PHPUnit and Drupal Test Traits

TABLE OF CONTENTS

Overview

Tests are written as PHP classes. These are executed in the terminal (or optionally from within PhpStorm). Each test consists of code with calls to various assert functions to test if an expected result matches an actual result. So if a function should return true, you will usually assertTrue($return_value) or if a unit returns 5, you will assertEquals(5, $return_value).

E.g.

```
self::assertTrue($return_value);
```

from https://phpunit.readthedocs.io/en/9.5/writing-tests-for-phpunit.html

```
<?php
declare(strict_types=1);

use PHPUnit\Framework\TestCase;

final class StackTest extends TestCase
{
    public function testPushAndPop(): void
    {
        $stack = [];
        $this->assertSame(0, count($stack));

        array_push($stack, 'foo');
        $this->assertSame('foo', $stack[count($stack)-1]);
        $this->assertSame(1, count($stack));

        $this->assertSame('foo', array_pop($stack));
        $this->assertSame(0, count($stack));
    }
}
```

In a Drupal context, there are 4 types of tests. From https://www.drupal.org/docs/automated-testing/types-of-tests :

- Unit: PHPUnit-based tests with minimal dependencies. Base class: Drupal\Tests\UnitTestCase class. They must be clean plain PHP.

- Kernel: PHPUnit-based tests with a bootstrapped kernel, and a minimal number of extensions enabled. Base class: Drupal\KernelTests\KernelTestBase class. More at https://www.drupal.org/docs/automated-testing/phpunit-in-drupal/kerneltestbase
- Functional: PHPUnit-based tests with a full booted Drupal instance. Base class: Drupal\Tests\BrowserTestBase.
- FunctionalJavascript: PHPUnit-based tests that use Webdriver to perform tests of Javascript and Ajax functionality in the browser. Base class: Drupal\FunctionalJavascriptTests\WebDriverTestBase.

In addition, the Drupal project has some tests for JavaScript, written in JavaScript, that use the Nightwatch framework.

Drupal Test Traits (DTT) adds 2 more types of tests:

- ExistingSite which use the Drupal API and existing site. Base class: weitzman\DrupalTestTraits\ExistingSiteBase
- ExistingSiteJavascript for testing Javacript or AJAX using Selenium and Chromedriver. Base class: weitzman\DrupalTestTraits\ExistingSiteSelenium2DriverTestBase

There are many examples of tests in core and contributed modules.

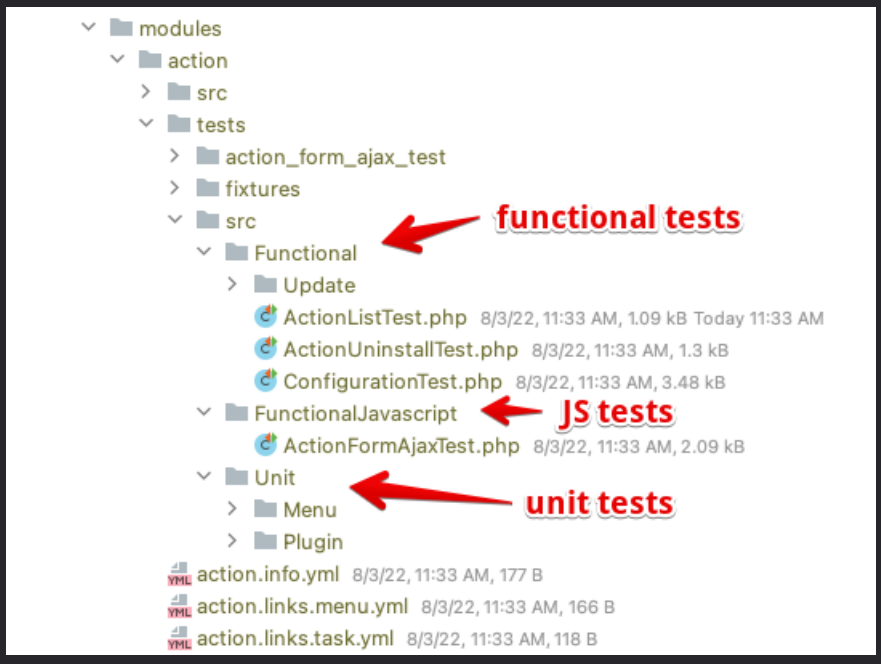
e.g. in docroot/modules/contrib/admin_toolbar/tests/src/Functional/ there are 3 functional tests: AdminToolbarAdminMenuTest.php, AdminToolbarAlterTest.php and AdminToolbarToolsSortTest.php

Every time a PHPUnit test is run, a fresh Drupal database and files are created. This guarantees that any existing data won't taint your test's outcomes. DTT bypasses this process and uses the existing site although it can clean up anything that is created in the test.

Location of PHPUnit Tests

Tests are always in the tests folder of modules. They are often in different folders e.g. tests/src/Unit, tests/src/Functional, tests/src/FunctionalJavascript

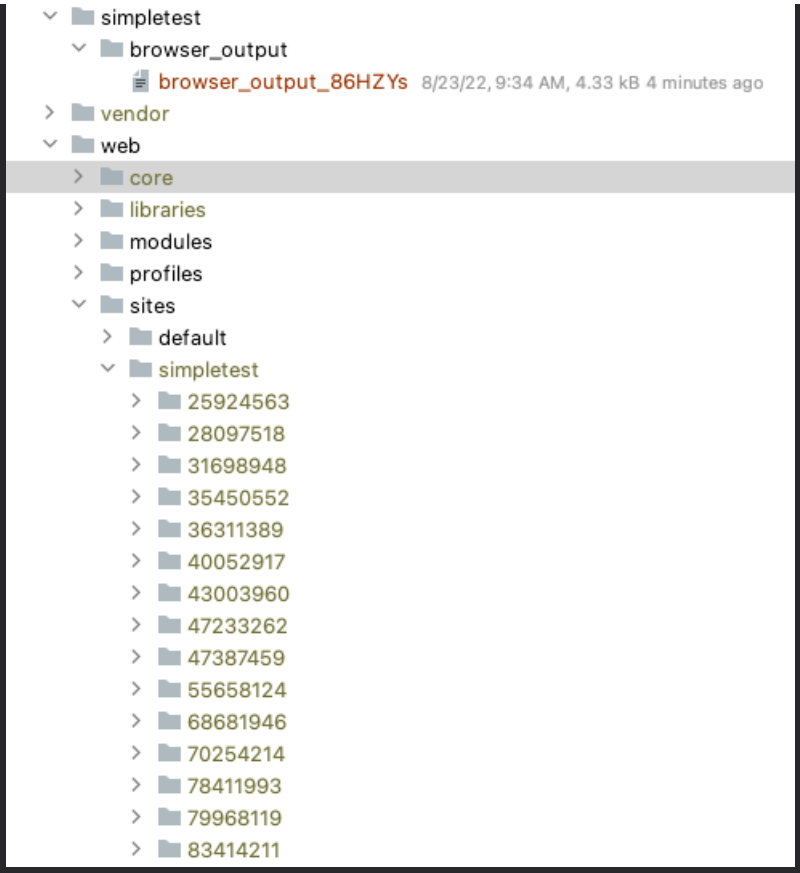
For core they are in web/core/modules (or docroot/core/modules) e.g. in the action module, the directory structure looks like this:



Output files from running phpunit

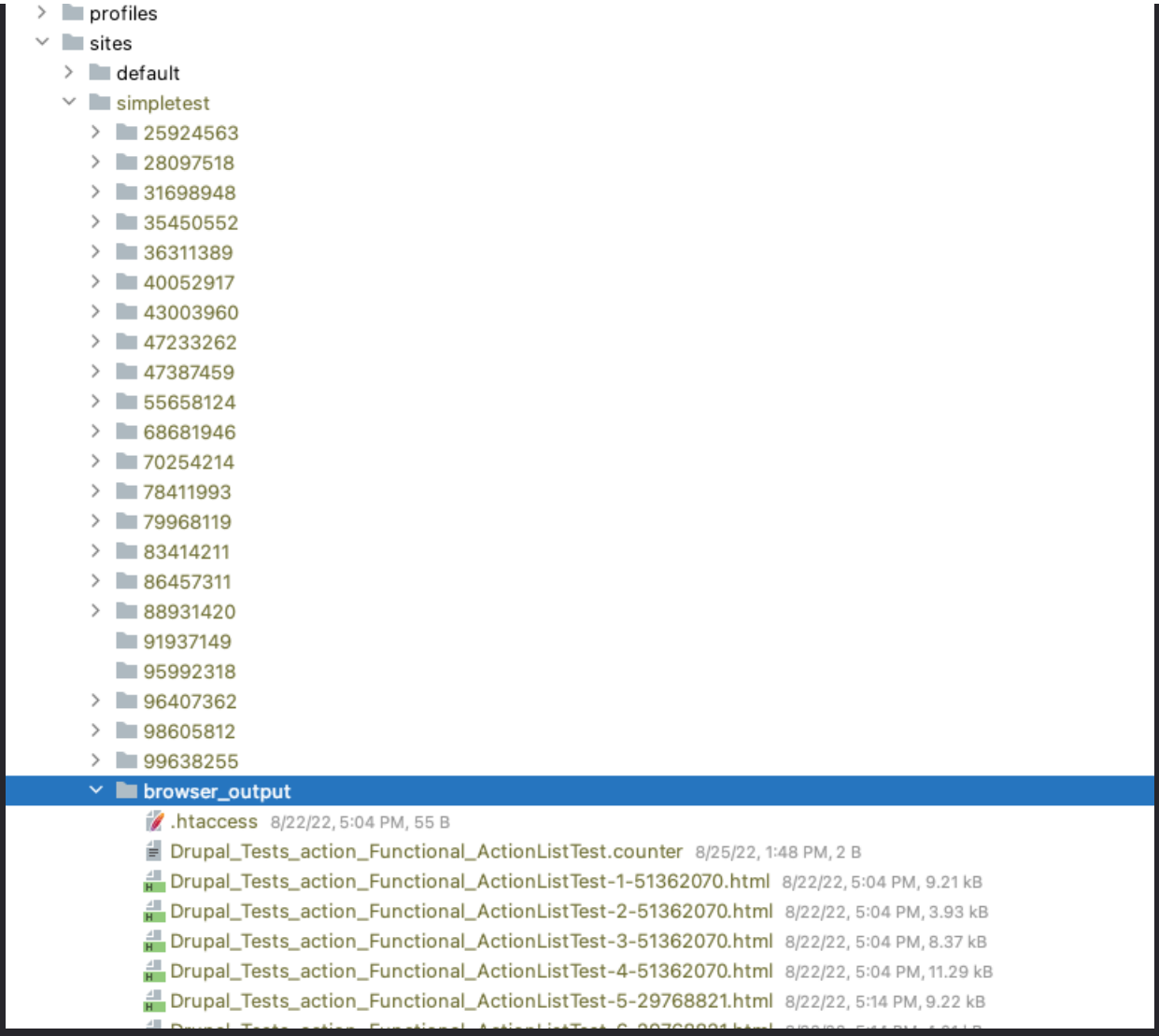
Depending on where you specified the output for your files in the phpunit.xml file e.g. in web/core/phpunit.xml I specified this path. Below you will see a bunch of directories (probably one for each run of the tests) in sites/simpletest/nnnnn

```
<!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /path/to/webroot/sites/simpletest/browser_output -->
<env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/var/www/html/simpletest/browser_output"/>
```

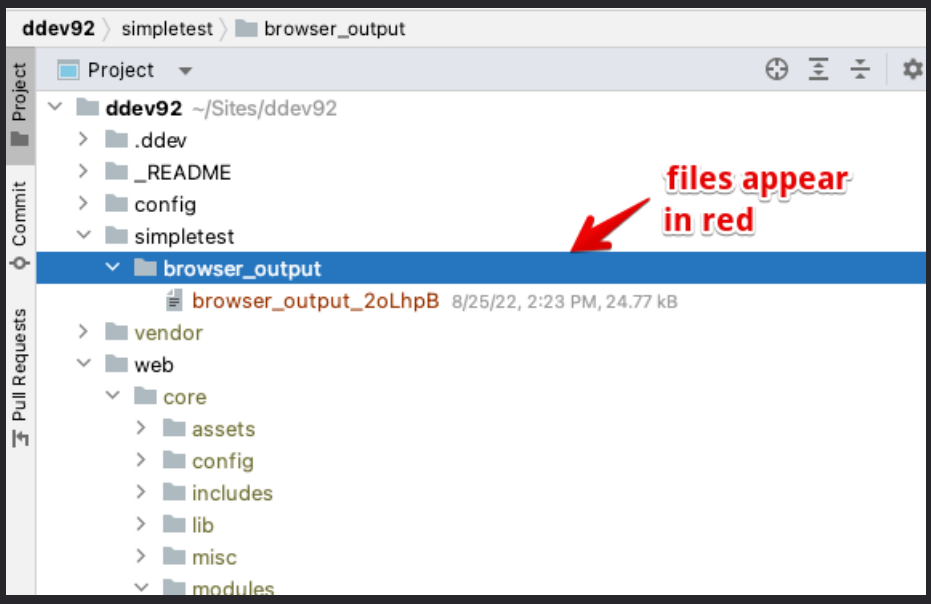


If you don't need to view the reports from your tests, you can safely delete these directories as well as the html files shown below.

