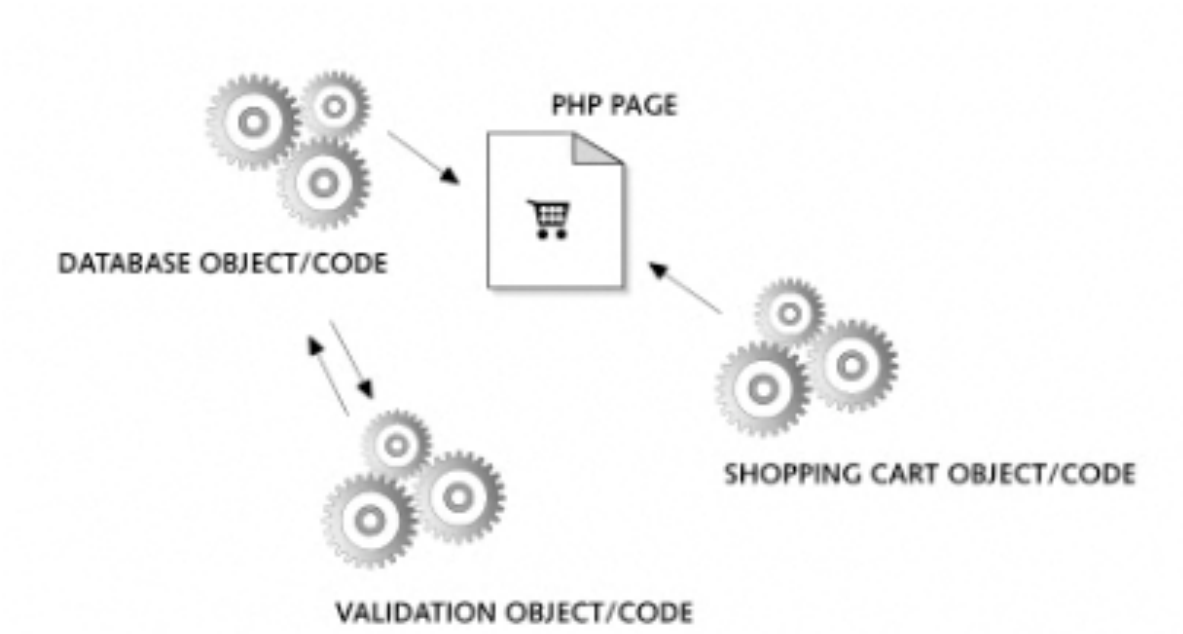
CST8285

PHP Continues
PHP – OOP and Arrays

PHP & OOP (Object Oriented Programming)

- PHP scripts can be both procedural and OOP since PHP 5 OO.
  - Once again this is a vast subject and we will simply be scratching the surface in this program.
  - This subject could be a course all on its own.
- What is object oriented programming?
  - Think of it as having smaller programs inside of a larger program

Simple PHP OOP Diagram



Taken from www.killerphp.com

OOP Uses / Advantages

- What are the advantages or OOP?
  - Allows you to better organize your code
  - Allows you to reuse code
  - Makes updating the application and maintenance much easier and efficient
    Why would we use this type of programming PHP?
    - Especially with larger applications
  - Having a vast collection of code libriaries can increase your ROI
  - Take advantage of existing PHP frameworks

Classes

- To use an object you need to create a class
  - A class is like a blueprint for the object
- The class itself contains all the code (data and functions)
  - Inside a class data/variables are called *properties*
  - Functions inside a class are called *methods*
- *Encapsulation* – this is a method whereby only the methods inside the class can manipulate its properties
  - This makes it easier to maintain the class/code

Declaring a Class

- Define a class using the *class* keyword

```
<?php

  $object = new User;

  class User
  {
    public $name, $password
   // the variable $name and $password are the properties of the class User

    // The save_user  function inside the class User is called a method
    function save_user()
    {
      echo "Save User code goes here";
    }

  }
?>
```

Inheritance – Superclass / Subclass

- Inheritance
  - One class inherits all properties of another class
- At times you may require two classes that are similar but accomplish a different tasks
  - Instead of rewriting code you can simply *inherit* and therefore use the properties of the other class
  - To extend the use of one class to another you use the *extends* operator.
- The original class is called the *superclass* and new class is called the *subclass*.

Creating an Object

- Think of an object as a jukebox
- To create a class you use the keyword *new*
- Example:

```
$object = new User;
```

Referencing the class to use.

Assigned an object named $object to the User class

Accessing Objects

```
$object = new User; // assigned the object

$object->name = "Michael";
//sets the value of the name property in the class User

$object ->password = "pass123";
//sets the value of the password property in the class User

$object->save_user();
// calls the method save_user in the class User
```

- Notice the use of the "->" pointer. The pointer allows access to the classes properties and methods.

