COmanage Testing

Unit Testing

The COmanage development team leverages the CakePHP integrated testing based on PHPUnit for unit testing of models, controllers, and other non-view classes.

Running Unit Tests

Before running unit tests on a system you will need to do the following:

1. Deploy the COmanage Registry code base as you normally would including running the console database and setup scripts. It is necessary that a database for the 'default' connection configured in Config/database.php exist and be populated with the COmanage Registry tables before running tests. The default database will be read to obtain model schema information but test data will **not** be written to the default database.
2. Create a second database instance in your database server to use for testing and configure the 'test' connection in Config/database.php appropriately, for example

```
    public $test = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => 'comanage',
        'password' => 'XXXXXXXX',
        'database' => 'registry_test',
        'prefix' => 'cm_'
    );
```

3. Install the latest version of the 3.x series of PHPUnit. The 4.x series is **not** compatible with CakePHP version 2.x. On a Debian 7 system one can do

```
sudo apt-get install phpunit
```

   to obtain version 3.6.10.
4. Optionally install the Xdebug extension for PhP. On a Debian 7 system one can do

```
sudo apt-get install php5-xdebug
```

5. Edit Config/core.php and set 'debug' to 2.
6. If you intend to run tests with a web browser (recommended during development of new tests) configure Apache HTTP Server as you normally would to deploy the Registry. It is not necessary to configure an authentication mechanism for the Registry login functionality but it also does not cause an issue.

Unit tests with a web browser

Browse to https://<server>/registry/test.php to expose the CakePHP Test Suite. Links to tests for plugins and the CakePHP core appear towards the top. Links to tests for the Registry application models and controllers appear towards the bottom. Click on a link for a model or controller test to execute the tests.

To aid in debugging, especially when writing tests, click "Enable Debug Output" or simply append `&debug=1` to the URL.

If you have installed Xdebug you can click on "Analyze Code Coverage".

Unit tests on the command line

Before running tests on the command line make sure that the user that will run the tests has the necessary privileges to write into the tmp/cache/ directory.

To execute tests name the model or controller like this:

```
./Console/cake test app Model/Co --stderr
```

Writing Unit Tests

Fixtures

The name of the file containing the fixture for a model class is not the name of the class prepended with 'Fixture', but rather the name of the model class with only the first letter capitalized. So for example when considering the class CoGroup in the file Model/CoGroup.php, the name of the file for the fixture is Test/Fixture/CogroupFixture.php and **not** Test/Fixture/CoGroupFixture.php.

Integration and UI Testing

COmanage Functional / Acceptance Testing with Selenium IDE

Selenium is used for functional and acceptance testing for COmanage Registry. The instructions below are for creating and running test suites with Selenium IDE plugin for Firefox. These tests will form the basis of automated tests using Mink to drive the tests - http://mink.behat.org/en/latest/.

Prerequisites

- Firefox and Selenium IDE Plugin:
  - http://www.seleniumhq.org/download/
  - scroll way down the page to find Selenium IDE
- Four test users must exist with the ability to authenticate; user ids in the tests are
  - cmp-admin
  - co-admin
  - cou-admin
  - co-user
- Test users use the password “comanage”
- These tests assume the ability to authenticate using HTTP basic auth. It is no longer possible to automatically login using the http://user:pass@domain.com pattern in later versions of Firefox (versions 30+). So, prior to running the tests:
  - Authenticate as the user for the appropriate test suite (e.g. cmp-admin for CmpAdminTestSuite)
  - Then click “logout”
  - You can now run the tests from the start through login. Because basic auth will stick around, you can emulate the experience of authenticating without having to do it again. (There are better solutions for this when using Webdriver via the API – see Mink.)
  - If authentication is not possible using this method, authenticate and run the test suite from the first test case post-login.