At the same level is a browser_output directory which has some html files which reference the directories above:



While tests are running, I noticed that files appear in the /simpletest/browser_output folder at the topmost level of the project. They go away when the tests complete..



Setup PHPUnit tests

Using Drupal version 9.4.5 let's get PHPunit tests running inside the ddev containers (not on the host machine).

To get started with phpunit, follow instructions at https://www.drupal.org/docs/automated-testing/phpunit-in-drupal/running-phpunit-tests (I updated this for ddev on 8-29-22)

Another useful reference is the web/core/tests/README.md that comes with Drupal.

If you installed using the drupal/recommended-project Composer template, development, you will need some dependencies which can be installed by requiring drupal/core-dev as a dependency in your project.

```
$ composer require drupal/core-dev --dev --update-with-all-dependencies
```

Install phpunit using:

```
$ composer require phpunit/phpunit
```

Also you might need prophecy for mocking:

```
$ composer require --dev phpspec/prophecy-phpunit:^2
```

You need a web/core/phpunit.xml so copy the one at web/core/phpunit.xml.dist.

Change the SIMPLETEST_BASE_URL, SIMPLETEST_DB AND BROWSERTEST_OUTPUT_DIRECTORY variables as shown below

```
<php>
  <!-- Set error reporting to E_ALL. -->
  <ini name="error_reporting" value="32767"/>
  <!-- Do not limit the amount of memory tests take to run. -->
  <ini name="memory_limit" value="-1"/>

  <env name="SIMPLETEST_BASE_URL" value="http://localhost"/>
  <env name="SIMPLETEST_DB" value="mysql://db:db@db/db"/>
  <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/var/www/html/simpletest/browser_output"/>
```

Refer to https://mglaman.dev/blog/running-drupals-phpunit-test-suites-ddev for more info on using ddev describe to get suitable values for those variables

From web/core/tests/README.md

Copy the core/phpunit.xml.dist file to phpunit.xml, and place it somewhere convenient (inside the core directory may not be the best spot, since that directory may be managed by Composer or Git). You can use the -c option on the command line to tell PHPUnit where this file is (use the full path).

Settings to change in this file:

- SIMPLETEST_BASE_URL: The URL of your site
- SIMPLETEST_DB: The URL of your Drupal database
- The bootstrap attribute of the top-level phpunit tag, to take into account the location of the file
- BROWSERTEST_OUTPUT_DIRECTORY: Set to sites/simpletest/browser_output; you will also want to uncomment the printerClass attribute of the top-level phpunit tag.

Running PHPUnit tests in the DDEV container

This will run the unit tests included with the action module

```
$ ddev ssh
$ cd web
../vendor/bin/phpunit -c core core/modules/action
```

The -c is a reference to the phpunit configuration file called phpunit.xml.

Another example test you could run is the book module:

You could also run `../vendor/bin/phpunit -c core core/modules/book` for a different set of tests. These tests are for the book module and take a very LONG time. (~30 minutes)

When running the tests for the action module, you should see:

```
PHPUnit 9.5.23 #StandWithUkraine

Warning:       Your XML configuration validates against a deprecated schema.
Suggestion:    Migrate your XML configuration using "--migrate-configuration"!

Testing /var/www/html/web/core/modules/action
....S..
7 / 7 (100%)

Time: 03:25.204, Memory: 8.00 MB

OK, but incomplete, skipped, or risky tests!
Tests: 7, Assertions: 229, Skipped: 1.

HTML output was generated
http://localhost/sites/simpletest/browser_output/Drupal_Tests_action_Functional_ActionListTest-1-51362070.html
http://localhost/sites/simpletest/browser_output/Drupal_Tests_action_Functional_ActionListTest-2-51362070.html
http://localhost/sites/simpletest/browser_output/Drupal_Tests_action_Functional_ActionListTest-3-51362070.html
http://localhost/sites/simpletest/browser_output/Drupal_Tests_action_Functional_ActionListTest-4-51362070.html
http://localhost/sites/simpletest/browser_output/Drupal_Tests_action_Functional_ActionUninstallTest-1-80829434.html
```

Note in the output: ….S.., periods mean success, S means skipped, I means ignored, E means error.

You can choose to skip(S) or ignore(I) tests during development. This is done to speed up test runs or if you are mid-development and want to only run specific tests. Also these can be conditional such as when you only want a certain test to run if a condition is met e.g. a driver is present.

More details at https://phpunit.readthedocs.io/en/9.5/textui.html

Run PHPUnit test in DDEV from the host

You can also run the test in the containers from the host with:

```
$ ddev exec ./vendor/bin/phpunit -c web/core web/core/modules/action
```

Run PHPUnit test on the host

If you have installed PHP on your host computer, you should also be able to run Phpunit tests there with:

From the very top of the project:

```
$ vendor/bin/phpunit -c web/core web/core/modules/action
```

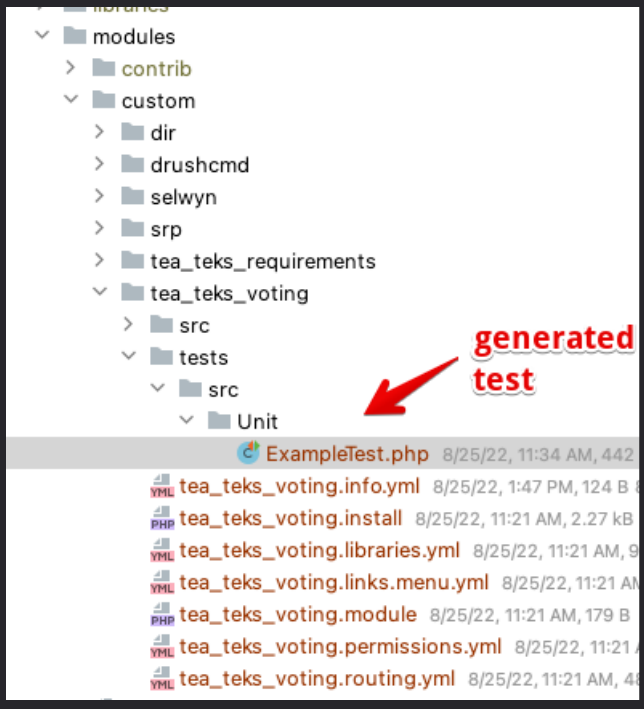
Generate a unit test with drush

You can use drush to generate a module e.g. drush gen module. (Just follow the prompts). Once you have a module, use drush gen test:unit for your module. Note. You need drush 11 for this.

Here is a generated file at modules/custom/tea_teks_voting/tests/src/Unit/ExampleTest.php with the following contents. It does not contain a real test; just a shell of a test which can run.

```php
<?php

namespace Drupal\Tests\tea_teks_voting\Unit;

use Drupal\Tests\UnitTestCase;

/**
 * Test description.
 *
 * @group tea_teks_voting
 */
class ExampleTest extends UnitTestCase {

  /**
   * {@inheritdoc}
   */
  protected function setUp(): void {
    parent::setUp();
    // @todo Mock required classes here.
  }

  /**
   * Tests something.
   */
  public function testSomething() {
    self::assertTrue(TRUE, 'This is TRUE!');
  }

}
```

Drush wrote the test file in modules/custom/tea_teks_voting/tests/src/Unit/ExampleTest.php



To run this test use

```
$ drush ssh

selwyn@ddev92-web:/var/www/html \$ vendor/bin/phpunit -c web/core/
web/modules/custom/tea_teks_voting/
```

You should see the following output:

```
PHPUnit 9.5.23 #StandWithUkraine

Warning:       Your XML configuration validates against a deprecated schema.
Suggestion:    Migrate your XML configuration using "--migrate-configuration"!

Testing /var/www/html/web/modules/custom/tea_teks_voting
.                                                   1 / 1 (100%)

Time: 00:00.058, Memory: 4.00 MB

OK (1 test, 1 assertion)
```

Note. It does expect the file web/core/phpunit.xml to exist and be configured correctly. See setup above for details.

My first PHPUnit test

This is my test of a class I wrote called Requirements. It does require some Drupal to be bootstrapped, so it can't be a unit test. It must be a functional test and therefor in the `modules/custom/tea_teks_requirements/tests/src/Functional` directory.

```
<?php

namespace Drupal\Tests\tea_teks_requirements\Unit;

//use PHPUnit\Framework\TestCase;
use Drupal\Tests\UnitTestCase;
use Drupal\tea_teks_requirements\Requirements;

class RequirementsTest extends UnitTestCase {

  public function testEmptyRequirements() {
    $r = new Requirements();
    $empty = $r->isEmpty();
    $this->assertFalse($empty);
  }
}
```

The module must be enabled.

Copy web/core/phpunit.xml.dist to web/core/phpunit.xml

Edit the following lines of the file to look something like this:

```
<env name="SIMPLETEST_BASE_URL" value="http://localhost"/>
<env name="SIMPLETEST_DB" value="mysql://db:db@db/db"/>
<env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/var/www/html/simpletest/browser_output"/>
```

Run the test with

```
$ ddev ssh

$ cd web

$ ../vendor/bin/phpunit -c core modules/custom/tea_teks_voting
```

You will see this output:

```
PHPUnit 9.5.23 #StandWithUkraine

Warning:       Your XML configuration validates against a deprecated schema.
Suggestion:    Migrate your XML configuration using "--migrate-configuration"!

Testing /var/www/html/web/modules/custom/tea_teks_voting
.
1 / 1 (100%)

Time: 00:00.055, Memory: 4.00 MB

OK (1 test, 1 assertion)
```

Note. The OK means it worked.

Drupal Test Traits Overview

Drupal Test Traits is for testing Drupal sites that have content (versus Phpunit tests which start Drupal up, create an empty database and are consequently slower).

From https://gitlab.com/weitzman/drupal-test-traits:

> Behat is great for facilitating conversations between business managers and developers. Those are useful conversations, but many organizations simply can't or won't converse via Gherkin. When you are on the hook for product quality and not conversations, this is a testing approach for you.
>
> Before Drupal Test Traits, this framework was impossible to use without wiping the site's database after each test. DTT lets you keep your database and still test using the features of Drupal's BrowserTestBase and friends. See DrupalTrait::setUp() for details (the bootstrap is inspired by Drush).

- Blog: [Introducing Drupal Test

  > Traits](https://medium.com/massgovdigital/introducing-drupal-test-traits-9fe09e84384c)

- Blog: [Introducing Drupal Test Traits: Drupal extension for testing

  > existing sites](https://www.previousnext.com.au/blog/introducing-drupal-testing-traits-drupal-extension-testing-existing-sites)

- Video: [Drupalcon presentation - Introducing Drupal Test

  > Traits](https://www.tag1consulting.com/blog/introducing-drupal-test-traits).

- Repo: https://gitlab.com/weitzman/drupal-test-traits

> DTT also supports testing through a real browser using headless Chrome or Selenium. So, testing client-side interactions like autocomplete, #states, viewports, and drag/drop is easy.
>
> Like Drupal core, DTT can save HTML snapshots for each URL that it navigates to. These files are very useful when debugging test failures.

Install/setup Drupal Test Traits

TLDR; You will need Drupal test traits installed with composer, possibly `drupal/core-dev` (also a composer install), a `/phpunit.xml` file, and a `/scripts/bootstrap-fast.php`. Add `weitzman/logintrait` with composer for adding users and logging in to your site. Finally for AJAX testing add a `docker-compose.testing.yaml` and using composer add `behat/mink-selenium2-driver`.

