CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2016

Week 5

PHP and MySQL

Week 4 Exercise

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

PHP and MySQL

Week 4 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 4 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 4 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
```

Week 5 Extra Reading - Part 2

Object Oriented Programming

- Notes as extra reading
- Exercise at the end of the notes
  - create a PHP class and test script
  - output the following
    - output a user's username
    - output a user's firstname and lastname
    - output a user's age and gender

PHP and MySQL

Error Handling - Basic (PHP Manual - Error Handling)

- What is error handling in programming?

    - reporting to help developers…
    - error feedback for users...

- How can we handle errors in PHP?

    - die() function or exit() function - PHP Manual
    - stops script running at point of error
    - custom error handler (see next slide)

- Graceful errors?

    - eg: check user input and logically respond to error
    - trigger_error("Age must be greater than 21!") - PHP Manual
    - exceptions (throw, try, catch) - PHP Manual
        - try, throw, catch...

- Recording errors

    - saved to error_log on server
    - custom error_log() to send email with error number, error string

PHP and MySQL

Error Handling - Custom Error Handler

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] - $errstr";
  }

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>

//sample output for above example
Error: [8] - Undefined variable: test
```

Example from: W3 Schools Overview

PHP and MySQL

TO DO - Basic Error Checking

- check for empty results
- handle errors and return feedback for the user

Handle user error reporting

- basic empty link errors
- empty result set or single empty result
- empty or invalid data returned per DB table row

…

PHP and MySQL

Basic Error Checking - ~ 10 minutes max

Check over http://students.ctsdh.luc.edu/teaching/demos/mysql/basicInclude/

- work your way through the site and identify potential points of error in the logic and flow

Updated with basic error checking:

http://students.ctsdh.luc.edu/teaching/demos/mysql/basicInclude2/

PHP and MySQL

Basic Error Checking - BasicInclude2

- results.php - handle empty ‘req’ from URL - Example

- mysql_connect.inc.php - handle empty ‘results’ dataset returned from DB - Example

- content_viewer.php - Example

- user_viewer.php - Example

Why do we not handle errors in the following *include* files? (For this initial stage of testing...)

- content_processor.inc.php (GitHub)
- results_format.inc.php (GitHub)

PHP Example - Errors | PHP Code Example (‘basicInclude2’ directory in GitHub repository)

PHP and MySQL

Further abstraction in the current code - BasicInclude3

- root.inc.php

    - per required directory to allow specification of root directories

- default_includes.inc.php

    - allows us to store all cross-framework links to include files in one single file
    - eg: MySQL connection & query,

- config.inc.php

    - define ‘assets’ directory for css & javascript files
    - define ‘media’ directory for images, video, audio…
    - define MySQL DB settings, tables…

PHP and MySQL

Further abstraction in the current code

- root.inc.php

  - allows abstracted specification of root directories
  - can be project root directory or per required directory or often both!
  - mainly used to prevent unwanted repetition of directory location in ‘include’ statements
  - eg: location of default includes directory, modules…

Code Example

```
<?php
$root_base = ‘modules/base/’;
$root_content = ‘modules/content/’;
$root_user = ‘modules/users/’;
$root_images = ‘media/images/’;
$root_includes = ‘includes/’;
?>
```

PHP and MySQL

Further abstraction in the current code

- default_includes.inc.php

    - allows us to store all cross-framework links to include files in one single file
    - eg: MySQL connection & query

Code Example

```
<?php
/*DB config etc*/
include($root_includes.'config.inc.php');
include($root_includes.'mysql_connect.inc.php');
?>
```

PHP and MySQL

Further abstraction in the current code

- config.inc.php

    - allows us to store all cross-framework links to include files in one single file
    - eg: MySQL connection & query, DB tables, template settings…

Code Example

```
<?php
//database server
define('DB_SERVER', 'localhost');
//database query user login name
define('DB_USER', '402user');
?>
```

- code and files now need to be updated to reflect this latest abstraction of code and settings

PHP and MySQL

Again, more abstraction - Updated Framework layout and Model

- default framework design components
    - header, sidebar, main content, footer...
- header abstracted to modules/template in framework

PHP Example | PHP Code Example ('basicInclude3' directory in GitHub repository)

...and more to abstract and update.

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