LOYOLA UNIVERSITY CHICAGO | CENTER FOR TEXTUAL STUDIES AND DIGITAL HUMANITIES

DIGH 402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2015

Week 6

PHP and MySQL

Week 5

1. GitHub - 402framework - v0.1

GitHub - Fork 402framework repo

Fork 402framework repo

- Repo updated on a weekly basis with the current stable build
  - 402framework repository
- Simply update your working directory to reflect weekly updates...
- weekly reference files will be artificially versioned in source repository

GitHub - Fork 402framework repo

GitHub Docs - Fork a Repo

- Fork 402framework repo on GitHub website to your own account
- clone 402framework fork from your remote account to a local copy
- configure Git to sync with the original 402framework repo on dighteach (otherwise sync with just forked repo on personal account)
    - cd to local cloned directory
    - ***git remote -v***
    - ***git remote add upstream https://github.com/dighteach/402framework.git***
    - then verify the new upstream repository
        - ***git remote -v***

GitHub - Fork 402framework repo

GitHub Docs - Syncing a Fork

- cd to local directory for 402framework
- fetch the branches and commits from the remote repository (upstream = original project location)
    - ***git fetch upstream***
    - this creates a new local branch -> upstream/master
- merge the changes from ‘upstream/master’ into local ‘master’ branch
    - this brings the local ‘master’ branch into sync with the upstream repository
    - ***git merge upstream/master***
- to ensure an up-to-date remote copy on your own forked repository issue a standard push request
- ***git branch*** (shows current local working branch)
- further information on resolving merge conflicts and GitHub collaboration

GitHub - Fork 402framework repo

[GitHub Docs - Syncing a Fork]

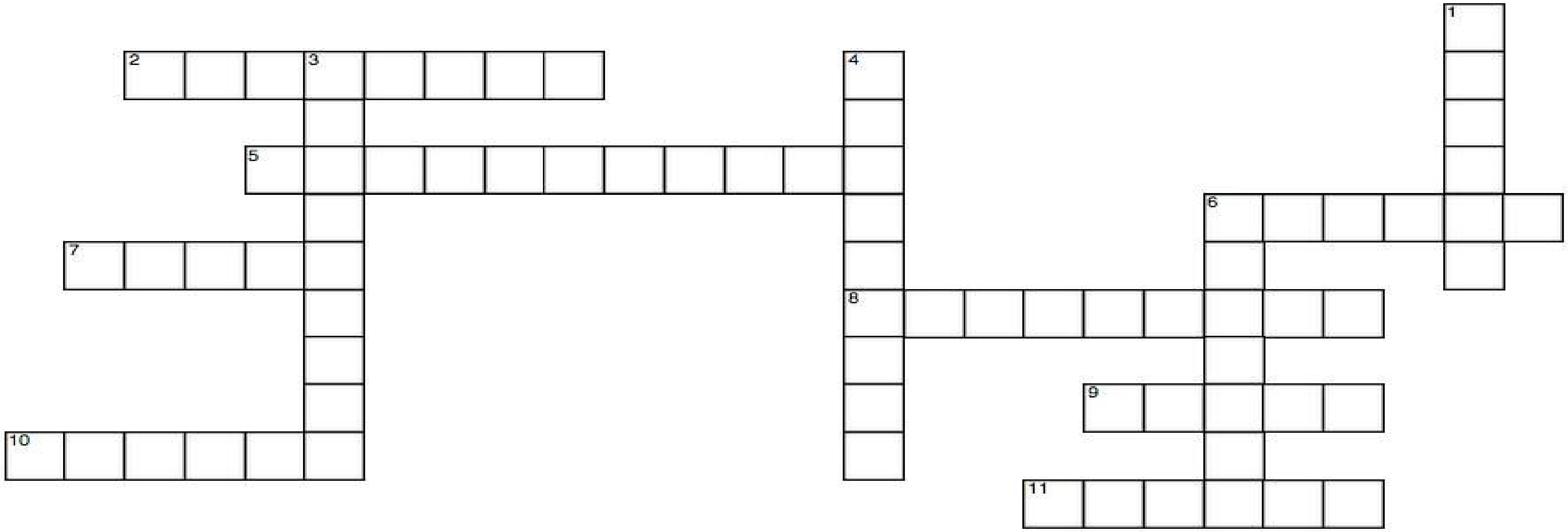
Sync Scenarios

- make local change and fetch and merge from upstream
    - no upstream changes to file, local changes persist
- make changes to original remote repo & not local
    - upstream merge will change local
- change the same file in both remote and local and try to merge
    - conflict between both files will need to be resolved…

