CS 547

Week 6 Day 2

Sessions, Cookies, Logins, and Site Structure

Agenda

Administrative announcements

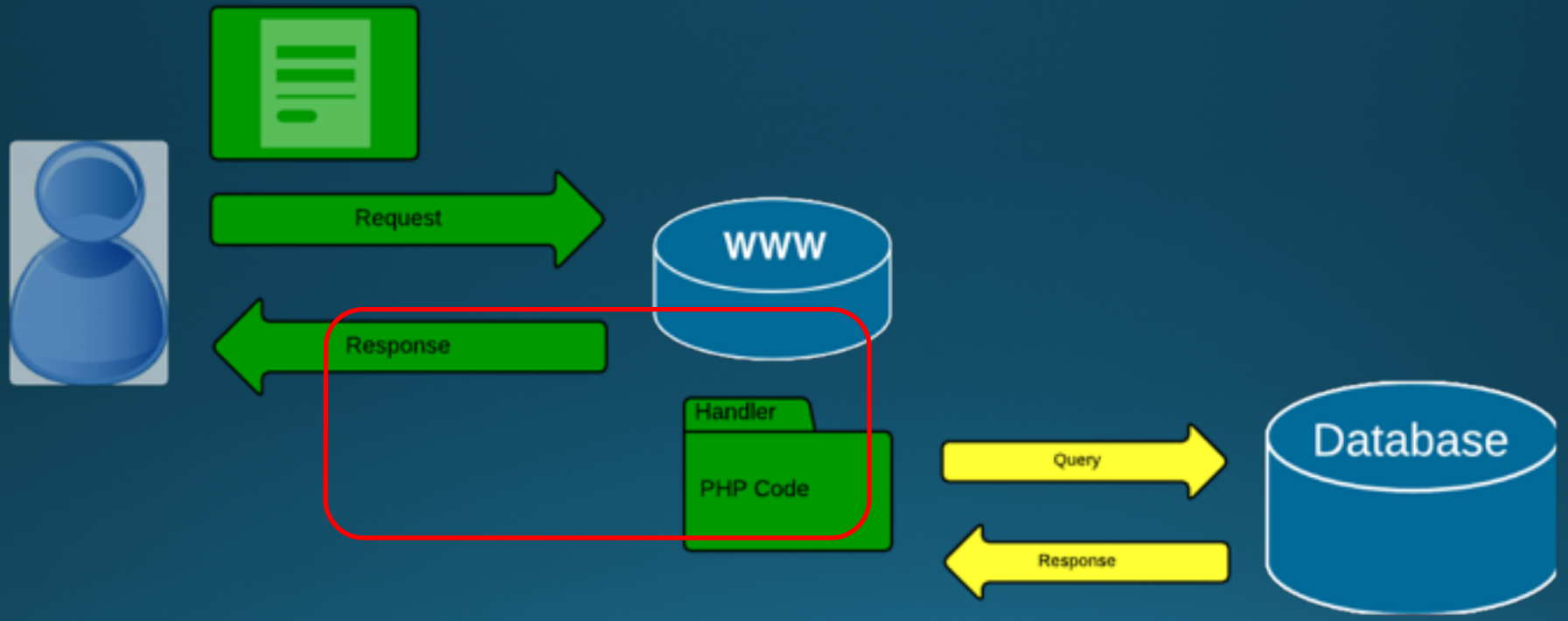
Classes, Objects and OOP

Packages

Administrative

- Next Class March 3, we will review the php section of the course.
- Midterm covering php will be after class.
- No Class Mach 5.
- We will begin Python March 10

Recall our conversation



PHP Objects and Classes

PHP supports both Procedural Code and Object oriented Code.

Which style to use is dependent on the design philosophy of the project.

Procedural

```
$m = mysqli_connect(...);
$res = mysqli_query($m, $query);
$results = array();
while ($row =
mysqli_fetch_assoc($res)) {
    $results[] = $row;
}
```

