CENTER FOR TEXTUAL STUDIES AND DIGITAL HUMANITIES

DIGH 402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2015

Week 4

PHP and MySQL

Week 3 Exercise

- any questions?
- any issues with querying the MySQL data?

PHP and MySQL

Week 3 Exercise - step by step

- start and load Apache2 and MySQL (using XAMPP etc…)
- open phpMyAdmin in browser at http://localhost/phpmyadmin/ (if using local hosted option)
  - create new database ‘402framework’ with collation set to ‘utf8_unicode_ci’
  - open ‘402framework’ database and set user privileges for 402user and 402admin
    - 402user = ‘select’ privileges & 402admin = ‘all privileges’ on ‘402framework’ DB
  - import sample DB ‘402week3.sql’ (ensure character set is utf-8 & other defaults are OK)
  - add some test data to the ‘content’ and ‘content_lookup’ tables - examples
- save test code, eg: ‘basicInclude3’ to your working test directory for Apache
  - eg: XAMPP/htdocs/testing/basicInclude3/
    - NB: XAMPP/htdocs/ is the root directory for Apache in XAMPP
      - in your browser = http://localhost/
      - therefore, XAMPP/htdocs/testing/basicInclude3/ = http://localhost/testing/basicInclude3/
    - update username and password for DB connection in your code
      - eg: modify basicInclude3/includes/config.inc.php
        - set the constant ‘DB_USER_QUERY’ to ‘402user’
        - set the constant ‘DB_PASS_QUERY’ to ‘mypassword’
        - set the constant ‘DB_DATABASE’ to ‘402framework’ - example
    - test your code and database
      - eg: at http://localhost/testing/basicInclude3/ - example

PHP and MySQL

Week 3 Exercise

- how the tables work and fit together… - example
- how the content_lookup table works…
  - currently three columns
    - content_id, content_type_id, & user_id
    - content_id needs to match a given record in the content table
    - content_type_id needs to match a given record in the content_type table
    - user_id needs to match a given record in the users table
  - examples records might be

content_id	content_type_id	user_id
1	1	1
2	1	2
3	2	1

PHP and MySQL

Week 3 Exercise

content_id	content_type_id	user_id
1	1	1
2	1	2
3	2	1

- some queries we can now generate from this single table?

```
SELECT content.contentname, content_type.content_type_name
FROM content_lookup
JOIN content ON content.contentid=content_lookup.content_id
JOIN content_type ON content_type.content_type_id=content_lookup.content_type_id

SELECT content.contentname, content_type.content_type_name
FROM content, content_type, content_lookup
WHERE content_lookup.content_id=content.contentid
AND content_lookup.content_type_id=content_type.content_type_id
```

Object Oriented Programming

Why Object Oriented Programming?

- it is possible to write complex and useful sites using procedural outline, functions…

- true value of OOP is in a concept known as ‘encapsulation’
    - this allows us to associate values and functions together in one unit : the **object**

- objects allow us to collect values together, add functionality to the unit…
    - easy to add and remove functionality

Basic Vocab

- class = outline / blueprint / design for creating a given object

- object = something that encapsulates the design etc of the class…

- method = similar to a function that belongs to an object (ie: called by name associated with an object)

- property = a variable that belongs to an object

Object Oriented Programming

Intro to Object Oriented Programming

- a given ‘class’ is a blueprint, a set of instructions for how to create our object
    - effectively, it describes an object

- classes can represent all sorts of entities within our application, for example components within our framework

a simple class

- shows the class declaration, and we’ll save as user.class.php

- we have one property, $username

- two methods, __construct() and name()
    - each method receives a parameter, $username and $name
    - we ‘return’ a value from each method

- $this is a special variable that is always available within an object’s scope
    - it refers to the current object

```
class User {

        public $username;

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

        public function name($name) {
        //return the user’s name
        return true;
        }

      }
```

Object Oriented Programming

Intro to Object Oriented Programming - Class Constructors

- the __construct() method has two underscores
    - it's a special method
    - it's called when we instantiate an object
    - it's known as the 'constructor'

- the constructor is always called when we instantiate an object

- used to set up and configure the object before it's returned for use in the main code

- a class does not have to include a constructor

```
class User {

    public $username;

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

    public function name($name) {
    //return the user's name
    return true;
    }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Instantiating an Object

- to instantiate, or create, an object
    - use the ‘new’ keyword
    - specify the name of the class for the given object
    - pass any required parameters expected by the constructor (if included in the class)

eg:
```
require ‘user.class.php’;
$user = new User(‘fulcanelli’);
```

- first, we require the file that contains the class definition

- then we instantiate a new ‘User’ object passing the name parameter the constructor expects

- we store the result in an object called ‘$user’