Constructors

- Similar to a function call, you can pass a list of arguments to a class being called.
- Arguments are passed to a special method within the class called the constructor.
- The constructor method initializes various properties within the class.

```
Class User
{

 function __constructor($param1,$param2)
   {
     //constructor codes goes here
   }

}
```

Methods

- Functions inside classes are called methods
- Methods are created and called the same way as functions are.
  - Do not start your method names with underscores
  - These are reserved for special methods as constructors and destructors
- In methods you have access to a special variable called *$this*
- The *$this* variable is used to access the current objects properties

Methods Con't

```
<?php

  $mypass = new User;
  echo $mypass->get_password();

  class User
  {
    public $name = "Michael";
    public $password = "mypass";

    function get_password()
    {
      return $this->password;
    }
  }

?>
```

- Observe the get_password method above.
- Using the special $this variable give access to the current object ($mypass) and then returns the value of the password property.
- Notice the $ is missing from the property $password when using the "->" pointer

Declaring Properties

- It is not necessary to declare properties within classes. They can be defined when first used.
  - **However**, not declaring properties is not a good practice. This makes your code within the class very clear.
- You can define Constant Properties using const keyword.

```
const tax = 0.13;
```

- Constant variables can be referenced using the *self* keyword and the double colon operator.

```
echo self::tax;
```

Property and Method Scope

- There are 3 keywords for controlling the scope of both properties and methods.
- public
  - Is the same as var and is the default when declaring properties and methods.
- protected
  - Properties and Methods declared *protected* can be referenced only by the objects class and subclasses
- private
  - Properties and Methods declared *private* can only be referenced by that particular class.

The final word on OOP

- There is so much material to cover we cannot cover it all in one class!
- You will do your own research on the following;
  - The **extends** operator
  - The **parent** operator
  - Investigate the Subclass constructors
  - The **final** keyword
- These will be on the final exam and you will be expected to know what they mean and how they are used.
- I implore you all to take the time to play around as much as possible with OOP and all the keywords and operators discussed in these slides.

PHP Arrays

- PHP arrays are like ordered maps that associates values to keys.
- There are three types of arrays
  1. Numerically indexed arrays
  2. Associative arrays
  3. Multidimensional arrays

Indexed Array

- Numerically Indexed Array

```
<?php
  $paper[]  = “Copier”;
  $paper[] = “Inkjet”;
  $paper[] = “Laser”;
  $paper[] = “Photo”;

  $print_r($paper);
?>
```

Try this example to see what print_r does!

Associative Arrays

- Associative arrays allow you to reference the items in an array by name rather than by number (as per the indexed array)
  - Here is an example of an associative array

```
<?php
  $paper[‘copier’] = “Copier & Multipurpose”;
   $paper[‘inkjet’] = “Inkjet Printer”;
   $paper[‘laser’]   = “Laser Printer”;
   $paper[‘photo’] = “Photo Printer”;
?>
```

- Each item has a unique name that can be used to reference its value.
- The names are called *indexes*
- Research how to create an array using keywords!

Multidimensional Arrays

Example 6-10. Creating a multidimensional associative array

```
<?php
$products = array(
    'paper' =>  array(
        'copier' => "Copier & Multipurpose",
        'inkjet' => "Inkjet Printer",
        'laser'  => "Laser Printer",
        'photo'  => "Photographic Paper"),

    'pens' => array(
        'ball'   => "Ball Point",
        'hilite' => "Highlighters",
        'marker' => "Markers"),

    'misc' => array(
        'tape'   => "Sticky Tape",
        'glue'   => "Adhesives",
        'clips'  => "Paperclips") );

echo "<pre>";
foreach ($products as $section => $items)
    foreach ($items as $key => $value)
        echo "$section:\t$key\t($value)<br>";
echo "</pre>";
?>
```

Multidimensional Arrays

Example 6-11. Creating a multidimensional numeric array

```
<?php
$chessboard = array(
    array('r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'),
    array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
    array('R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R'));

echo "<pre>";
foreach ($chessboard as $row)
{
    foreach ($row as $piece)
        echo "$piece ";
    echo "<br />";
}
echo "</pre>";
?>
```

Accessing Arrays

- There are multiple ways of accessing arrays.
  - for loops (easy for numeric arrays)
  - foreach … as loop
  - The list() and each() functions

Array Functions

- There are a number of array functions that help us manage and manipulate the values inside arrays
- Become familiar with these functions
  - is_array()
    - Determines if a variable is an array
  - count()
    - Determine how many elements are in an array
  - sort()
    - Sorts the values of an array (numeric and string)
  - shuffle()
    - Arranges the values of an array in random order

Array Functions Con't

- explode()
  - Takes a string and places it into an array
- extract()
  - Useful for creating associative arrays from GET and POST arrays
- compact()
  - Creates arrays from variables
- reset()
  - Places the pointer to the first element in an array
- end()
  - Places the pointer to the last element in an array