The details are as follows:

1. Install DTT. At the time of this writing the 1.6 version was out but there is a 2.x dev branch. Moshe recommends using that so use the following command to install it:

```
$ composer require --dev weitzman/drupal-test-traits:^2
```

1. Install the dev requirements with:

```
$ composer require drupal/core-dev --dev --update-with-all-dependencies
```

If you skip the `drupal/core-dev` step, you will see errors when you try to run the tests like this:

> selwyn@tea3-web:/var/www/html$ ./vendor/bin/phpunit --bootstrap=./vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php ./docroot/modules/custom/tea_teks/modules/tea_teks_requirements/tests/src/ExistingSite/RequirementsCreationTest.php
>
> PHP Fatal error: Trait "Symfony\Bridge\PhpUnit\Legacy\PolyfillAssertTrait" not found in /var/www/html/docroot/sites/simpletest/Assert.php on line 91
>
> Fatal error: Trait "Symfony\Bridge\PhpUnit\Legacy\PolyfillAssertTrait" not found in /var/www/html/docroot/sites/simpletest/Assert.php on line 91

1. Create a `phpunit.xml` file. I put `phpunit.xml` in the root of the project (not docroot or web.) See example file contents below.
2. Create a `bootstrap-fast.php`. I put `bootstrap-fast.php` in the /scripts (also in the root of the project). This file is included in DTT at `vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php`.
3. Add a .ddev/docker-compose.testing.yaml (don't accidentally type docker-compose**R** as Ddev won't create that container and you'll be left puzzled.
4. Install the mink-selenium2 driver with:\

```
$ composer require 'behat/mink-selenium2-driver' --dev
```

1. Install login traits with:

```
$ composer require weitzman/logintrait
```

Here are the `/phpunit.xml` file contents. Note the location of the `bootstrap-fast.php`:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- Copy the samples below into your own phpunit.xml file.-->

<!-- Using this project's bootstrap file allows tests in `ExistingSite`,
    `ExistingSiteSelenium2DriverTest`, and `ExistingSiteWebDriverTest`
    to run alongside core's test types. -->

<!-- If you use the default `bootstrap-fast.php` and get 'class not
    found' errors while running tests, head over to
    https://gitlab.com/weitzman/drupal-test-traits/-/blob/master/src/bootstrap-fast.php
    for explanation on how to register those classes.
-->
<!--<phpunit bootstrap="vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php">-->
<phpunit bootstrap="scripts/bootstrap-fast.php">
  <php>
<!--    <env name="DTT_BASE_URL" value="https://ddev92.ddev.site"/>-->
    <env name="DTT_BASE_URL" value="http://localhost"/>
    <env name="DTT_API_URL" value="http://localhost:9222"/>
    <!-- <env name="DTT_MINK_DRIVER_ARGS" value='["chrome", { "chromeOptions" : { "w3c": false } }, "http://localhost:4444/wd/hub"]'/> -->
    <env name="DTT_MINK_DRIVER_ARGS" value='["firefox", null, "http://localhost:4444/wd/hub"]'/>
    <env name="DTT_API_OPTIONS" value='{"socketTimeout": 360, "domWaitTimeout": 3600000}' />
    <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /tmp
         Specify a temporary directory for storing debug images and html documents.
         These artifacts get copied to /sites/simpletest/browser_output by BrowserTestBase. -->
    <env name="BROWSERTEST_OUTPUT_DIRECTORY" value=""/>
    <!-- To disable deprecation testing completely uncomment the next line. -->
    <!--<env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>-->
    <!-- Specify the default directory screenshots should be placed. -->
    <!--<env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>-->
    <!-- Specify the default directory page captures should be placed.
        When using the \Drupal\Tests\Listeners\HtmlOutputPrinter printerClass this will default to
        /sites/simpletest/browser_output. If using another printer such as teamcity this must be defined.
        -->
    <!--<env name="DTT_HTML_OUTPUT_DIRECTORY" value=""/>-->
  </php>

  <testsuites>
    <testsuite name="unit">
      <directory>./web/modules/custom/*/tests/src/Unit</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/Unit</directory>-->
    </testsuite>
    <testsuite name="kernel">
      <directory>./web/modules/custom/*/tests/src/Kernel</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/Kernel</directory>-->
    </testsuite>
    <testsuite name="existing-site">
      <!-- Assumes tests are namespaced as \Drupal\Tests\custom_foo\ExistingSite. -->
      <directory>./web/modules/custom/*/tests/src/ExistingSite</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/ExistingSite</directory>-->
    </testsuite>
    <testsuite name="existing-site-javascript">
      <!-- Assumes tests are namespaced as \Drupal\Tests\custom_foo\ExistingSiteJavascript. -->
      <directory>./web/modules/custom/*/tests/src/ExistingSiteJavascript</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/ExistingSiteJavascript</directory>-->
    </testsuite>
  </testsuites>
</phpunit>
```

Here are the /scripts/bootstrap-fast.php contents from vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php:

```
<?php
/**
 * @file
 *   A bootstrap file for `phpunit` test runner.
 *
 * This bootstrap file from DTT is fast and customizable.
 *
 * If you get 'class not found' errors while running tests, you should copy this
 * file to a location inside your code-base --such as `/scripts`. Then add the
 * missing namespaces to the bottom of the copied field. Specify your custom
 * `bootstrap-fast.php` file as the bootstrap in `phpunit.xml`.
 *
 * Alternatively, use the bootstrap.php file, in this same directory, which is
 * slower but registers all the namespaces that Drupal tests expect.
 */

use Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter;
use weitzman\DrupalTestTraits\AddPsr4;

list($finder, $class_loader) = AddPsr4::add();
$root = $finder->getDrupalRoot();

// So that test cases may be simultaneously compatible with multiple major versions of PHPUnit.
$class_loader->addPsr4('Drupal\TestTools\\', "$root/core/tests");
if (class_exists('Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter')) {
  ClassWriter::mutateTestBase($class_loader);
}

// Register more namespaces, as needed.
# $class_loader->addPsr4('Drupal\Tests\my_module\\', "$root/modules/custom/my_module/tests/src");
```

Here is the docker-compose.testing.yaml from Michael Strelan on drupal.org. Once you add this and restart DDEV, you will be able to do AJAX and Javascript testing of DTT tests. Note this is a “just works” situation as Michael puts it. Note that if you have this file in place, you don’t need to provide all the env values in the phpunit.xml above.

```
version: '3.6'
services:
  web:
    links:
      - chromedriver:$DDEV_HOSTNAME
    environment:
      SYMFONY_DEPRECATIONS_HELPER: weak
      SIMPLETEST_DB_MYSQL: &simpletest_db_mysql mysql://db:db@db:3306/db
      SIMPLETEST_DB_SQLITE: &simpletest_db_sqlite sqlite://tmp/sites/simpletest/simpletest.db
      SIMPLETEST_DB: *simpletest_db_sqlite
      SIMPLETEST_BASE_URL: &base_url http://${DDEV_HOSTNAME}
      MINK_DRIVER_ARGS_WEBDRIVER: &mink_driver_args '["chrome", {"browserName":"chrome","goog:chromeOptions":{"args":["--disable-gpu","--headless", "--no-sandbox",
      BROWSERTEST_OUTPUT_DIRECTORY: /var/www/html/web/sites/simpletest/browser_output
      BROWSERTEST_OUTPUT_BASE_URL: ${DDEV_PRIMARY_URL}
      DTT_BASE_URL: *base_url
      DTT_MINK_DRIVER_ARGS: *mink_driver_args

  chromedriver:
    image:  drupalci/webdriver-chromedriver:production
    container_name: ddev-${DDEV_SITENAME}-chromedriver
    labels:
      com.ddev.site-name: ${DDEV_SITENAME}
      com.ddev.approot: $DDEV_APPROOT
    external_links:
      - ddev-router:${DDEV_SITENAME}.${DDEV_TLD}
    networks: [default, ddev_default]
```

What about testing browser interaction?

Without the docker-compose.testing.yaml, you can only run the most basic DTT tests such as ExampleTest.php.

The lack of a headless Chrome container, will cause the ExampleWebDriverTest.php and ExampleSeleniumDriverTest.php to fail with errors like this when you try to run tests:

```
Error: Class "DMore\ChromeDriver\ChromeDriver\" not found
```

Using behat/mink-selenium2-driver, you can run DTT against Chrome, Firefox or Edge (in WebDriver mode.) This setup also allows us to run Drupal core JS testing using Nightwatch. Although DTT supports browser testing using headless Chrome or Selenium. This facility is deprecated.

From https://gitlab.com/weitzman/drupal-test-traits:

> (deprecated) **ExistingSiteWebDriverTest**. See ExampleWebDriverTest.php. These tests make use of a headless Chrome browser but using Chrome's Debugger Protocol. They are suited to testing Ajax and similar client side interactions. These tests run slower than ExistingSite. To use this test type you need to composer require 'dmore/chrome-mink-driver' --dev. Tests of this type should be placed in tests/src/ExistingSiteJavascript. Contrary to its name, this test type does not use the WebDriver protocol at all.

My first DTT tests

Tests must be in the ExistingSite directory and be namespaced as Drupal\Tests\module_name\ExistingSite.

Create modules/custom/tea_teks_requirements/tests/src/ExistingSite/RequirementsCreationTest.php with the following contents. This has 2 tests. Notice that we can use Drupal API calls just like when we write modules. This test is provided as an example of a simple test so you can see what they look like in real life.

```php
<?php

namespace Drupal\Tests\tea_teks_requirements\ExistingSite;

use Drupal\node\Entity\Node;
use Drupal\tea_teks_requirements\Requirements;
use weitzman\DrupalTestTraits\ExistingSiteBase;

class RequirementsCreationTest extends ExistingSiteBase {


  public function testEmptyRequirements() {
    $r = new Requirements();
    $empty = $r->isEmpty();
    $this->assertTrue($empty);
  }

  public function testLoadingRequirementsFromStandard() {
    $teks_standard_node = Node::load(8);
    $r = new Requirements($teks_standard_node);
    $empty = $r->isEmpty();
    $this->assertFalse($empty);
  }

}
```

Run it with:

```
$ ddev ssh

$ vendor/bin/phpunit --bootstrap=./vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php ./web/modules/custom/tea_teks_requirements/tests/src/ExistingSite/RequirementsCr
```

The output looks something like this:

> PHPUnit 9.5.23 #StandWithUkraine
>
> .. 2 / 2 (100%)
>
> Time: 00:01.338, Memory: 16.00 MB
>
> OK (2 tests, 2 assertions)

Along with a boatload of deprecation notices. (Note. we can hide these with `<env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>` in the `<php>` section of the `phpunit.xml` file):

> Remaining direct deprecation notices (3)
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\MimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\MimeTypes" instead.
>
> 1x in RequirementsCreationTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\FileBinaryMimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\FileBinaryMimeTypeGuesser" instead.
>
> 1x in RequirementsCreationTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\FileinfoMimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\FileinfoMimeTypeGuesser" instead.
>
> 1x in RequirementsCreationTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite

> Remaining indirect deprecation notices (2)
>
> 1x: Return type of GuzzleHttp\Cookie\CookieJar::count() should either be compatible with Countable::count(): int, or the #[\ReturnTypeWillChange] attribute should be used to temporarily suppress the notice
>
> 1x in RequirementsCreationTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: Return type of GuzzleHttp\Cookie\CookieJar::getIterator() should either be compatible with IteratorAggregate::getIterator(): Traversable, or the #[\ReturnTypeWillChange] attribute should be used to temporarily suppress the notice
>
> 1x in RequirementsCreationTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite

Running DTT tests

To run all the tests in the `modules/custom/tea_teks_requirements` directory, use the following:

```
$ ddev ssh

$ vendor/bin/phpunit --bootstrap=./vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php modules/custom/tea_teks_requirements
```

The output for a successful run looks like this:

> PHPUnit 9.5.23 #StandWithUkraine
>
> . 1 / 1 (100%)
>
> Time: 00:07.798, Memory: 20.00 MB
>
> OK (1 test, 7 assertions)

Along with a boatload of deprecation notices. (Note. we can hide these with `<env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>` in the `<php>` section of the `phpunit.xml` file)

> Remaining direct deprecation notices (3)
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\MimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\MimeTypes" instead.
>
> 1x in ExampleTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\FileBinaryMimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\FileBinaryMimeTypeGuesser" instead.
>
> 1x in ExampleTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: The "Symfony\Component\HttpFoundation\File\MimeType\FileinfoMimeTypeGuesser" class is deprecated since Symfony 4.3, use "Symfony\Component\Mime\FileinfoMimeTypeGuesser" instead.
>
> 1x in ExampleTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> Remaining indirect deprecation notices (3)
>
> 1x: Return type of GuzzleHttp\Cookie\CookieJar::count() should either be compatible with Countable::count(): int, or the #[\ReturnTypeWillChange] attribute should be used to temporarily suppress the notice
>
> 1x in ExampleTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: Return type of GuzzleHttp\Cookie\CookieJar::getIterator() should either be compatible with IteratorAggregate::getIterator(): Traversable, or the #[\ReturnTypeWillChange] attribute should be used to temporarily suppress the notice
>
> 1x in ExampleTest::setUp from Drupal\Tests\tea_teks_requirements\ExistingSite
>
> 1x: "Symfony\Component\DomCrawler\Crawler::text()" will normalize whitespaces by default in Symfony 5.0, set the second "$normalizeWhitespace" argument to false to retrieve the non-normalized version of the text.
>
> 1x in ExampleTest::testLlama from Drupal\Tests\tea_teks_requirements\ExistingSite

Once you've set up your bootstrap-fast.php file in /scripts, you can specify it in the phpunit.xml as

```
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="scripts/bootstrap-fast.php"
         verbose="true"
        >
```

Then you don't need to pass it as a parameter. So running the test can look like this. Note no more reference to --bootstrap on the command line:

```
./vendor/bin/phpunit ./docroot/modules/custom/tea_teks/modules/tea_teks_voting/Tests/src/ExistingSite/RequirementsCreationTest.php
```

Or more simply:

```
vendor/bin/phpunit  docroot/modules/custom/tea_teks/modules/tea_teks_voting/Tests/src/ExistingSite/RequirementsCreationTest.php
```

Also note you can specify the phpunit.xml file with -c parameter.

