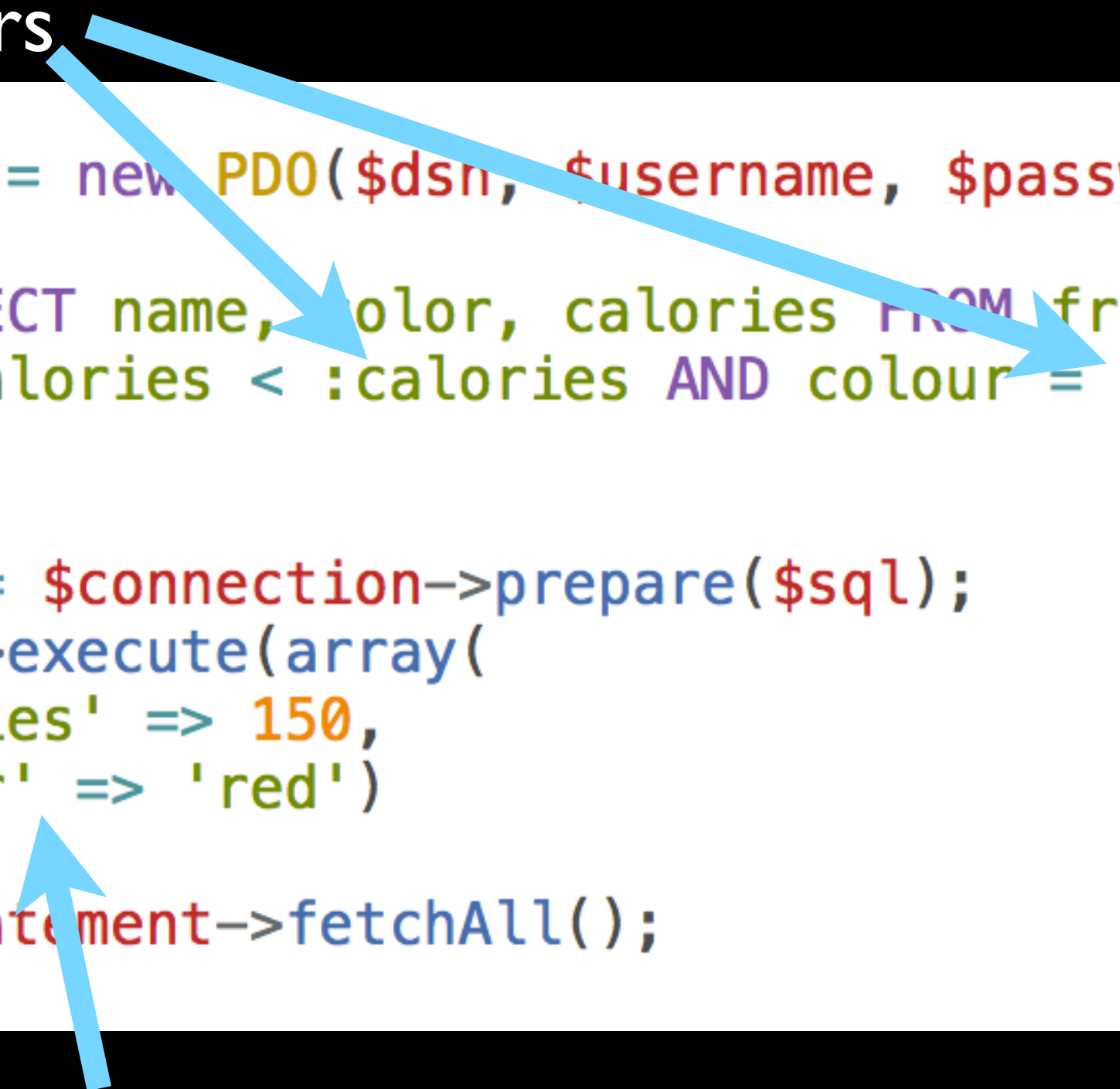
PDO - Safe Queries

Placeholders

```
$connection = new PDO($dsn, $username, $password);

$sql = 'SELECT name, color, calories FROM fruit
      WHERE calories < :calories AND colour = :colour';

$stmt = $connection->prepare($sql);
$stmt->execute(array(
    ':calories' => 150,
    ':colour'   => 'red'
));
$data = $stmt->fetchAll();
```

A diagram with three blue arrows. One arrow points from the word 'Placeholders' to the colon in ':calories' in the SQL query. A second arrow points from the same word to the colon in ':colour' in the same query. A third arrow points from the word 'Map placeholders to values' at the bottom to the array of values in the execute() call, specifically pointing to the '150' value for the ':calories' placeholder.

Map placeholders to values

Closing a PDO Connection

- When using PDO to connect to a database, it's important to disconnect when we're done.
- `$connection = null;`
- This frees up the resource for other scripts to use it.
- If this isn't done, PHP will close the connection when the script is finished.

PDO Error Handling

- Adding a database to your application means you're adding an additional point of failure.
- What happens when the script cannot connect to the database?
- Your application should know how to handle that.

PDO Error Handling

```
try {  
    $connection = new PDO($dsn, $username, $password);  
    $connection->setAttribute(  
        PDO::ATTR_ERRMODE,  
        PDO::ERRMODE_EXCEPTION  
    );  
} catch (PDOException $e) {  
    echo 'Connection failed: ' . $e->getMessage();  
}
```

- Calling the setAttribute method allows you to tell PDO how to handles errors.
- This tells PDO to use PHP Exceptions

PHP Exceptions

- When something goes wrong, an exception is “thrown”.
- Sometimes exceptions can be “caught” if there’s a way to handle the error gracefully and continue to execute the program.
- Sometimes exceptions keep bubbling up, if it’s not caught, and then the program returns a 500 Internal Server Error.

PHP Includes

PHP Includes

- Include one PHP file in another.
- This is good if you have a file full of functions, or a class that you'd like to use in multiple scripts.
- Language constructs: `require_once`, `include_once`, `require`, `include`
- `require_once '/path/to/file.php';`

functions.php

```
<?php

function validateUserName($user) {
    if ctype_alnum($user) {
        return true;
    }
    return false;
}

function validateName($name) {
    if ctype_alpha($name) {
        return true;
    }
    return false;
}
```

form_handle.php

```
<?php

require_once 'functions.php';

if (strtolower($_SERVER['REQUEST_METHOD']) == 'post') {
    if(!validateUserName($_POST['username'])) {
        // username is not valid.
    }
}
```

include, require, require_once, include_once

- require and include doesn't care if a script has already been included, it will include it again. Be careful which one you choose, otherwise it may cause errors.
- require_once and include_once will check to see if the script has already been included, and then ignore it, not include it again.

include, require, require_once, include_once

- include and require are the same except include produces a PHP warning and the script keeps running, whereas require halts the script.
- include_once and require_once are the same except include_once produces a PHP warning and the script keeps running, whereas require_once halts the script.

To the Lab!