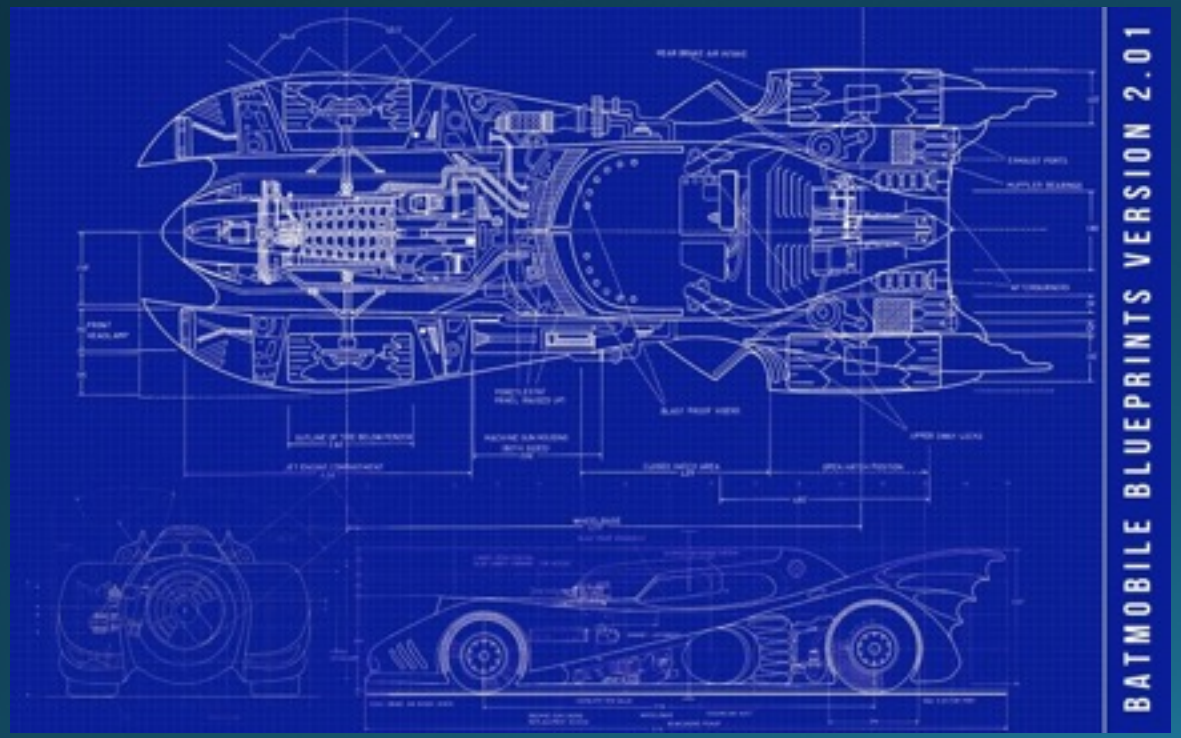
Object

```
$m = new MySQLi(...);
$res = $m->query($query);
$results = array();
while ($row = $m->fetch_assoc($res))
{
    $results[] = $row;
}
```

Objects and Classes

Differences between objects and classes.

Classes are like blueprints



Object are the *actual* things

Objects and Classes

Classes for a structure of both Data and Actions available for use to build an object.

More than one object can be built from a class at the same time, each independent of the others.

Classes structure

Syntax:

```
<?php

class MyClass
{
  // Class properties and methods go here
}

?>
```

Objects

Objects of a class are created with the new keyword.

Syntax:

```
<?php
 class MyClass
{  // Class properties and methods go here}

$object = new MyClass;
var_dump($object);
?>
```

Class Properties

To add data to a class, class specific variables, or properties are used.

```
<?php
 class MyClass
{
      public $carType = "Batmobile";
}
 $object = new MyClass;
var_dump($object);
?>
```

```
object(MyClass)#1 (1) {
  ["carType"]=>
  string(9) "Batmobile"
}
```

Class Visibility

There are three levels of visibility

```
public $public = 'Public';
protected $protected = 'Protected';
private $private = 'Private';
```

The keyword **public protected private or** determines the visibility of the property.

Class Visibility

public: can be accessed everywhere

protected: can be accessed only within the class itself and by inherited and parent classes

private: may only be accessed by the class that defines the member

Class Methods are class specific functions.

```
<?php
class MyClass{
  public $carType = "Batmobile";

  public function setProperty($newval)  {
      $this->carType = $newval;  }

  public function getProperty()  {
      return "I'm a " .$this->carType . "\n";  }
}
$obj = new MyClass;
echo $obj->carType . "\n";
echo $obj->getProperty();
$obj->setProperty("CatMobile"); // Set a new one
echo $obj->getProperty();
?>
```

This code outputs:

```
Batmobile
I'm a Batmobile
I'm a CatMobile
```

Constructors and Destructors

PHP also provides a number of magic methods, or special methods that are called when certain common actions occur within objects. Such as creating or deleting an object

void __construct ([mixed $args = "" [, $...]])

void __destruct (void)

Constructors

```
class MyClass{
  public $carType ;

  function __construct() {
        print "In constructor\n";
        $this->carType = "Batmobile";
   }

$obj = new MyClass;
echo $obj->carType . "\n";
echo $obj->getProperty();
$obj->setProperty("CatMobile now"); // Set a new one
echo $obj->getProperty();
?>
```