```
./vendor/bin/phpunit -c ./phpunit.xml  ./docroot/modules/custom/tea_teks/modules/tea_teks_voting/Tests/src/ExistingSite/RequirementsCreationTest.php
```

Run tests on the host

You can use the same commands that you run in the DDEV containers if you have php 8.1 installed and running. They will run faster on the host. e.g.:

```
$ vendor/bin/phpunit docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/TeamTest.php
```

Note. You can specify the location of bootstrap-fast.php in your /phpunit.xml. This file is found at vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php. Here a copy of bootstrap-fast.php is in the /scripts directory:

```
<?xml version="1.0" encoding="UTF-8"?>

<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="scripts/bootstrap-fast.php"
         verbose="true"
        >
```

The source for bootstrap-fast.php is:

```php
<?php
/**
 * @file
 *   A bootstrap file for `phpunit` test runner.
 *
 * This bootstrap file from DTT is fast and customizable.
 *
 * If you get 'class not found' errors while running tests, you should copy this
 * file to a location inside your code-base --such as `/scripts`. Then add the
 * missing namespaces to the bottom of the copied field. Specify your custom
 * `bootstrap-fast.php` file as the bootstrap in `phpunit.xml`.
 *
 * Alternatively, use the bootstrap.php file, in this same directory, which is
 * slower but registers all the namespaces that Drupal tests expect.
 */

use Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter;
use weitzman\DrupalTestTraits\AddPsr4;

list($finder, $class_loader) = AddPsr4::add();
$root = $finder->getDrupalRoot();

// So that test cases may be simultaneously compatible with multiple major versions of PHPUnit.
$class_loader->addPsr4('Drupal\TestTools\\', "$root/core/tests");
if (class_exists('Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter')) {
  ClassWriter::mutateTestBase($class_loader);
}

// Register more namespaces, as needed.
# $class_loader->addPsr4('Drupal\Tests\my_module\\', "$root/modules/custom/my_module/tests/src");
```

Run tests in a specific file

You can be specific and only run all tests in a particular test file e.g.

```
$ vendor/bin/phpunit web/modules/custom/tea_teks_requirements/tests/src/ExistingSite/ExampleTest.php
```

Here is an example where the location of the bootstrap file is specified with --bootstrap:

```
$ ./vendor/bin/phpunit --bootstrap=./vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php ./web/modules/custom/tea_teks_requirements/tests/src/ExistingSite/ExampleTest.pl
```

Run a specific test in a file

You can run a test that is called ‘testVoter1Vote’ in the VotingPageTest.php file with the following command. Note. If you have another test that starts with testVoter1Vote e.g. testVoter1VoteBlah, that test will be run also.

```
$ vendor/bin/phpunit --filter testVoter1Vote docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/VotingPageTest.php
```

Logging Test Output

PhpUnit can do all sorts of logging.

Capture every page loaded

For debugging, capturing all HTML requests can be useful.

At https://gitlab.com/weitzman/drupal-test-traits#debugging-tests Moshe suggests:

• All HTML requests can be logged. To do so, add BROWSERTEST_OUTPUT_DIRECTORY=/tmp and --printer '\\Drupal\\Tests\\Listeners\\HtmlOutputPrinter' to the phpunit call. To disable deprecation notices, include SYMFONY_DEPRECATIONS_HELPER=disabled. Alternatively, you can specify these in your phpunit.xml (example phpunit.xml).

To add the printerclass to the phpunit.html see the printerClass line below:

```
<?xml version="1.0" encoding="UTF-8"?>

<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="scripts/bootstrap-fast.php"
         verbose="true"
         printerClass="\Drupal\Tests\Listeners\HtmlOutputPrinter">
  <php>
```

Putting this in the <php> section of the file causes all html requests to be output to /sites/simpletest/browser_output. I tried specifying a different directory with but it had no effect. Use this with caution (or only for debugging if you don't want to fill up hard drives.)

```
  <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/tmp"/>
```

See the entire <php> section below:

```
<php>
  <env name="DTT_BASE_URL" value="http://tea.ddev.site"/>
  <env name="DTT_API_URL" value="http://chrome:9222"/>
  <env name="DTT_MINK_DRIVER_ARGS" value='["chrome", {"browserName":"chrome","chromeOptions":{"args":["--disable-gpu","--headless", "--no-sandbox"]}}, "http://chrome
  <env name="DTT_API_OPTIONS" value='{"socketTimeout": 360, "domWaitTimeout": 3600000}' />
  <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /tmp
       Specify a temporary directory for storing debug images and html documents.
       These artifacts get copied to /sites/simpletest/browser_output by BrowserTestBase. -->
  <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/tmp"/>
  <!-- To disable deprecation testing completely uncomment the next line. -->
  <env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>
  <!-- Specify the default directory screenshots should be placed. -->
  <!--<env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>-->
  <!-- Specify the default directory page captures should be placed.
       When using the \Drupal\Tests\Listeners\HtmlOutputPrinter printerClass this will default to
       /sites/simpletest/browser_output. If using another printer such as teamcity this must be defined.
       -->
  <!--<env name="DTT_HTML_OUTPUT_DIRECTORY" value=""/>-->
</php>
```

Capture an HTML page

From https://gitlab.com/weitzman/drupal-test-traits#debugging-tests:

• To write the current HTML of the page to a file, use $this->capturePageContent(). If using HtmlOutputPrinter this will be saved to the browser_output directory. Alternatively you can specify DTT_HTML_OUTPUT_DIRECTORY=/path/to/output_directory which is required when using a different printer, such as Teamcity, which is enforced by PHPStorm.

Example of capturing a page

```
$url = Url::fromRoute('tea_teks_publisher.correlation_detail', [
    'node1' => $this->teaPublisherNid,
    'node2' => $this->testProgramOneNid,
    'node3' => $expectation_nid,
    'node4' => $correlation_nid,
  ]
);
$page_source = $this->drupalGet($url);
$this->capturePageContent();
```

Use the following setting to specify the output directory in the phpunit.xml file

```
<env name="DTT_HTML_OUTPUT_DIRECTORY" value="sites/simpletest/browser_output"/>
```

Here is the whole <php> section:

```
<php>
  <env name="DTT_BASE_URL" value="http://tea.ddev.site"/>
  <env name="DTT_API_URL" value="http://chrome:9222"/>
  <env name="DTT_MINK_DRIVER_ARGS" value='["chrome", {"browserName":"chrome","chromeOptions":{"args":["--disable-gpu","--headless", "--no-sandbox"]}}, "http://chrome
  <env name="DTT_API_OPTIONS" value='{"socketTimeout": 360, "domWaitTimeout": 3600000}' />
  <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /tmp
       Specify a temporary directory for storing debug images and html documents.
       These artifacts get copied to /sites/simpletest/browser_output by BrowserTestBase. -->
  <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/tmp"/>
  <!-- To disable deprecation testing completely uncomment the next line. -->
  <env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>
  <!-- Specify the default directory screenshots should be placed. -->
  <!--<env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>-->
  <!-- Specify the default directory page captures should be placed.
       When using the \Drupal\Tests\Listeners\HtmlOutputPrinter printerClass this will default to
       /sites/simpletest/browser_output. If using another printer such as teamcity this must be defined.
       -->
  <env name="DTT_HTML_OUTPUT_DIRECTORY" value="sites/simpletest/browser_output"/>
</php>
```

Screenshot using ExistingSiteSelenium2DriverTest

From https://gitlab.com/weitzman/drupal-test-traits#debugging-tests:

• To take a screenshot of the current page under ExistingSiteSelenium2DriverTest, use \weitzman\DrupalTestTraits\ScreenShotTrait::captureScreenshot. Be careful when using this to debug tests that are "randomly" failing. Most likely, these tests are failing due to missing [waitForElementVisible]{.underline} checks, as the act of taking a screenshot gives the browser additional time to finish rendering the page.

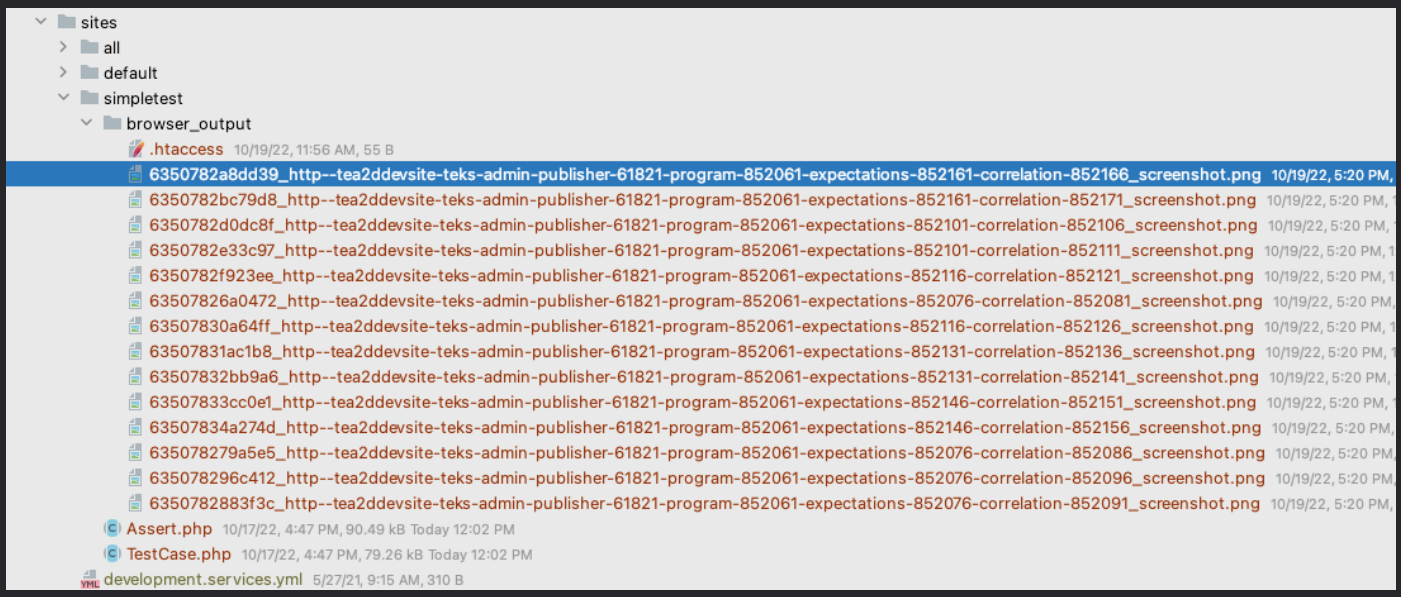
This trait doesn't need to be installed separately, it is included with DTT. Just add a use statement to your class e.g.

