CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

DIGH 402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2015

Week 5

PHP and MySQL

Week 4 Exercise

PHP class and test script, which is able to produce the following output,

1. output a user's username
2. output a user's firstname and lastname
3. output a user's age and gender

PHP and MySQL

Week 4 Exercise - output a user's username - v1 - Example

```
<?php
class User {

	public $username;

	public function __construct($username) {
	$this->username = $username;
	return true;
	}
}
?>

<?php
require 'user.class.php';

$user = new User('fulcanelli');

echo 'Username = '.$user->username;
?>
```

PHP and MySQL

Week 4 Exercise - output a user's username - v2 - Example

```
<?php
class User {
	//properties for class
	public $name;

	//setter for name
	function set_name($newname) {
		$this->name = $newname;
	}

	//getter for name
	function get_name() {
		return $this->name;
	}
}
?>
```

```
<?php
//require the file
require 'user.class.php';

//instantiate object of User class
$user = new User();

//set name
$user->set_name("ancientlives");
//get name
$newname = $user->get_name();
//output name
echo 'new name = '.$newname;

?>
```

PHP and MySQL

Week 4 Exercise - output a user's firstname and lastname -

- how to solve this issue in the user class?
  - variant solutions
- how to output from index.php?
- any error checking?

PHP and MySQL

Week 4 Exercise - output a user's age and gender - Example

```
<?php
class User
{
	//properties for class
	private $age;
	private $gender;
	public $name;
	//constructor
	function __construct($fullname) {
		$this->name = $fullname;
	}
	//set age
	function set_age($newage) {
		$this->age = $newage;
	}
	//get age
	function get_age() {
		return $this->age;
	}
	//set gender
	function set_gender($newgender) {
		$this->gender = $newgender;
	}
	//get gender
	function get_gender() {
		return $this->gender;
	}
}
?>
```

```
<?php

//require the file
require 'user.class.php';

//instantiate object of User class
$user = new User("ancientlives");

//get and output age
echo '<p>name = '.$user->name.'</p>';
//set age
$user->set_age(20);
//get and output age
echo '<p>age = '.$user->get_age().'</p>';
//set gender
$user->set_gender("male");
//get and output gender
echo '<p>gender = '.$user->get_gender().'</p>';

?>
```

Object Oriented Programming

Intro to Object Oriented Programming

That’s enough for now - let’s consider OOP code for our framework

Object Oriented Programming

Object Oriented Programming - How to convert our code to OOP

- abstract overview of structure

- classes and inheritance

- what is public, private, protected?

- examples and how it works...

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline

- index.php file (loaded by the web server upon initially opening the home page)

- framework application directory (/frame)
    - contains a framework ‘bootstrap’ file
    - will contain directories for files to handle
        - model
        - view
        - controller

- configuration directory (/config)
    - config settings for framework
    - any necessary global settings
    - directory constants and settings

- system directory (/system)
    - constants directory
    - library directory

- assets and template (/design) will be added later

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - index.php file

- we'll use this initial file to setup our framework

- we'll 'require' the frameworks directories, set default paths, set constants

- then load our framework bootstrap loader

    - instantiate an object for the loader class
    - load framework settings
    - initialise the database connection, settings...
    - initialise required sessions for the framework
    - initialise set view theme for the framework
    - initialise the required assets including css, javascript…
    - load controller for the framework
    - load view menus, title…
    - finally render and create the required view for the user
    - etc….

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - /config/directory.php file

- set base framework directory using ‘get current working directory’
    - getcwd()
- set base folder for framework
    - set base directory for framework reference...
- set base folders for framework
    - config
    - design
    - frame
    - system

- set framework system folders

- set framework design folders

- set framework MVC folders

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - /frame/bootstrap.php file

- stored in the framework (/frame) folder

- called once from the index.php file in the root public directory for our framework and site

- initialises our framework and allows us to control loading of parts of our framework
    - main loader file for framework (we'll go through next…)
    - initialise settings
    - initialise database
    - initialise session (covered later on…)
    - initialise our selected theme for the framework design (later...)
    - load framework aspects including menus etc…(later...)
    - assign required variables for layout etc… (later…)
    - load and output the required layout for our framework (later…)

…

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic /system/library/loader.php file

- require our functions.php file for various generic framework functions (empty at the moment…)

- require our constants.php file for framework constants

- require our error_functions.php file for abstracted error handling for framework (empty…)

- require our controller.php file from our system/library/ to allow loading of our MVC (empty…)

- 'Loader' class to allow us to initialise and load various functions and framework requirements

eg:

- initialise settings function loaded from bootstrap.php file
- initialise the database settings and class allowing us to connect to our MySQL database

