Testing for Project Group1-Calendar

CIS 422, 2019 Winter

Group 1

Feb 4th, Mon

Author: Jarvis Dong, Shuijian Zhang

Objectives

This project is mostly driven by JavaScript and PHP files. To make sure all data output and function behaviors are as expected, this project includes unit test cases to assist implementation.

Tester Info

CIS 422 2019 Winter Group 1

Jarvis Dong, Shuijian Zhang

Unit Testing

Code Source Description

- This project is using an open source testing framework called Jest.
- The open source framework is conducted and mainly contributed by Facebook, Inc..
- Jest is selected to be the calandar project becasue Jest is designed to test JavaScript, simple to set up and efficient to use.
- To set up and use the framework, Npm.js is needed. The set up instruction is in below.
- There is one direcotry called *node_modules*, which includes necessary required JavaScript libraries for unit test.
- There is two files called *package-lock.json* and *package.json*.
    - File *package-lock.json* is generated by Npm.js to describe a single representation of a dependency tree such that teammates and deployments are guaranteed to install exactly the same version dependencies.
    - File *package.json* is writen by tester to define project properties, description, author & license info. For using Jest to test, *package.json* specifies its test scripts to "Jest".

- There are 4 seperate files writen by tester in one directory *test*. Each is responsible for testing a different JavaScript file.(e.g. *data.test.js* is testing *data.js*)

How To Test

1. Install Npm.js

1. Npm.js is installed with Node.js. To install Node.js, click the link and follow the instruction on the page.
2. To check if you have Node.js installed, run this command in your terminal:

   ```
   node -v
   ```

3. To confirm that you have npm installed you can run this command in your terminal:

   ```
   npm -v
   ```

4. If both print out version number v11.9.0 and 6.7.0, Node.js and Npm.js are successfully installed.

2. Install Jest

This is a simple step if Npm.js is installed. Run command in terminal:

```
npm install --save-dev jest
```

3. Test Project

In directory *Group1-Calendar*, run command in terminal:

```
npm test
```

Testing Result

- After a few seconds, a complete report on testing result will show on terminal, which includes all detail infomation on how many test cases get passed and how many do not.
- For test cases that not get passed, there are print statements specify where is the cases that not passed and reason why the expected result didn't show up.

Database Test

Code Source Description

- Database test is used an open source testing framework called phpunit.

- The open source framework is conducted and mainly contributed by Sebastian Bergmann and it's the best way to test database in php so far.
- To set up and use the framework, composer is needed. Follow the instruction as prepare environment called phpunit8 and library DbUnit for necessary. The set up instruction is in below.
- There is two files called *composer-lock.json* and *composer.json*.
    - File *composer-lock.json* records the exact versions that are installed. So that you are in the same versions with your co-workers.
    - File *composer.json* is generated by composer to describe a single representation of a dependency tree such that teammates and deployments are guaranteed to install exactly the same version dependencies.
    - **two files are almost same but composer-lock.json is encrypted.**
- Test file named DBTest, it just simply test the connection to database, input query file and check assertions for number of row in table, value in table, delete table.

How to test

1. **Install composer**

   To quickly install Composer in the current directory, run the following scripts in your terminal in order.

   ```
   php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
   ```

   ```
   php -r "if (hash_file('sha384', 'composer-setup.php') ===
   '48e3236262b34d30969dca3c37281b3b4bbe3221bda826ac6a9a62d6444cdb0dcd0615698a
   5cbe587c3f0fe57a54d8f5') { echo 'Installer verified'; } else { echo
   'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
   ```

   ```
   php composer-setup.php
   ```

   ```
   php -r "unlink('composer-setup.php');"
   ```

2. **You can add PHPUnit as a local, per-project, development-time dependency to your project using Composer:**

   ```
   ➜ composer require --dev phpunit/phpunit ^8

   ➜ ./vendor/bin/phpunit --version
   PHPUnit 8.0.0 by Sebastian Bergmann and contributors.
   ```

The example shown above assumes that `composer` is on your `$PATH`.

Your `composer.json` should look similar to this:

```
{
    "autoload": {
        "classmap": [
            "src/"
        ]
    },
    "require-dev": {
        "phpunit/phpunit": "^8"
    }
}
```

3. **Install DbUnit**

   If you use Composer to manage the dependencies of your project then you can add DbUnit as a development-time dependency to your project:

   ```
   $ composer require --dev phpunit/dbunit
   ```

4. **Test Execution**

   ```
   ➜ ./vendor/bin/phpunit --bootstrap vendor/autoload.php tests/EmailTest
   PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

   ...                                                                 3 / 3
   (100%)

   Time: 70 ms, Memory: 10.00MB

   OK (3 tests, 3 assertions)
   ```

Test Result:

Output should be located under the command line you just executed. Here are three different situations:

- **"." mention successful assertion**. An example is in the test execution above.
- **"E" mention Error in program.** Sometimes when your code has syntax error or runtime error, it'll output E.
- **"F" mention False assertion**. Return F means your assertion is differnet as expect. It can be tracked by compiler for what reasons.