```
class Pub1Test extends ExistingSiteSelenium2DriverTestBase {
  use \weitzman\DrupalTestTraits\ScreenShotTrait;

  private int $teaPublisherUid = 5071;
  private int $teaPublisherNid = 61821;

  public function testSetup() {
    $this->markTestIncomplete('later');
...
```

Calling it in code:

```php
// Load the publisher correlation detail page.
do {
  $url = Url::fromRoute('too_teks_publisher.correlation_detail', [
     'node1' => $this->tooPublisherNid,
     'node2' => $this->testProgramOneNid,
     'node3' => $expectation_nid,
     'node4' => $correlation_nid,
   ]
  );
  $page_source = $this->drupalGet($url);
  //$this->capturePageContent();
  $this->captureScreenshot();
```

The output appears as png files like:



Curiously, this only captures the visible part of the page - I notice parts of it were clipped.

Location of the files is specified in the phpunit.xml in the <php> section as

```
<env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>
```

That whole <php> section looks like:

```
<php>
  <env name="DTT_BASE_URL" value="http://tea.ddev.site"/>
  <env name="DTT_API_URL" value="http://chrome:9222"/>
  <env name="DTT_MINK_DRIVER_ARGS" value='["chrome", {"browserName":"chrome","chromeOptions":{"args":["--disable-gpu","--headless", "--no-sandbox"]}}, "http://chrome
  <env name="DTT_API_OPTIONS" value='{"socketTimeout": 360, "domWaitTimeout": 3600000}' />
  <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /tmp
       Specify a temporary directory for storing debug images and html documents.
       These artifacts get copied to /sites/simpletest/browser_output by BrowserTestBase. -->
  <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/tmp"/>
  <!-- To disable deprecation testing completely uncomment the next line. -->
  <env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>
  <!-- Specify the default directory screenshots should be placed. -->
  <!-- <env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>-->
  <!-- Specify the default directory page captures should be placed.
       When using the \Drupal\Tests\Listeners\HtmlOutputPrinter printerClass this will default to
       /sites/simpletest/browser_output. If using another printer such as teamcity this must be defined.
       -->
  <env name="DTT_HTML_OUTPUT_DIRECTORY" value="sites/simpletest/browser_output"/>
</php>
```

Writing DTT Tests

Test locations

Tests that require no Ajax or Javascript are put in the ExistingSite directory. These will run quite quickly especially if you run them on the host (instead of in the DDEV/Docker containers). These tests are derived from ExistingSiteBase.

Putting tests in the ExistingSiteJavascript directory (and deriving them from ExistingSiteSelenium2DriverTestBase) will cause the test to be run against the Chromedriver which can handle Javascript and Ajax.

Generate DTT tests with drush

Use the latest drush (11 at this time) to generate both types of tests using:

```
$ drush generate test:existing
```

And

```
$ drush generate test:existing-js
```

More at https://www.drush.org/latest/generators/

Example tests

ExampleTest.php creating user term, article

From https://gitlab.com/weitzman/drupal-test-traits/-/blob/2.x/tests/ExampleTest.php this example creates a user, a taxonomy term, an article, retrieves it, checks for a return code of 200, logs in as the user and retrieves the node edit page.

```php
<?php

namespace Drupal\Tests\tea_teks_voting\ExistingSite;

use Drupal\taxonomy\Entity\Vocabulary;
use weitzman\DrupalTestTraits\ExistingSiteBase;

/**
 * A model test case using traits from Drupal Test Traits.
 */
class ExampleTest extends ExistingSiteBase {

  protected function setUp(): void {
    parent::setUp();

    // Cause tests to fail if an error is sent to Drupal logs.
    $this->failOnLoggedErrors();
  }

  /**
   * An example test method; note that Drupal API's and Mink are available.
   *
   * @throws \Drupal\Core\Entity\EntityStorageException
   * @throws \Drupal\Core\Entity\EntityMalformedException
   * @throws \Behat\Mink\Exception\ExpectationException
   */
  public function testLlama() {
    // Creates a user. Will be automatically cleaned up at the end of the test.
    $author = $this->createUser([], null, true);

    // Create a taxonomy term. Will be automatically cleaned up at the end of the test.
    $vocab = Vocabulary::load('tags');
    $term = $this->createTerm($vocab);

    // Create a "Llama" article. Will be automatically cleaned up at end of test.
    $node = $this->createNode([
      'title' => 'Llama',
      'type' => 'article',
      'field_tags' => [
        'target_id' => $term->id(),
      ],
      'uid' => $author->id(),
    ]);
    $node->setPublished()->save();
    $this->assertEquals($author->id(), $node->getOwnerId());

    // We can browse pages.
    $this->drupalGet($node->toUrl());
    $this->assertSession()->statusCodeEquals(200);

    // We can login and browse admin pages.
    $this->drupalLogin($author);
    $this->drupalGet($node->toUrl('edit-form'));
  }
}
```

VotingPageTest

This test runs code in a project and at various times asserts that various things are true or equal to expected values. It also does some setup

including logging in as a voter. More about that below.

```
class VotingPageTest extends ExistingSiteBase {

  protected int $testProgramOneNid = 852061;
  protected int $publisher_user_id = 3501;  // DannyTest Lufkin.
  protected int $correlationNid = 852086;
  protected int $expectationNid = 852076;
  protected int $teaPublisherNid = 61821;
  protected ?int $adminUserId = NULL;

  public function testVoter1Vote() {
//    $this->markTestIncomplete('later');

    $this->setupTestProgram1ForVotingRound0();

    $this->loginVoter1();
    $program_nid = $this->testProgramOneNid;
    $this->program  = new Program($program_nid);
    $this->persona = new Persona($this->voter1UserId);

    $citation_nid = 858676;
    $citation_nid = 858741;
    $citation_node = Node::load($citation_nid);
    $expectation_nid = $citation_node->get('field_tks_pub_expectation')->target_id;
    $correlation_nid = $citation_node->get('field_tks_pub_correlation')->target_id;

    $options = [
      'program_nid' => $program_nid,
      'expectation_nid' => $expectation_nid,
      'correlation_nid' => $correlation_nid,
      'citation_nid' => $citation_nid,
      'rejection_reason' => 'auto-test rejection reason',
    ];

    $this->votingProcessor = \Drupal::service('tea_teks_srp.vote_processor');
    $correlation_node = Node::load($correlation_nid);
    $valid_voting_path = $this->votingProcessor->loadVotingPath($this->program->getProgramNode(), 'all');
    $this->votingProcessor->setVotingRequirements($correlation_node);
    self::assertTrue($valid_voting_path);

    $this->votingProcessor->voteOnCitation($options, 'accepted');
    $voting_record_node = $this->votingProcessor->loadVotingRecord($citation_nid, $this->voter1UserId);
    $vote = $voting_record_node->get('field_vote')->value;
    self::assertEquals('accepted', $vote);

    $this->votingProcessor->voteOnCitation($options, 'rejected');
    $voting_record_node = $this->votingProcessor->loadVotingRecord($citation_nid, $this->voter1UserId);
    $vote = $voting_record_node->get('field_vote')->value;
    self::assertEquals('rejected', $vote);

    $this->votingProcessor->cancelVoteOnCitation($citation_nid, $correlation_nid);
    $voting_record_node = $this->votingProcessor->loadVotingRecord($citation_nid, $this->voter1UserId);
    self::assertEmpty($voting_record_node);
  }
}
```

Here is the setupTestProgram1ForVoting

This checks some data in nodes, logs in as an admin user and fills out a form that executes a batch api process. I wasn't able to make a batch api process run directly from a test. Not sure why. Moshe Weitzman says it should work but that the batch API is ancient.

```
protected function setupTestProgram1ForVotingRound0() {
  //$this->resetProgram1Voting();

  $program_node = Node::load($this->testProgramOneNid);
  $teks_program_status = $program_node->get('field_tks_program_status')->value;
  if ($teks_program_status == 'publisher_complete') {
    $this->loginAdminUser();
    $page_source = $this->drupalGet(Url::fromRoute('tea_teks_publisher.change_input_collection_status', ['node' => 852071,]));

    // To check if the form displayed correctly you can look for something in the $page_source and check the return code.
    $this->assertSession()->statusCodeEquals(200);

    $this->submitForm([
      'program_status' => 'ready_for_release',
    ], 'Change Status');

    // Confirm that the voting requirements were written to a correlation.
    $correlation_node = Node::load(852081);
    $voting_requirements_json = $correlation_node->get('field_voting_requirements_json')->value;
    self::assertNotNull($voting_requirements_json);
  }
  // Set team.
  $this->setupVotingTeamA();
  $this->setupTestProgram1TeamAForRound0();
}
```

Other supporting functions

```
protected function setupVoter1() {
  $query = \Drupal::entityQuery('user')
    ->condition('name', 'Voter 1 Test');
  $uids = $query->execute();
  if (empty($uids)) {
    $this->setupVotingTeamA();
    $query = \Drupal::entityQuery('user')
      ->condition('name', 'Voter 1 Test');
    $uids = $query->execute();
  }
  $uids = array_values($uids);
  $this->voter1UserId = $uids[0];
}
```

and

```
protected function loginVoter1() {
  $this->setupVoter1();
  $user = User::load($this->voter1UserId);
  $user->passRaw = 'password';
  $this->drupalLogin($user);
}
```

ExampleLoginTest.php

From https://gitlab.com/weitzman/logintrait/-/blob/master/src/ExampleLoginTest.php This example shows how to use the Login trait.

```php
<?php

// Use your module's testing namespace such as the one below.
namespace Drupal\Tests\moduleName\ExistingSite;

use weitzman\DrupalTestTraits\ExistingSiteBase;
use weitzman\LoginTrait\LoginTrait;

/**
 * Login and Logout via user reset URL instead of forms. Useful when TFA/SAML are enabled.
 */
class ExampleLoginTest extends ExistingSiteBase {
    use LoginTrait;

    /**
     * Login and logout via password reset URL.
     */
    public function testLoginLogout() {
        // Creates a user. Will be automatically cleaned up at the end of the test.
        $user = $this->createUser();
        $this->drupalLogin($user);
        $this->drupalGet('user');
        $user2 = $this->createUser();
        $this->drupalLogin($user2);
    }
}
```

Selenium2DriverTest

This is an example from the Weitzman repo using AJAX

```php
<?php

// Use your module's testing namespace such as the one below.
//namespace Drupal\Tests\tea_teks_voting\ExampleSelenium2DriverTest;
namespace Drupal\Tests\tea_teks_voting\ExistingSiteJavascript;

use Drupal\node\Entity\Node;
use Drupal\taxonomy\Entity\Vocabulary;
use Drupal\user\Entity\User;
use weitzman\DrupalTestTraits\ExistingSiteSelenium2DriverTestBase;

/**
 * A WebDriver test suitable for testing Ajax and client-side interactions.
 */
class ExampleSelenium2DriverTest extends ExistingSiteSelenium2DriverTestBase
{
  public function testContentCreation()
  {
    // Create a taxonomy term. Will be automatically cleaned up at the end of the test.
    $web_assert = $this->assertSession();
    $vocab = Vocabulary::load('tags');
    $this->createTerm($vocab, ['name' => 'Term 1']);
    $this->createTerm($vocab, ['name' => 'Term 2']);
    $admin = User::load(1);
    $admin->passRaw = 'password';
    $this->drupalLogin($admin);

    // @codingStandardsIgnoreStart
    // These lines are left here as examples of how to debug requests.
    // \weitzman\DrupalTestTraits\ScreenShotTrait::captureScreenshot();
    // $this->capturePageContent();
    // @codingStandardsIgnoreStop

    // Test autocomplete on article creation.
    $this->drupalGet('/node/add/article');
    $page = $this->getCurrentPage();
    $page->fillField('title[0][value]', 'Article Title');
    $tags = $page->findField('field_tags[target_id]');
    $tags->setValue('Ter');
    $tags->keyDown('m');
    $result = $web_assert->waitForElementVisible('css', '.ui-autocomplete li');
    $this->assertNotNull($result);
    // Click the autocomplete option
    $result->click();
    // Verify that correct the input is selected.
    $this->assertStringContainsString('Term 1', $tags->getValue());
    $submit_button = $page->findButton('Save');
    $submit_button->press();
    // Verify the URL and get the nid.
    $this->assertTrue((bool) preg_match('/.+node\/(?P<nid>\d+)/', $this->getUrl(), $matches));
    $node = Node::load($matches['nid']);
    $this->markEntityForCleanup($node);
    // Verify the text on the page.
    $web_assert->pageTextContains('Article Title');
    $web_assert->pageTextContains('Term 1');
  }
}
```

Login to your site

There are some contributed packages with useful features. The login trait repo adds some useful functionality.

Install via Composer with:

```
$ composer require weitzman/logintrait
```

Create a new user and login as that user:

From https://gitlab.com/weitzman/logintrait/-/blob/master/src/ExampleLoginTest.php

```
<?php

// Use your module's testing namespace such as the one below.
namespace Drupal\Tests\moduleName\ExistingSite;

use weitzman\DrupalTestTraits\ExistingSiteBase;
use weitzman\LoginTrait\LoginTrait;

/**
 * Login and Logout via user reset URL instead of forms. Useful when TFA/SAML are enabled.
 */
class ExampleLoginTest extends ExistingSiteBase {
    use LoginTrait;

    /**
     * Login and logout via password reset URL.
     */
    public function testLoginLogout() {
        // Creates a user. Will be automatically cleaned up at the end of the test.
        $user = $this->createUser();
        $this->drupalLogin($user);
        $this->drupalGet('user');
        $user2 = $this->createUser();
        $this->drupalLogin($user2);
    }
}
```

Create an admin user

This will create a user named Fred Bloggs who is in a new randomly named group. The user and the group will be deleted when the test run finishes.