Output

```
In constructor
Batmobile
I'm a Batmobile
I'm a CatMobile now
```

Destructors

```
class MyClass{
  public $carType ;

  function __construct() {
        print "In constructor\n";

        $this->carType ="BatMobile";
  }

  function __destruct() {
          print "Batman Forever! \n";
  }
...
$obj = new MyClass();
echo $obj->carType . "\n";
echo $obj->getProperty();
$obj->setProperty("CatMobile now"); // Set a new one
echo $obj->getProperty();
?>
```

```
In constructor
BatMobile
I'm a BatMobile
I'm a CatMobile now
Batman Forever!
```

Printing Object

PHP provides a magic function to print objects:

```
__toString();
```

This helps you print the object without causing a fatal error.

__toString()

```
class MyClass{
  public $carType ;
…

  function __toString() {
        echo "Object toString method:\n";
        return $this->getProperty();
  }…

$obj = new MyClass();
echo $obj->carType . "\n";
echo $obj->getProperty();
$obj->setProperty("CatMobile now");
echo $obj;
?>
```

```
In constructor
BatMobile
I'm a BatMobile
Object toString method:
I'm a CatMobile now
Batman Forever!
```

Inheritance

Classes can inherit the methods and properties of another class using the extends keyword. For instance, to create a second class that extends MyClass and adds a method, you would add the following…

Inheritance

```
class MyClass{
  public $carType ;

  function __construct() {
        print "In "  . __CLASS__ . " constructor\n";
        $this->carType ="BatMobile";
  }

class MyOtherClass extends MyClass{
  public function newMethod()  {
      echo "From a new method in " . __CLASS__ . ".\n";  }
}

// Create a new object
$newobj = new MyOtherClass;
// Output the object as a string
echo $newobj->newMethod();
// Use a method from the parent class
echo $newobj->getProperty();
```

Outputs

```
In MyClass constructor
From a new method in MyOtherClass.
I'm a BatMobile
Batman Forever!
```

Overwriting Inherited Properties and Methods

To change the behavior of an existing property or method in the new class, you can simply overwrite it by declaring it again in the new class.

Overwriting Inherited

```
class MyOtherClass extends MyClass
{

  public function __construct()  {
      echo "A new constructor in " . __CLASS__ . ".<br />";
  }
  public function newMethod()
  {
      echo "From a new method in " . __CLASS__ . ".\n";
  }
}
```

Outputs:
A new constructor in MyOtherClass.
From a new method in MyOtherClass.
I'm a
Batman Forever!

Preserving Original Functionality While Overwriting Methods

To add new functionality to an inherited method while keeping the original method intact, use the parent keyword with the scope resolution operator (::):

```
parent::__construct(); // Call the parent class's constructor
```

Overwriting Inherited

```
class MyOtherClass extends MyClass
{

  public function __construct()  {
      parent::__construct(); // Call the parent class's constructor
      echo "A new constructor in " . __CLASS__ . ".<br />";
  }
  public function newMethod()
  {
      echo "From a new method in " . __CLASS__ . ".\n";
  }
}
```

Outputs:

```
In MyClass constructor
A new constructor in MyOtherClass.
From a new method in MyOtherClass.
I'm a BatMobile
Batman Forever!
```

Static Properties and Methods

A method or property declared **static** can be accessed without first instantiating the class; you simply supply the class name, scope resolution operator, and the property or method name.

```
public static $count = 0;
```

Overwriting Inherited

```
class MyOtherClass extends MyClass
{

  public function __construct()  {
      parent::__construct(); // Call the parent class's constructor
      echo "A new constructor in " . __CLASS__ . ".<br />";
  }
  public function newMethod()
  {
      echo "From a new method in " . __CLASS__ . ".\n";
  }
}
```

Outputs:

```
In MyClass constructor
A new constructor in MyOtherClass.
From a new method in MyOtherClass.
I'm a BatMobile
Batman Forever!
```

PHP Packages

There are a rich variety of add on libraries for php. The two main ones are

1. PEAR

2. PECL (pronounced 'pickle')

PEAR

PEAR - PHP Extension and Application Repository

- PEAR is a framework and distribution system for reusable PHP components.
- Built in to the repository

PECL PHP Extension Community Library

PHP Extension Community Library is conceptually very similar to PEAR, and indeed PECL modules are installed with the PEAR Package Manager.

PECL contains C extensions for compiling into PHP. Needs administrative access

PHP Packages

Over the weekend visit the sites and become familiar with them.

1. PEAR http://pear.php.net/

2. PECL http://pecl.php.net/

Pear Demo