NB: ***git status***

OOP Crossword

Quick test



ACROSS
2 a variable that belongs to an object
5 if present will always be called when we instantiate an object
6 can be accessed everywhere
7 allows us to create a separate copy of an object
8 an object oriented approach to handling an error
9 outline, blueprint, design for creating a given object
10 a function that belongs to an object
11 something that encapsulates the design etc of the class

DOWN
1 can be used without instantiating the object first
3 can be accessed only within the class itself and by inheritance from the parent
4 variable passed to a method as an argument
6 can only be accessed by the class itself

OOP Crossword

Across

2 = Property (a variable that belongs to an object)
5 = Constructor (if present will always be called when we instantiate an object)
6 = Public (can be accessed everywhere)
7 = Clone (allows us to create a separate copy of an object)
8 = Exception (an object oriented approach to handling an error)
9 = Class (outline, blueprint, design for creating a given object)
10 = Method (a function that belongs to an object)
11 = Object (something that encapsulates the design etc of the class)

Down

1 = Static (can be used without instantiating the object first)
3 = Protected (can be accessed only within the class itself and by inheritance from the parent)
4 = Parameter (variable passed to a method as an argument)
6 = Private (can only be accessed by the class itself)

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

A Quick Initial Outline - basic sessions

- what is a ‘session’ in PHP?

    - use (and sometimes abuse) of session variables
    - store information about a user session
    - change settings specific to a given user
    - session variables available to all pages of a framework/site
    - session data is temporary and can be deleted after the user exits the framework/site
    - commonly stored in a cookie

- what can we do with session variables?

    - store usernames, user details…
    - track page visits within our framework/site
    - keep temporary data including session preferences, selections…

- difference between cookies and session variable?

W3Schools Overview

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - /system/library/loader.php file (Part 2)

- we now need to be able to load some actual content

- we now introduce two more methods
    - auto_load_controller
    - load_controller

- auto_load_controller is called from the bootstrap.php file and works with router.php and the load_controller method

- load_controller actually loads and returns the user selected content within the framework
    eg: v.0.2

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - /system/library/auto_load_controller method

- requires (includes) our router.php file to help process the user submitted URI
    - we'll go through router.php next...
- instantiates an object from the Router class

- sets various variables we need for the load_controller method using getters in the Router class

eg:

    ?node=content/text&id=2

    - controller = 'content'
    - controller_dir = 'frame/controller'
    - format = 'text'
    - params = 'id=2'

- then calls the load_controller method

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - /system/library/router.php and Router class

- sets the route for the framework based upon user requested URI

- numerous private static properties

- constructor to check if $route is already set and if not calls method init()

- five getter methods to allow auto_load_controller() to get required variables

- private init() method to process and return user requested URI
    - get user requested route
    - define route without parameters in case we need base URI request
    - define controller variable
    - define controller directory
    - define format
    - define parameters

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - load_controller method (/system/library/loader.php)

- method called from auto_load_controller method

- determine required controller and load controller

- define class name for required controller

- check if controller class exists and instantiate object

- check content format requested by user and any parameters for the controller

- call get_content() method using controller object

- output basic requested content

GitHub Code (loader.php) | Github Code (content.php)

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

v0.2

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - ContentController class

- set private static property for default content

- getter method to return selected basic content

- very basic test of content query from DB using ‘id’ parameter

- set value for default content property which can be used in Loader class to output content

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - basic constants.php file

- required (included) as part of the loader.php file

- we now have a couple of general constants for use within our framework
    - FRAME_EXTENSION to allow us to specify '.php' for file endings
    - CONTROLLER_CLASS_NAME to allow us to specify that all controller class must follow the same naming pattern ie: ContentController etc

GitHub Code

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - loading the content so far (Part 1)

- user opens framework and requests a URI, which loads the following
    - index.php
        - config/directory.php
        - frame/bootstrap.php

- bootstrap.php requires loader.php file, and instantiates a loader object, which is used to call
    - init_settings() method
    - init_db() method
    - auto_load_controller() method

- auto_load_controller() method requires router.php, instantiates a router object, and calls the getter methods to output required variables for load_controller() method

cont'd...

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - loading the content so far (Part 2)

- load_controller() method uses the supplied parameters for controller, controller_dir, format, params to
    - determine the required controller (eg:content)
    - load the required controller class (eg:ContentController)
    - check and instantiate an object for required controller class
    - check content format and parameters for controller
    - use getter method in controller class to request and output basic returned content
        - raw content is returned based upon ID, but format will be used later to call viewer

Example basic content output

Object Oriented Programming

Object Oriented Programming - Abstract overview of current framework structure

Initial Outline - next on our to-do list

- send content to view before returning for output
    - allows us to introduce view plugins, viewers etc

- add some templating and design for our framework

- allow for greater flexibility and abstraction of content, format, and parameters

- add some more error checking and reporting…

- add default content handler for non controller/format/params URI requests including index.php

and so on...