How to test and debug a PHP project

In particular, proj2
PHP with echo statements that trace the execution of the code

// calculate the future value
$future_value = $investment;
echo '$future_value: ' . $future_value . '<br>';
echo '$interest_rate: ' . $interest_rate . '<br>,'
$years: ' . $years . '<br>','
$interest_rate * $future_value);
$future_value = ($future_value + ($future_value * $interest_rate));
echo '$i: ' . $i . '<br>','
$future_value: ' . $future_value . '<br>','

}
The data displayed in a browser
When echo can’t be used

- Echo is great for simple debugging of client-side code, in cases where the controller includes a view file.
- If the controller does a redirect, the echoed text is lost.
- Demo: using echo in pizza2’s day controller, in change-to-next day code that needs work for proj2.
  - Put `echo 'in list handling'` in controller code that forwards to `day_list.php`, see echo in output.
  - Put `echo 'in init-db handling'` in code that redirects to ., see echo output and also an orange box for an error.
    - This is because a redirect has no HTML, only a response code and headers, so having the echo output already generated for the response is an error.
    - This is new behavior for PHP: older PHP just drops the echo output.
  - Note that the next-day handling here does not redirect, unlike the pizza1 solution.
- Also, can’t use echo in `rest/index.php`, because it will be sent back to client and mess up that response.
Errors from using echo before redirect: fixes

• One workaround: use error_log instead of echo.
• Another way: temporarily replace REDIRECT with simple exit.
• Or recode a little and include day_list.php
• Demo: with redirect → exit, now we see just the echo output

in init-db handling

Luckily, plain text works as HTML! (our browsers are very forgiving)
REDIRECT can also hide error displays

- Just like echo output, the orange box output for PHP errors is discarded on REDIRECT.
- Of course, the PHP error log report is there.
- So check the error log whenever something goes wrong.
Error logs

• Check the error log whenever something goes wrong.
• Proj2: pizza2’s error log is in a new place, in pizza2/php-errors.log (both under XAMPP and on topcat)
• Proj2/proj2_server’s error log for web services is in proj2_server/php-server-errors.log
• On topcat (Linux) or home Mac, need to “chmod 777 *.log” in both pizza2 and proj2_server to allow writes to the logs.
  – (Not needed on Windows)
Error Configuration

• XAMPP as installed is set up for development, not production use, so displays most errors.
• php.ini has error configuration for the server, so we could convert the server to production use by editing it.
• From php.ini of XAMPP:
  
  ```
  error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
  display_errors = On
  ```

• From php.ini on topcat (/etc/php5/php.ini, protected):
  ```
  error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
  display_errors = Off
  ```

• We can override the error configuration by calls in our PHP program. For proj2, that is done in main.php.
Getting errors to display on topcat

```php
<?php
echo 'setting display_errors on';
ini_set('display_errors',1); // This is done in main.php
echo $foo;  // Undefined variable
echo "Done";
```
Exceptions vs. Errors in PHP

- We saw that PDO objects throw exceptions, themselves Exception objects
- In pizza2, we are using the Guzzle component, which also throws on errors

- Exceptions are fairly new to PHP, with PHP 5, so exceptions are different from errors, the older non-OO setup.

- An uncaught exception causes a fatal error.
  - Thus we should catch possible exceptions
- A caught exception causes no error.
Useful Exception methods

Example from Tuts+ PHP Exception Tutorial:

```php
} catch (Exception $e) {
    echo "Message: " . $e->getMessage(). "\n\n";
    echo "File: " . $e->getFile(). "\n\n";
    echo "Line: " . $e->getLine(). "\n\n";
    echo "Trace: \n" . $e->getTraceAsString(). "\n\n";
}
```

Message: Email is invalid
File: C:\wamp\www\test\validator.php
Line: 7
Trace:
#0 C:\wamp\www\test\user.php(11): Validator->validate_email('(!$%#$%#*)'
#1 C:\wamp\www\test\test.php(12): User->save()
#2 {main}
New error.php in pizza2 (no more database_error.php)

Looks at object type of Exception for label, so can handle Guzzle, PDO, ...