```
// Creates a user. Will be automatically cleaned up at the end of the test.
$user = $this->createUser([], 'Fred Bloggs', TRUE);
$this->drupalLogin($user);
$this->drupalGet('user');
```

Login as an existing user

Be sure to set that user's password to "password" (in the Drupal U/I or in code) in order to make this work.

```
$voter1_user_id = 5284;
$voter1 = User::load($voter1_user_id);
$voter1->passRaw = 'password';
$this->drupalLogin($voter1);
```

Add users using Drupal API

```
$voter1 = User::create([
  'name' => $this->voter1['name'],
  'field_firstname' => $this->voter1['field_firstname'],
  'field_lastname' => $this->voter1['field_lastname'],
  'field_srp_voter_role' => $this->voter1['field_srp_voter_role'],
  'field_phone' => $this->voter1['field_phone'],
  'field_title' => $this->voter1['field_title'],
]);
$voter1->addRole('srp_voter');
$voter1->setEmail('voter1@mightycitizen.com');
$voter1->setPassword('password');
$voter1->activate();
$voter1->save();
$voter1_uid = $voter1->id();
```

Fill out a form

Here is the code from docroot/core/tests/Drupal/Tests/UiHelperTrait.php to fill out the login form:

```
$this->drupalGet(Url::fromRoute('user.login'));
$this->submitForm([
  'name' => $account->getAccountName(),
  'pass' => $account->passRaw,
], 'Log in');
```

Here is another example:

```
// Load the form.
$url = Url::fromRoute('tea_teks_admin.sanity_checker', [
    'program' => $this->testProgramOneNid,
  ]
);
// Confirm that it loaded without errors.
$this->assertSession()->statusCodeEquals(200);
// Check the destructive checkbox and click the 'verify vote counts' button.
$this->submitForm(['destructive' => 1], 'Verify Vote Counts');
```

In the above example the code in render array for the form that builds the destructive checkbox and the submit button looks like this:

```
$form['sanity_fieldset']['destructive'] = [
  '#type' => 'checkbox',
  '#title' => t('Check this box to permanently update statuses.'),
  '#description' => t('Recalculate all votes and statuses for current vote number. Leave unchecked for testing.'),
];
$form['sanity_fieldset']['actions'] = [
  '#type' => 'actions',
];
$form['sanity_fieldset']['actions']['submit'] = [
  '#type' => 'submit',
  '#value' => $this->t('Verify Vote Counts'),
];
```

Parameter gotcha

And my effort to fill out a form with a dropdown. This route required a node id to be passed as a parameter to the form – hence the [‘node’=> 852071] and my submit button is called “Change Status”

```
$this->drupalGet(Url::fromRoute('tea_teks_publisher.change_input_collection_status', ['node' => 852071,]) );
$this->submitForm([
  'program_status' => 'ready_for_release',
], 'Change Status');
```

Note. When you define a form in Drupal, it permits you to use a different variable name in the routing file versus the parameter in the buildForm() function. E.g. Here the parameter is called “program”:

```
tea_teks_srp.reset_program_votes:
  path: '/teks/admin/srp/program/{program}/resetvotes'
  defaults:
    _form: '\Drupal\tea_teks_srp\Form\SrpResetProgramVotesForm'
    _title: 'Reset Program Votes'
  requirements:
    _permission: 'manage teks srp process'
  options:
    parameters:
      program:
        type: entity:node
    no_cache: 'TRUE'
```

In the form, the parameter can be something different. i.e. the `$node` parameter here represents the program parameter above. If you change them to match i.e. change the parameter in the buildForm function below, it should work fine.

```
public function buildForm(array $form, FormStateInterface $form_state, EntityInterface $node = NULL) {
  $form['#theme'] = 'tea_teks_srp__reset_votes';
  if ( ($node->id() && $node->bundle() == 'teks_pub_program') ) {
    $request = \Drupal::request();
    $referer = $request->headers->get('referer');
    $base_url = Request::createFromGlobals()->getSchemeAndHttpHost();
    $alias = substr($referer, strlen($base_url));
    $form_state->set('referrer_alias', $alias);
    $current_user = \Drupal::currentUser();
    if ($current_user->hasPermission('manage teks srp process')) {
      $form['actions'] = [
        '#type' => 'actions',
      ];
      $form['actions']['submit'] = [
        '#type' => 'submit',
        '#value' => $this
          ->t('TESTING ONLY: Reset Program Votes/Data'),
      ];
      $form_state->set('program_id', $node->id());
      $form_state->set('program_title', $node->title->value);
    }
  }
  return $form;
}
```

So if you try in the test to execute this form and pass it a parameter called “program” it will fail to load the form. You will see errors like:

> There was 1 error:
>
> 1) Drupal\Tests\tea_teks_voting\ExistingSite\Vote1::testSetup
>
> Behat\Mink\Exception\ElementNotFoundException: Button with id|name|label|value "TESTING ONLY: Reset Program Votes/Data" not found.

Note. This code will do the same thing if you put it in the ExistingSite or the ExistingSiteJavascript directory however, putting it in the ExistingSiteJavascript directory (and deriving the test from ExistingSiteSelenium2DriverTestBase) will cause the test to be run against the

Chromedriver which can handle Javascript and Ajax.

Data Provider

Tests can be repeated with varying values by providing a data provider function. The data provider just returns an array of values and the function below has annotation indicating to PHPUnit to rerun the test once for each value in the data provider.

```
public function providerForTest1(): array {
  return [
    [858641, 'accepted',],
    [858651, 'accepted',],
    [858661, 'accepted',],
    [858676, 'accepted',],
    [858721, 'accepted',],
    [858726, 'accepted',],
    [858736, 'accepted',],
    [858741, 'accepted',],
  ];
}

/**
 * @dataProvider ProviderForTest1
 */
public function test1(int $citation_nid, string $vote) {
  static $voter1_uid = 0;
  $this->setupTestProgram1ForVotingRound0();
  $this->loginVoter1();
  $this->program  = new Program($this->testProgramOneNid);
  $this->persona = new Persona($voter1_uid);
  $this->votingProcessor = \Drupal::service('tea_teks_srp.vote_processor');
  $valid_voting_path = $this->votingProcessor->loadVotingPath($this->program->getProgramNode(), 'all');
  self::assertTrue($valid_voting_path);

  $citation_node = Node::load($citation_nid);
  $correlation_nid = $citation_node->get('field_tks_pub_correlation')->target_id;
  $expectation_nid = $citation_node->get('field_tks_pub_expectation')->target_id;
  $correlation_node = Node::load($correlation_nid);
  $this->votingProcessor->setVotingRequirements($correlation_node);
  $voting_options = [
    'program_nid' => $this->testProgramOneNid,
    'expectation_nid' => $expectation_nid,
    'correlation_nid' => $correlation_nid,
    'citation_nid' => $citation_nid,
    'rejection_reason' => 'auto-test rejection reason',
  ];
  $this->votingProcessor->voteOnCitation($voting_options, 'accepted');
  $voting_record_node = $this->votingProcessor->loadVotingRecord($citation_nid, $this->voter1UserId);
  $vote_value_from_node = $voting_record_node->get('field_vote')->value;
  self::assertEquals($vote, $vote_value_from_node);

}
```

Fill a queue and run a trait

From Moshe Weitzman 9-27-22

I have seen tests that fill a queue and then run the queue with https://github.com/drupaltest/queue-runner-trait/

//@TODO: Explore this

Mink

Checking page return code

This only works for non-Selenium/Chromedriver type test:

```
$session = $this->getSession();
$status_code = $session->getStatusCode();
print "\n Current Status code: $status_code";
```

In the ExampleTest.php there was this example:

```
// We can browse pages.
$this->drupalGet($node->toUrl());
$this->assertSession()->statusCodeEquals(200);
```

Grab the text from the page

You can do some interesting things when running Selenium type tests. Here we can grab the text and search in it for a particular string.

This will get you the text that is visible on the page. It is unformatted and is one long string.

```
$session = $this->getSession();
$page = $session->getPage();
$page_text = $page->getText();
```

Current URL

```
$url_string = $this->getSession()->getCurrentUrl();
print "\n Current URL: $url_string";
```

Load and parse a CSV file

I found it useful for tests to be able to load a CSV file to drive a test by inputting repeatable data over and over. This is similar to using a data provider function.

```
private function readCsv2(): array {
  $file = getcwd() . '/modules/custom/tea_teks/modules/tea_teks_publisher/tests/ExistingSiteJavascript/test2.csv';
  $csv = array_map('str_getcsv', file($file));
  array_walk($csv, function(&$a) use ($csv) {
    $a = array_combine($csv[0], $a);
  });
  array_shift($csv); # remove column header

  /*
   * The above code produces arrays of values for the CSV file:
   * [0] =>[
        [Num] => 1
        [Program] => 852061
        [Expectation] => 852076
        [Correlation] => 852081
        [SKIP] => N
        [Romanette] => i
        [KSS-SE] => met
        [xofy] => 1 1 1 1
        [Citations] => SN, SA
        [BrkStatus] => complete
        [ExpecStatus] => unmet
     ]
   * Use this code to display it on screen:
   * echo '<pre>';
   * print_r($csv);
   * echo '</pre>';
   *
   */
  return $csv;
}
```

Here is the CSV file:

```
**Num**,**Program**,**Expectation**,**Correlation**,**SKIP**,**Romanette**,**KSS-SE**,**xofy**,**Citations**,**BrkStatus**,**ExpecStatus\
1**,**852061**,**852076**,**852081**,**N**,**i**,**1.A**,**\"1, 1, 1,
1\"**,**\"SN, SA\"**,**met**,**unmet\
2**,**852061**,**852076**,**852086**,**N**,**ii**,**1.A**,**\"1, 1, 1,
1\"**,**\"SN, SA, TN, TA\"**,**met**,**unmet\
3**,**852061**,**852076**,**852091**,**N**,**iii**,**1.A**,**\"0, 1, 0,
1\"**,**SA**,**unmet**,**unmet\
4**,**852061**,**852076**,**852096**,**N**,**iv**,**1.A**,**\"0, 1, 0,
1\"**,**SA**,**unmet**,**unmet\
5**,**852061**,**852161**,**852166**,**N**,**i**,**1.B**,**\"0, 0 ,1,
1\"**,**\"TN, TA\"**,**unmet**,**unmet\
6**,**852061**,**852161**,**852171**,**N**,**ii**,**1.B**,**\"0, 1, 1,
1\"**,**\"SA, SA, SA, SA, TN, TA\"**,**unmet**,**unmet\
7**,**852061**,**852101**,**852106**,**N**,**i**,**2.A**,,**\"TN, TN,
TN, TN, TA\"**,**unmet**,**unmet\
8**,**852061**,**852101**,**852111**,**N**,**ii**,**2.A**,,**\"SN, SA,
SA, TN, TN, TA, TA\"**,**met**,**unmet\
9**,**852061**,**852116**,**852121**,**N**,**i**,**2.B**,,**\"SN,
TA\"**,**unmet**,**unmet\
10**,**852061**,**852116**,**852126**,**N**,**ii**,**2.B**,,**\"SN,
SA\"**,**unmet**,**unmet\
11**,**852061**,**852131**,**852136**,**N**,**i**,**3.A**,,**\"SN, SN,
SN, SN, SA, SA, SA, SA\"**,**met**,**met\
12**,**852061**,**852131**,**852141**,**N**,**ii**,**3.A**,,**TN**,**met**,**met\
13**,**852061**,**852146**,**852151**,**N**,**i**,**3.B**,,**TA**,**met**,**met\
14**,**852061**,**852146**,**852156**,**N**,**ii**,**3.B**,,**\"TN,
TA\"**,**met**,**met**
```

Picture of CSV file with color formatting:

```
Num,Program,Expectation,Correlation,SKIP,Romanette,KSS-SE,xofy,Citations,BrkStatus,ExpecStatus
1,852061,852076,852081,N,i,1.A,"1, 1, 1, 1","SN, SA",met,unmet
2,852061,852076,852086,N,ii,1.A,"1, 1, 1, 1","SN, SA, TN, TA",met,unmet
3,852061,852076,852091,N,iii,1.A,"0, 1, 0, 1",SA,unmet,unmet
4,852061,852076,852096,N,iv,1.A,"0, 1, 0, 1",SA,unmet,unmet
5,852061,852161,852166,N,i,1.B,"0, 0 ,1, 1","TN, TA",unmet,unmet
6,852061,852161,852171,N,ii,1.B,"0, 1, 1, 1","SA, SA, SA, SA, TN, TA",unmet,unmet
7,852061,852101,852106,N,i,2.A,,"TN, TN, TN, TN, TA",unmet,unmet
8,852061,852101,852111,N,ii,2.A,,"SN, SA, SA, TN, TN, TA, TA",met,unmet
9,852061,852116,852121,N,i,2.B,,"SN, TA",unmet,unmet
10,852061,852116,852126,N,ii,2.B,,"SN, SA",unmet,unmet
11,852061,852131,852136,N,i,3.A,,"SN, SN, SN, SN, SA, SA, SA, SA",met,met
12,852061,852131,852141,N,ii,3.A,,TN,met,met
13,852061,852146,852151,N,i,3.B,,TA,met,met
14,852061,852146,852156,N,ii,3.B,,"TN, TA",met,met
```

Adding DTT to an existing site

Install DTT and dev requirements with:

Follow these steps to quickly get DTT running on your project.