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic /config/settings.php file

- for now we are adding a few basic global settings for our framework

eg:
- setting the title for the framework
- setting the project director…

- we can also set project metadata for the HTML etc

- keywords, charset, description…

- plus further framework settings such as

- default language
- get base URL for project framework

GitHub Code

Object Oriented Programming

Refresher - Using Static Properties and Methods

- we can define class properties and methods that are ‘static’
    - a static method or property can be used without instantiating the object first

- mark as static by putting the ‘static’ keyword after ‘public’ etc

- ‘scope resolution operator’ :: is used to access ‘static’ properties or methods
    - eg: $user = User::get_instance();

- ‘static’ property is a variable that belongs to the class only, not any object
    - isolated from any other property, even another of the same name in an object of this class

- ‘static’ method has no need to access any other part of the class
    - you can’t refer to $this inside a static method (because no object has been created to refer to)

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic /system/library/db.php and /config/config_db files

db.php (Part 1)
- database class for connection and management using PHP's PDO (PHP data objects) extension. Class contains the following
    - declare various static protected variables
    - setup function for connecting to the database
    - initialise function called to connect to the database within our framework
        - this is called during the bootstrap via the loader class
    - general query method & get row method... GitHub Code

config_db.php
- create a multi-dimensional array to store connection settings for our framework database
- two arrays including one for development settings and another for production settings
eg:
    - hostname, username, password, database GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic db.php file

db.php (Part 2) - why use PDO instead of mysqli

- more modern extension for connecting to databases through PHP
- PDO has a better interface compared to mysql and mysqli
- PDO has different drivers for different SQL database vendors
- instead of concatenating escaped strings into SQL, PDO binds parameters
    - this is a cleaner and easier way of securing queries
        - lack of exposure...
    - allows for performance increase when calling same SQL query with slightly different parameters
- multiple methods for error handling
    - object oriented exception handling
    - consistent style of error handling using PDO

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic db.php file

db.php (Part 3) - querying the database

- multiple options in PDO for returning result dataset from database
    - use a foreach loop
    - or a while loop
    - or one of the available PDO fetch modes

- PDO also has many built-in options to help fetch results
    - simple fetch()
    - fetchAll() returns an associative array with the field names as keys
    - count rows from query dataset using rowCount()

- we can also use PDO to insert, update or delete records in our database

- it's easy to use PDO statements with parameters

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

v0.1

GitHub Setup & Usage

General Setup - Part 1

- Setup Guide

- Create a new repository on GitHub

- GitHub on Mac

GitHub Setup & Usage

General Setup - Part 2

- register for an account on GitHub

- install Git on your local machine

- configure git on local machine using the terminal (command line)

```
git config --global user.name “Your name here”
(**change to your preferred name for GitHub**)
git config --global user.email “your_email@youremail.com”
(**add the same email used to register at GitHub**)
```

- create a repository on GitHub (use an obvious project name)

- create a local repository for your Git projects

```
mkdir ~/MyProject
(**use the same project name as the above new GitHub repository**)
cd ~/MyProject
(**change to the new directory**)
```

GitHub Setup & Usage

General Setup - Part 3

- in the same directory, MyProject, create a Readme file

```
touch Readme.txt
(**basically creates a blank file called Readme.txt - you can also use markdown etc**)
git init
(**this tells Git to recognise this directory as a local Git repository**)
git status
(**shows status at current master branch - ‘untracked’ = Git currently ignoring file(s)**)
git add Readme.txt
(**added file**)
git commit -m “Add Readme.txt”
(**commit files in directory with -m associated commit comment**)
```

- connect local repository to GitHub repository

```
git remote add origin https://github.com/username/myproject.git
(**add a new place where files originate - remote indicates origin**)
git remote -v
(**shows all remote origins for your current repository**)
```

GitHub Setup & Usage

General Setup - Part 4

- push initial changes to remote repository

```
git push -u origin master
(**push changes for master branch of repository**)
```

- push changes made to files in local repository to remote repo

```
git add Readme.txt
(**propose a single file change**)
git add *
(**propose all files in the current directory for change**)
git commit -m “Update Readme.txt”
(**commit file/file for local repo**)
git push origin master
(**push commit files to master branch of remote repo**)
```

GitHub Setup & Usage

General Setup - Part 5

- clone a remote repository

    - cd to local directory for storing remote files eg: ~/github

    git clone https://github.com/ancientlives/digh402.git
    (**downloads a complete copy of the remote repository and sets it up as a local directory ready for use with Git**)

- check and update local repository to match cloned remote repository

    - cd to local directory for repository eg: ~/github/digh402

    git pull
    (**downloads changes from remote repository and merges with local directory**)

Further information