Provides backtrace
Example of error.php output: expected error on rerun of restclient/index.php

GuzzleHttp\Exception\ServerException Error
Server error: `POST http://localhost/cs637/eoneil/proj2/proj2_server/rest/products/` resulted in a `500 Internal Server Error` response at line 113 in file C:\xampp\htdocs\cs637\eoneil\proj2\pizza2\vendor\guzzlehttp\guzzle\src\Exception\RequestException.php
full backtrace:
#0
C:\xampp\htdocs\cs637\eoneil\proj2\pizza2\vendor\guzzlehttp\guzzle\src\Middleware.php(65):
...
#9
C:\xampp\htdocs\cs637\eoneil\proj2\pizza2\vendor\guzzlehttp\guzzle\src\Client.php(131): GuzzleHttp\Promise\Promise->wait()
#10 C:\xampp\htdocs\cs637\eoneil\proj2\pizza2\restclient\index.php(69):
  ➔important info: line 69 of restclient/index.php did the POST that failed
    GuzzleHttp\Client->request('POST', 'http://localhost...', Array)
#11 {main}
#0 include() called at
[C:\xampp\htdocs\cs637\eoneil\proj2\pizza2\restclient\index.php:73]

5/3/2018
Inventory tracking by pizza2

- Pizza2 tracks flour and sugar inventory
  - Each pizza ordered by a student uses one unit of flour, one unit of cheese.
  - So you need a new inventory table.
- When the system sees supplies low, it orders more, using web services.
  - And saves the supply order ids, so it can later check for their delivery one or two days later (using a web service)
  - So you need another table for saving supply order ids.
- Specifically, the day manager checks inventory at the start of each day, even day 1, and
  - checks if any older orders have been delivered, and credits their quantities to the inventory
  - sends off web services for new orders as needed.
Code that needs work in pizza2: mostly in day manager

- pizza2/database/createdb.sql, dropdb.sql: needs new code to set up/drop inventory table, undelivered_orders table.
- pizza2/pizza/index.php: needs to fail an order if there is insufficient inventory. Deducts from inventory for successful order.
- pizza2/day/index.php: when doing next_day action, needs to do the inventory management and ordering. On any access, needs to find out inventory levels, etc. Calls into web_services.php to do needed web services to server to order more flour, cheese.
- pizza2/day/order_list.php: now needs to displays inventory information and undelivered supply orders
- pizza2/day/web_services.php: functions for each web service, using Guzzle to do the actual GET and POSTs. As in model, let the caller handle exceptions.
- pizza2/model/inventory_db.php: database actions for inventory management (also need to edit initial.php)
If doing both client and server

• Which to do first?
• I’d do the server-side first, because the provided proj2_tests allow for pretty good testing of the server
• Then the client will have an intelligent server to talk to, complain about erroneous requests
• But the client side can be coded with the help of the supplied server
If working alone on pizza2

- Or implementing pizza2 before the server, you need to be able to see the supply requests going out.
- Note that the provided server already prints one line on each incoming request, in proj2_server/php-server-errors.php:
  
  ```php
  starting REST server request, method=POST, uri = .../rest/products/
  ```

- This output is coming from line 32 of rest/index.php
  
  ```php
  error_log('starting REST server request, method=' . $method . ', uri = ...'. $project_uri);
  ```

- You can add to this, to print the body (in JSON) too:
  
  ```php
  error_log('body: ' . file_get_contents('php://input'));
  ```

- Or use `json_decode` on it to see it in PHP arrays

- Note if using Mac or Linux, you may need to `chmod 777 */*.log` from proj2 to make the logs writable, and if they aren’t there, first reconstruct them using `touch php-errors.log in pizza2 and touch php-server-errors.log in proj2_server`
Working on pizza2 without a finished server

- You have some experience with cooking up fake data ("mocks") from hw5
- Here you can temporarily code `get_server_supply_orders()` (or whatever you call it) to return a hard-coded array of supply orders for your inventory logic to work on.
- It could even use the day number to produce an appropriate sequence of orders
Code that needs work in proj2_server: mainly in rest/index.php

• proj2_server/rest/index.php: top-level web service code: need to fix GET/POST /day code, add /orders services
• proj2_server/rest/.htaccess: needed to get the Apache web server to execute index.php for any incoming URL .../rest/... (no edits needed)
• proj2_server/model/order_db.php: provided code has new add_order(), this may be sufficient.
• proj2_server/model/day.php: needs to retrieve and update the system day on the server
Command-line curl

- We have seen PHP’s libcurl in action
- Separately, we can use curl at the command line in Windows or Linux/Mac
- Download curl for Windows at http://curl.haxx.se/download.html (this is tricky, needs DLLs from Microsoft to work)
- Linux/Mac: should have curl already
- Also see tutorial there: http://curl.haxx.se/docs/httpscripting.html
Command-line curl example 1

From pa2.html:
curl localhost/cs637/username/proj2/proj2_server/rest/day/

This fires a GET to http://localhost/cs637...
i.e. does the Web service to get the current day from the server:

topcat$ curl localhost/cs637/username/proj2/proj2_server/rest/day/
6topcat$

Result (no end-of-line after the number, so shows up at start of next line)

Note: will always return 6 until you fix its code.
Command-line curl example 2

From pa2.html:
curl -i -d 9 -H Content-Type:text/plain
   http://localhost/cs637/username/proj2/proj2_server/rest/day/