```
$ composer require weitzman/drupal-test-traits --dev

$ composer require drupal/core-dev --dev --update-with-all-dependencies
```

Setup `phpunit.xml` in the root of the project (not docroot or web). There will usually be a `phpunit.xml.dist` file there. Use that file and add your tweaks to it using https://gitlab.com/weitzman/drupal-test-traits/-/blob/master/docs/phpunit.xml as the basis.

Create phpunit.xml file

My `phpunit.xml.dist`:

```
<?xml version="1.0" encoding="UTF-8"?>

<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="vendor/autoload.php"
         verbose="true"
        >
    <testsuites>
        <testsuite name="drupal-composer-project tests">
            <directory>./test/</directory>
        </testsuite>
    </testsuites>
</phpunit>
```

My phpunit.xml with edits for site tea3.ddev.site. replace tea3 with the sitename for your ddev site in the `<env name="DTT_BASE_URL" value="http://tea3.ddev.site"/>`.

```
<?xml version="1.0" encoding="UTF-8"?>

<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="scripts/bootstrap-fast.php"
         verbose="true"
        >
  <php>
    <env name="DTT_BASE_URL" value="http://tea3.ddev.site"/>
    <env name="DTT_API_URL" value="http://chrome:9222"/>
    <env name="DTT_MINK_DRIVER_ARGS" value='["chrome", {"browserName":"chrome","chromeOptions":{"args":["--disable-gpu","--headless", "--no-sandbox"]}}, "http://chrom
    <env name="DTT_API_OPTIONS" value='{"socketTimeout": 360, "domWaitTimeout": 3600000}' />
    <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /tmp
         Specify a temporary directory for storing debug images and html documents.
         These artifacts get copied to /sites/simpletest/browser_output by BrowserTestBase. -->
    <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/tmp"/>
    <!-- To disable deprecation testing completely uncomment the next line. -->
    <env name="SYMFONY_DEPRECATIONS_HELPER" value="disabled"/>
    <!-- Specify the default directory screenshots should be placed. -->
    <!--<env name="DTT_SCREENSHOT_REPORT_DIRECTORY" value=""/>-->
    <!-- Specify the default directory page captures should be placed.
        When using the \Drupal\Tests\Listeners\HtmlOutputPrinter printerClass this will default to
        /sites/simpletest/browser_output. If using another printer such as teamcity this must be defined.
        -->
    <!--<env name="DTT_HTML_OUTPUT_DIRECTORY" value=""/>-->
  </php>
  <testsuites>
    <testsuite name="drupal-composer-project tests">
      <directory>./test/</directory>
    </testsuite>
    <testsuite name="unit">
      <directory>./web/modules/custom/*/tests/src/Unit</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/Unit</directory>-->
    </testsuite>
    <testsuite name="kernel">
      <directory>./web/modules/custom/*/tests/src/Kernel</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/Kernel</directory>-->
    </testsuite>
    <testsuite name="existing-site">
      <!-- Assumes tests are namespaced as \Drupal\Tests\custom_foo\ExistingSite. -->
      <directory>./web/modules/custom/*/tests/src/ExistingSite</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/ExistingSite</directory>-->
    </testsuite>
    <testsuite name="existing-site-javascript">
      <!-- Assumes tests are namespaced as \Drupal\Tests\custom_foo\ExistingSiteJavascript. -->
      <directory>./web/modules/custom/*/tests/src/ExistingSiteJavascript</directory>
      <!--<directory>./web/profiles/custom/*/tests/src/ExistingSiteJavascript</directory>-->
    </testsuite>
  </testsuites>
</phpunit>
```

Create bootstrap-fast.php

Create /scripts/bootstrap-fast.php with the following contents

```php
<?php
/**
 * @file
 *   A bootstrap file for `phpunit` test runner.
 *
 * This bootstrap file from DTT is fast and customizable.
 *
 * If you get 'class not found' errors while running tests, you should copy this
 * file to a location inside your code-base --such as `/scripts`. Then add the
 * missing namespaces to the bottom of the copied field. Specify your custom
 * `bootstrap-fast.php` file as the bootstrap in `phpunit.xml`.
 *
 * Alternatively, use the bootstrap.php file, in this same directory, which is
 * slower but registers all the namespaces that Drupal tests expect.
 */

use Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter;
use weitzman\DrupalTestTraits\AddPsr4;

list($finder, $class_loader) = AddPsr4::add();
$root = $finder->getDrupalRoot();

// So that test cases may be simultaneously compatible with multiple major versions of PHPUnit.
$class_loader->addPsr4('Drupal\TestTools\\', "$root/core/tests");
if (class_exists('Drupal\TestTools\PhpUnitCompatibility\PhpUnit8\ClassWriter')) {
  ClassWriter::mutateTestBase($class_loader);
}

// Register more namespaces, as needed.
# $class_loader->addPsr4('Drupal\Tests\my_module\\', "$root/modules/custom/my_module/tests/src");
```

Add .phpunit.result.cache file to .gitignore

To stop result cache getting checked into the repo, add the .phpunit.result.cache to the .gitignore file.

You could also change this file location by editing phpunit.xml:

```
<phpunit
  ...
  cacheResultFile="../.temp/fs_cache/.phpunit.result.cache"
>
```

Or completely disable it by:

```
<phpunit
  ...
  cacheResult ="false"
>
```

Remove DTT and core-dev

For production deployment, you can remove DTT and core-dev with:

```
$ composer remove drupal/core-dev --dev

$ composer remove weitzman/drupal-test-traits --dev
```

Alternatively, just run

```
$ composer update --no-dev
```

Hide deprecation notices

To hide deprecation notices when running test on the host, update your php.ini (run `php --ini` to find the php.ini file) and change the error_reporting line from:

```
error_reporting = E_ALL
```

to:

```
error_reporting = E_ALL & ~E_DEPRECATE
```

Now tests should look like this:

> $ vendor/bin/phpunit docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/VotingPageTest.php
>
> PHPUnit 9.5.24 #StandWithUkraine
>
> Runtime: PHP 8.1.9
>
> Configuration: /Users/selwyn/Sites/tea/phpunit.xml
>
> . 1 / 1 (100%)
>
> Time: 00:00.830, Memory: 46.50 MB
>
> OK (1 test, 3 assertions)

Troubleshooting DTT Tests

Which test?

For tests that involve the Drupal API, if a test fails, you might see output like this:

```
1) Drupal\Tests\tea_teks_requirements\ExistingSite\Requirements1Test::testEmptyRequirements
ArgumentCountError: Too few arguments to function Drupal\Core\Entity\EntityBase::load(), 0 passed in /var/www/html/web/modules/custom/tea_teks_requirements/tests/src/Exis
```

This is indicating the first test by: “1)”. If this were the second test in the file, it would show “2)”. The error is that too few arguments were passed to EntityBase::load() – in my case, I was passing null to a Node::load() function.

PolyfillAssertTrait not found

After installing DTT you see errors when you try to run the tests like this:

> selwyn@tea3-web:/var/www/html$ ./vendor/bin/phpunit --bootstrap=./vendor/weitzman/drupal-test-traits/src/bootstrap-fast.php ./docroot/modules/custom/tea_teks/modules/tea_teks_requirements/tests/src/ExistingSite/RequirementsCreationTest.php
>
> PHP Fatal error: Trait "Symfony\Bridge\PhpUnit\Legacy\PolyfillAssertTrait" not found in /var/www/html/docroot/sites/simpletest/Assert.php on line 91
>
> Fatal error: Trait "Symfony\Bridge\PhpUnit\Legacy\PolyfillAssertTrait" not found in /var/www/html/docroot/sites/simpletest/Assert.php on line 91

You will need to install the dev requirements with:

```
$ composer require drupal/core-dev --dev
--update-with-all-dependencies
```

Class not found errors

If you see something like this on your brand new class you created:

```
1) Drupal\Tests\tea_teks_voting\ExistingSite\ProgramTest::testIsVotingPermitted
Error: Class "Drupal\tea_teks_voting\Program" not found
```

This is a real forehead slapper! Be sure to enable your custom module under Drupal's extend menu option.

Also

When running the tests, if they start throwing "class not found" errors, this may indicate some outdated code in your codebase.

> $ vendor/bin/phpunit docroot/modules/custom/tea_teks/modules/tea_teks_voting/Tests/src/ExistingSite/RequirementsCreationTest.php
>
> PHP Fatal error: Uncaught Error: Class "PHPUnit\TextUI\Command" not found in /var/www/html/vendor/phpunit/phpunit/phpunit:98
>
> Stack trace:
>
> #0 /var/www/html/vendor/bin/phpunit(123): include()
>
> #1 {main}
>
> thrown in /var/www/html/vendor/phpunit/phpunit/phpunit on line 98
>
> Fatal error: Uncaught Error: Class "PHPUnit\TextUI\Command" not found in /var/www/html/vendor/phpunit/phpunit/phpunit:98
>
> Stack trace:
>
> #0 /var/www/html/vendor/bin/phpunit(123): include()
>
> #1 {main}
>
> thrown in /var/www/html/vendor/phpunit/phpunit/phpunit on line 98

The fix in this case was a composer update.

Also If you use another test as a starting point (ie. Copy the file) and forget to change the class name, that would cause a similar error:

```
$ vendor/bin/phpunit docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/PersonaTest.php

Class 'PersonaTest' could not be found in '/Users/selwyn/Sites/tea/docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/PersonaTest.php'.
```

In this example, the filename was PersonaTest.php but the class name is accidentally called TeamTest so the interpreter could not find a PersonaTest. Oops. Here is the errant `PersonaTest.php` file:

```php
<?php

namespace Drupal\Tests\tea_teks_voting\ExistingSite;

use Drupal\node\Entity\Node;
use Drupal\tea_teks_voting\Persona;
use weitzman\DrupalTestTraits\ExistingSiteBase;

class TeamTest extends ExistingSiteBase {

  public function testPersonaEmptyLoading() {
    $p = new Persona();
    $user_id = $p->getUserId();
    self::assertEquals(0, $user_id);
  }

}
```

var_dump, echo, print

For quick variable dumps, it is quite valid to use var_dump in your tests. Here is a test with a var_dump() call.

You can also print or echo variables e.g.