```
class User {

    public $username;

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

    public function name($name) {
    //return the user’s name
    return true;
    }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Instantiating an Object

Example output for class User

- we can see that it is an object of class 'User'

- there is currently one property

- we can see the name and value of each property

EXAMPLE OUTPUT

```
class User {

        public $username;

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

        public function name($name) {
        //return the user's name
        return true;
        }

}

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

$user = new User('fulcanelli');

var_dump($user);

?>
```

Object Oriented Programming

Intro to Object Oriented Programming - Using Objects

- we’ve instantiated an object

- we need to access a property, call a method…eg:

```
//require
require ‘user.class.php’;
//instantiate an object
$user = new User(‘fulcanelli’);

//access a property
echo ‘Username = ‘.$user->username;

//call a method
$user->name($name);
```

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

$user = new User('fulcanelli');

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

?>
```

EXAMPLE OUTPUT

- object operator = ->
    - goes between the object and the property or method you want to access or call

Object Oriented Programming

Intro to Object Oriented Programming - Visibility in a class

- visibility of a property or method can be set using keywords ‘public’, ‘protected’, ‘private’

- ‘public’ can be accessed everywhere

- ‘protected’ can be accessed only within the class itself and by inheritance from the parent

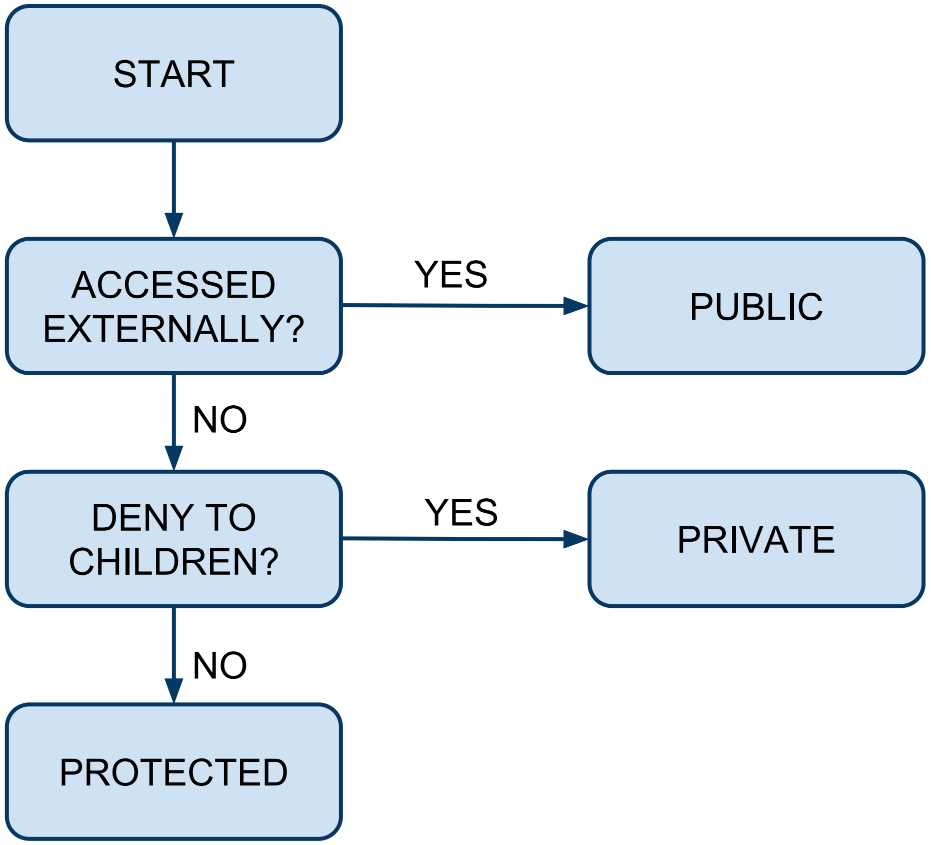
- ‘private’ can only be accessed by the class itself

eg:

```
class {
	public $ver1 = ‘ver 1’;
	protected $ver2 = ‘ver 2’;
	private $ver3 = ‘ver 3’;
}
```

Object Oriented Programming

Intro to Object Oriented Programming - Choosing the Correct Visibility



Object Oriented Programming

Intro to Object Oriented Programming - 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)

- NB: often used in PHP libraries, references etc where the functionality is independent of any object properties, and mimics the older procedural call to a global function...

Object Oriented Programming

Intro to Object Oriented Programming - Object Inheritance

- ‘Inheritance’ is effectively how classes relate to each other
    - a class can inherit from another class
    - parent to child…

- classes can inherit or extend a parent class

- classes are unaware of other classes inheriting from them
    - therefore, no limits on how many child classes per parent

- child class has the parent’s characteristics
    - we can add or change any elements that need to be different for the child
    - not parent’s private scope

- using ‘User’ class as an example we can create child classes
    - user type etc
    - registration
    - login
    - user details...

Object Oriented Programming

Intro to Object Oriented Programming - Type Hinting in PHP

- type hinting
    - allows methods to only accept parameters from objects of a specified class

eg:
```
public function delete(User $user) {
//delete the user from the DB...
return true;
}
```

- you can type hint any object name
    - PHP is a dynamic and weakly typed language
    - no type hinting for simple types such as strings, number types etc

- type hinting allows us to be sure about the type of object passed to a given method/function

(Method is similar to a function used in the context of a class/object - an object can have methods and properties...)

- this allows us to make assumptions in our code
    - eg: properties and methods available as a result

Object Oriented Programming

Intro to Object Oriented Programming - Objects and References in PHP

```
$user1 = $user2
```

- variable $user1 will contain the same value as $user2
    - we end up with two independent variables, but same value

- objects work in a different way

```
$user1 = new User();
$user1->username = ‘fulcanelli’;
$user2 = $user1;
$user2->username = ‘amelie’;
```

- $user2 is not a copy of $user1, but another reference to the same object for the class User
    - this is called a ‘reference’
    - also allows access to class’ ‘private’ methods etc...

Object Oriented Programming

Intro to Object Oriented Programming - Objects and References in PHP (cont'd)

- objects are always passed by reference
    - when you pass an object into a function/method, the function/method operates on the same object
    - any change inside the function/method is reflected outside the function/method
    - objects provide a reference to the original object rather than produce a copy

- this means that if a function/method operates on an object passed in
    - there's no need to return the object from the function/method
    - any change will be reflected in the original object

- 'clone' keyword allows us to create a separate copy of an object
    - cloned object will have same properties etc as original object
    - changes made to cloned object will not change original object
    - original object will maintain its own values etc...

Object Oriented Programming

Intro to Object Oriented Programming - Getters and Setters in PHP

- public, protected, or private to control visibility for a property or method

- another option is to mark all properties as protected
  - we can then use ‘getter’ and ‘setter’ method to access them
  - they basically allows us to ‘get’ and ‘set’ the values

eg:

```
class User {
	protected $username;

	function getName() {
	return $this->username;
	}

	function setName($value) {
	$this->username = $value;
	return true;
	}
}
```

Object Oriented Programming

Intro to Object Oriented Programming - Getters and Setters in PHP (cont'd)

- very useful for tracing object code that accesses properties

- the getter and setter methods offer an access point each time we need a property
    - this provides a 'hook' or 'intercept' point
    - we might use these methods to log what information was updated
    - or perhaps to add some access control logic…

- often a personal choice whether to use 'getters' and 'setters' or access properties directly

Object Oriented Programming

Intro to Object Oriented Programming - Exceptions

- an object oriented approach to handling errors

- exceptions themselves are ‘objects’ and ‘Exception’ is a built-in class in PHP

- an ‘exception’ object contains information about
    - where the error occurred (filename and line number)
    - an error message
    - and optionally an error code

- exceptions are considered a more elegant way of handling errors

- they allow us to react to exceptions in the course of execution, dependent upon the severity of the problem

- we can assess the code situation and then react by either recovering or bailing out gracefully

Object Oriented Programming

Intro to Object Oriented Programming - Exceptions (cont’d)

- we can also extend exception objects, customise their data and behaviour…

eg: Try/Catch block

```
try {
	$db = new PDO(‘mysql:host=db’);
	echo “Connected to Database”;
} catch (Exception $e) {
	echo ‘Oh no! ‘.$e->getMessage();
}
```

- we can also throw our own exceptions

eg:

```
throw new Exception (‘a useful error message string!’);
```

Build your own Class

- PHP class and test script to produce the following output
    - output a user’s username
    - output a user’s firstname and lastname
    - output a user’s age and gender

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