This fires a POST to http://localhost/cs637...
i.e. does the Web service to set the current day to 9 in the server, and overrides
   the default Content-Type (URL-encoded, as needed for params with POST)

Without –i or –v for status info, or verbose status info:

topcat$ curl -d 9 -H Content-Type:text/plain
   http://localhost/cs637/username/proj2/proj2_server/rest/day/
topcat$

Nothing at all seen—how can we tell it worked?
Command-line curl example 2

With --v for verbose:

topcat$ curl --v -d 9 -H Content-Type:text/plain http://localhost/cs637/eoneil37/proj2_server/rest/day/

* Hostname was NOT found in DNS cache
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> POST /cs637/eoneil37/proj2_server/rest/day/ HTTP/1.1  ← POST command
> User-Agent: curl/7.35.0
> Host: localhost
> Accept: */*
> Content-Type:text/plain
> Content-Length: 1
>
* upload completely sent off: 1 out of 1 bytes  ← good news!
< HTTP/1.1 200 OK
< Date: Mon, 04 May 2015 14:33:16 GMT
* Server: Apache/2.4.7 (Ubuntu) is not blacklisted
< Server: Apache/2.4.7 (Ubuntu)
< X-Powered-By: PHP/5.5.9-1ubuntu4.9
< Content-Length: 0
< Content-Type: text/html
<
* Connection #0 to host localhost left intact
< more good news
Command-line curl example 2

With –i for status info: less clutter, get the basic facts:

curl -i -d 9 -H Content-Type:text/plain
   http://localhost/cs637/eoneill/proj2/proj2_server/rest/day/
HTTP/1.1 200 OK
Date: Sat, 21 Nov 2015 19:54:43 GMT
Server: Apache/2.4.9 (Win32) OpenSSL/1.0.1g PHP/5.5.11
X-Powered-By: PHP/5.5.11
Content-Length: 0
Content-Type: text/html
topcat$ curl -i -d 9 -H Content-Type:text/plain
    http://localhost/cs637/eoneill/proj2/proj2_server/rest/day/

Explanation of arguments:
- **-i**  return status info
- **-d 9 data**  use method POST, with POST data “9”. Defaults to Content-Type for encoded parameters, like form submission fields x=10&y=20 or whatever
- **-H Content-Type:text/plain**  override the default Content-Type to this type, plain text

So this command does a POST to the URL with just “9” as POST data, and reports verbosely on the action
Shell scripts

We can automate command line work with shell scripts (even on Windows)

```
topcat$ more test1.sh
curl -i -d 9 -H Content-Type:text/plain
    http://localhost/cs637/$1/proj2/proj2_server/rest/day/
topcat$ chmod +x test1.sh
topcat$ test1.sh eoneil
Fills in eoneil for $1 in script:
curl -i -d 9 -H Content-Type:text/plain
    http://localhost/cs637/eoneil/proj2/proj2_server/rest/day/
```

For Windows: test1.cmd: use %1% instead of $1.
See shell and .cmd files in proj2_tests directory.
proj2_tests

• Note: these test the server-side code only
  ➢ servertest0: tests provided functionality of proj2_server
  ➢ servertest1: test that server day number can be set, setting of day 0 reinitializes orders
  ➢ servertest2: sets day 0 to reinit orders, then sends an order in, gets the order back as order 1, or possibly a higher number on reruns
  ➢ servertest3: like servertest2, then set server day to 3, get order back, should be delivered
proj2_tests

Windows tests: needs curl installed
servertest0.cmd servertest1.cmd servertest2.cmd servertest3.cmd

Linux/Mac tests: system should have curl
servertest0.sh servertest1.sh servertest2.sh servertest3.sh
Be sure to “chmod +x *.sh” for easy execution
servertest0.sh run on topcat

topcat$ servertest0.sh eoneil

-------------get server day
HTTP/1.1 200 OK
...
Content-Length: 1
Content-Type: text/html

6
-------------set server day -- ineffective until coded right
HTTP/1.1 200 OK
...
-------------get server day again
HTTP/1.1 200 OK
...
6  ➤GOAL: this should be 9
-------------get product 1
...

5/3/2018
Grading run

- Run server test series
- Reinitialize: use dropdb, createdb of pizza2/database, send day 0 to proj2_server
- Run Selenium script:
  - Test0: initialize, see initial supply order for 100 units of flour, 50 units of cheese (or a little more)
  - Test1: order 10 pizzas and check inventory
  - Test2: order too many pizzas, check for error
  - Test3: reinit, advance day and check for supply order
  - Test4: check a supply order is delivered
- Possibly other tests...