```
// \n will put this output on a new line.
$temp = "blah";
echo "\nResults = $temp";
print "\nResults = $temp";
```

```
<?php

namespace Drupal\Tests\tea_teks_voting\ExistingSite;

use Drupal\node\Entity\Node;
use Drupal\tea_teks_voting\Team;
use weitzman\DrupalTestTraits\ExistingSiteBase;

class TeamTest extends ExistingSiteBase {

  public function testTeamLoading() {
    // Team D - Voter 1, 2, 3
    // 5101, 5106, 5116
    $t = new Team(868296);
    $team_member_info = $t->getTeamMemberInfo(5101);
    var_dump($team_member_info);
    $this->assertSame($team_member_info['name'], 'Voter 1');
    $this->assertSame($team_member_info['fullname'], 'Voter 1 Test');
    $this->assertSame($team_member_info['mail'], 'voter1@mightycitizen.com');
    $this->assertSame($team_member_info['roles'][0], 'authenticated');
    $this->assertSame($team_member_info['roles'][1], 'srp_voter');
  }
}
```

Here is the output. Note. I removed the deprecated messages for clarity. First the command to run the test:

```
$ vendor/bin/phpunit docroot/modules/custom/tea_teks/modules/tea_teks_voting/tests/src/ExistingSite/TeamTest.php
```

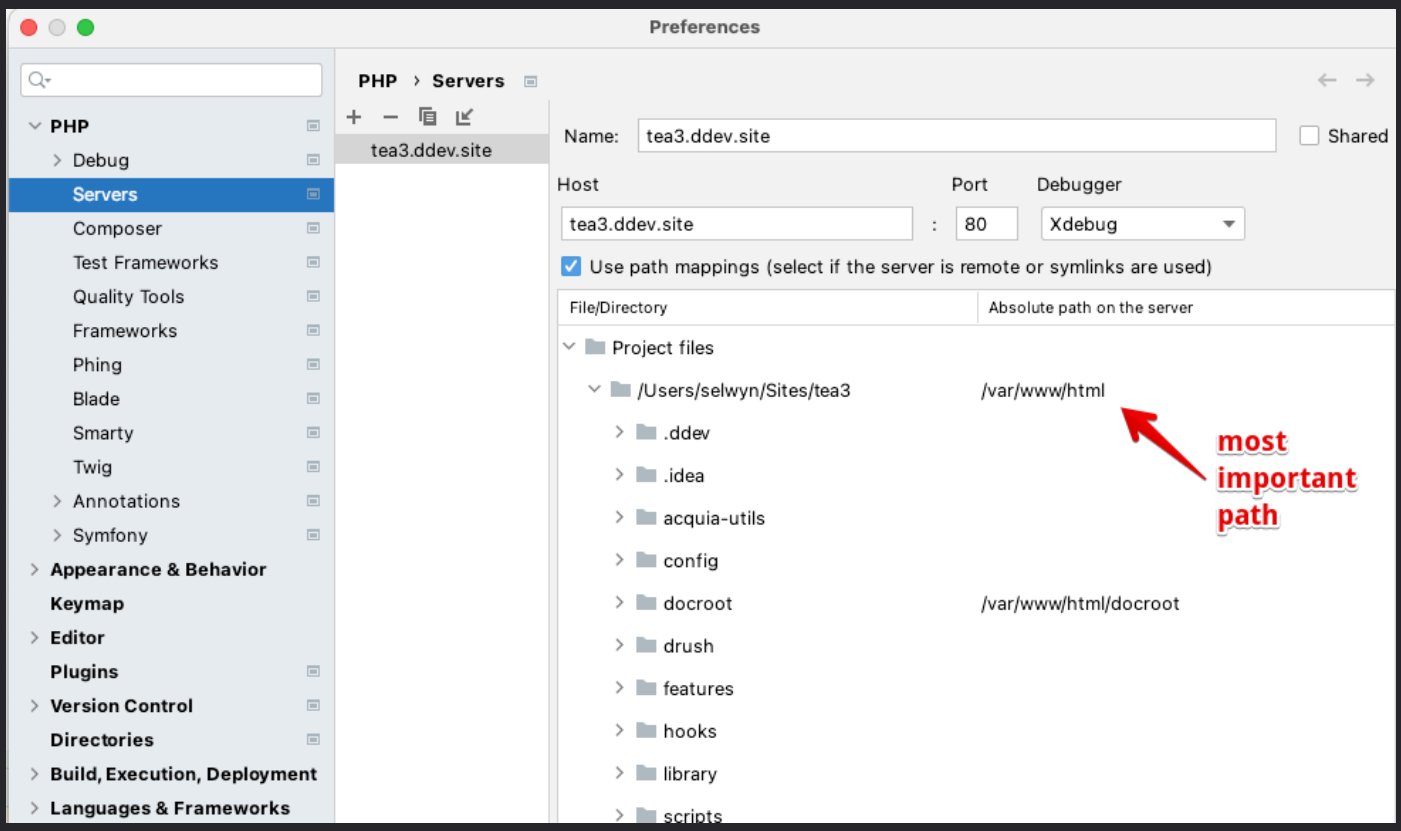
and the output:

> PHPUnit 9.5.24 #StandWithUkraine
>
> Runtime: PHP 8.1.9
>
> Configuration: /Users/selwyn/Sites/tea/phpunit.xml
>
> .
>
> 1 / 1 (100%)
>
> array(11) {
>
> ["uid"]=> string(4) "5101"
>
> ["fullname"]=> string(12) "Voter 1 Test"
>
> ["name"]=> string(7) "Voter 1"
>
> ["mail"]=> string(24) "voter1@mightycitizen.com"
>
> ["status"]=> string(1) "1"
>
> ["firstname"]=> string(7) "Voter 1"

```
["lastname"]=> string(4) "Test"
["title"]=> NULL
["phone"]=> NULL
["roles"]=> array(2) {
[0]=> string(13) "authenticated"
[1]=> string(9) "srp_voter"
}
["voter_role"]=> string(8) "educator"
}
Time: 00:00.773, Memory: 44.50 MB
OK (1 test, 5 assertions)
```

Using Xdebug and PHPStorm to debug DTT scripts

It is easiest to make sure you have PHPStorm Xdebug working first, then make sure the path mappings are correct. Note. This process is almost identical to debugging drush commands.




```
$ ddev exec enable_xdebug

$ ddev ssh
```

Note. Servername has to match the servername in your phpstorm setup on a per project – see screenshot below.

Sometimes this step doesn't seem to be required. Not sure why.

```
$ export PHP_IDE_CONFIG=\"serverName=tea.ddev.site\"
```

(the tea part above needs to match your ddev project name. e.g. drupal.ddev.site or selwyn.ddev.site etc.)

click listen in PHPStorm

click on the line number in PHPStorm to set a breakpoint

Issue the phpunit command in the vendor directory:

```
$ vendor/bin/phpunit
docroot/modules/custom/tea_teks/modules/tea_teks_requirements/Tests/src/ExistingSite/RequirementsCreationTest.php
```

When Phpstorm pops up, specify that the vendor directory is at `/var/www/html/vendor` - note you only have to do that once and then PHPStorm will remember it.

Resources

General

- Video intro to DTT with Moshe Weitzman from DrupalCon Global Sep 2020 https://www.youtube.com/watch?v=TExPFQ1-AA0&t=10s
- Unit Tests in Drupal: the road to test-driven development video from DrupalCon Global 2020 by CivicActions folks: Gerardo Gonzalez and Eric Napier. This gets into the details but is hard to make out the code as it is small. https://www.youtube.com/watch?v=UGg3G0dsAGw
- Understanding Automated Tests in Drupal video from DrupalGov 2020 by Ridhima Abrol & Sujeet Kumar Verma. They are showing examples using MAMP. This covers unit vs kernel vs functional vs functional JS. There are fairly legible code examples. https://www.youtube.com/watch?v=kQEMnk4boP4&list=TLPQMTcwODIwMjKbpqkPMTkhyA&index=4
- How to unit test your code in Drupal 8 video by Daniel Nitsche for DrupalSouth 2017 1-10-2017 (Loft??). Australian chap (very hands-on) going through live demo on PHPStorm running phpunit on Drupal 8. Using mockery. https://www.youtube.com/watch?v=7FjjZ3OoD6Y
- Running and debugging PHPUnit tests in PHPStorm with ddev and xdebug video by Australian Michael Strelan on 8-18-21. The audio is very soft. He walks through details of setting up PHPStorm for DDEV etc. https://www.youtube.com/watch?v=OdoEyY8Kl9w . There is also a companion article from Michael Strelan at https://www.previousnext.com.au/blog/running-and-debugging-phpunit-tests-phpstorm-ddev-and-xdebug and a repo: https://github.com/mstrelan/ddev-phpunit-demo. It is useful to mention that since this video was created there is a DDEV integration plugin for PHPStorm which automatically configures things like path mappings, cli interpreters and phpunit configuration. This makes some of what he describes a lot easier.
- From a discord chat with Randy Fey and \@shaal. This may be useful to explore phpunit tests on tugboat when we get that all set up. Here's a good PR where we added phpunit into DrupalPod https://github.com/shaal/DrupalPod/pull/41/files
- Benji Fisher Drupal Testing repo from February 2022. Benji set up this repository to help with testing Drupal modules for coding standards and Drupal 10 compatibility.It is based on drupal/recommended-project with some parts borrowed from Matt Glaman's Drupal & Nightwatch.js training . Main features: DDev configuration (mostly standard), upgrade Status module, custom DDev commands phpunit, phpcs, phpcbf, Docker config to support PHPUnit testing. - https://github.com/benjifisher/drupal-testing
- The abovementioned Matt Glaman's repo from March 2021 on Drupal & Nightwatch.js training. This repository is based on a Composer build and not meant for core contributions, it is fine for contrib. - https://github.com/bluehorndigital/drupal-testing-workshop
- And its companion website with some details on how to get tests running: https://bluehorndigital.github.io/drupal-testing-workshop/getting-tests-running/ddev.html
- Generating tests with drush: https://www.drush.org/latest/generators/

Documentation

- Mink documentation: https://mink.behat.org/en/latest/

Testing setup

- Debug any of Drupal's PHPUnit tests in PhpStorm with a DDEV-Local Environment from August 3, 2021 by Joe Shindelar. Includes how to set up Chromedriver in DDEV: https://drupalize.me/blog/debug-any-drupals-phpunit-tests-phpstorm-ddev-local-environment

- Joe references this article from 5-13-2021 called Update 2021 - Fully integrate DDEV and PHPStorm - including Unit Tests with Coverage by \@sasunegomo https://susi.dev/fully-integrate-ddev-and-phpstorm-including-unit-tests-with-coverage-update-2021/
- Setup Behat for Drupal 8/9 with DDEV-Local and Selenium Recipe on Github as part of the ddev-contrib repo from July 2021 by Mike Miles: https://github.com/drud/ddev-contrib/tree/master/docker-compose-services/drupal8-behat-selenium
- Randy Fay lays out some details about running Selenium/Behat inside DDEV containers in August 2020 https://stackoverflow.com/questions/51527663/running-selenium-tests-using-behat-drupal-extension-inside-ddev-containers
- Running and debugging PHPUnit tests in PHPStorm with DDev and xdebug from 8-19-21 by Michael Strelan of Australia which includes 15 minute video showing all this. You can also checkout the ddev-phpunit-demo repo if you want to try it out yourself with DDev and PHPUnit pre-configured. https://www.previousnext.com.au/blog/running-and-debugging-phpunit-tests-phpstorm-ddev-and-xdebug
- Matt Glaman's Guide to Test-Driven Development with DDEV and Drupal by Heather McNamee 1-30-2019. She runs through the series of 2018 articles that Matt published
  - **Part 1. Running Drupal's PHPUnit test suites on DDEV.** How to execute PHPUnit from within the web container in DDEV for Unit, Kernel, and Functional tests.
  - **Part 2. Running Drupal's FunctionalJavascript tests on DDEV.** How to Chromedriver running to execute the FunctionalJavascript test suite.
  - **Part 3. Running Drupal's Nightwatch test suite on DDEV.** How to run Drupal's newest testing framework: Nightwatch.js, for end-to-end tests in Node.js run against a Selenium/WebDriver server.
- Running Drupal's PHPUnit test suites on DDEV by Matt Glaman from October 2018. The first in his series referenced above https://glamanate.com/blog/running-drupals-phpunit-test-suites-ddev
- DDEV Contrib Repo https://github.com/drud/ddev-contrib

Mocking

- From Matthew Radcliffe mradcliffe \@mattkineme – Mocking Drupal: Unit Testing in Drupal 8 slides from a 2015 presentation at http://drupalcampohio.org/sites/default/files/slides/dco2015-mocking-drupal.pdf
- Drupal 8/9: Unit Test cases mocking the global Drupal object and Services by Vishwa Chikate - Talks about prophecy objecting mocking which is better than PHPunit's built in mocking. https://medium.com/@vishwa.chikate/drupal-8-9-unit-test-cases-mocking-the-global-drupal-object-and-services-bc536477edff

---

Back to top

Page last modified: Jun 27 2023.

Edit this page on GitHub

This site uses Just the Docs, a documentation theme for Jekyll.