PHPUnit helper doc

The intention of this doc is to guide the developer, on how to start writing the PhpUnit test cases for the modules they are working on. This doc contains the basic information like:

Setup

1. Install PHPUnit, command (`composer require drupal/core-dev --dev --update-with-all-dependencies`)
2. Copy and rename `docroot/core/phpunit.xml.dist` to `docroot/core/phpunit.xml`
3. Update `phpunit.xml` file with below
   Update `SIMPLETEST_BASE_URL, SIMPLETEST_DB` variable values.

```
<phpunit bootstrap="tests/bootstrap.php" colors="true"
         beStrictAboutTestsThatDoNotTestAnything="true"
         beStrictAboutOutputDuringTests="true"
         beStrictAboutChangesToGlobalState="true"
         printerClass="\Drupal\Tests\Listeners\HtmlOutputPrinter">
  <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"/>
    <!-- Example SIMPLETEST_BASE_URL value: http://localhost -->
    <env name="SIMPLETEST_BASE_URL" value="https://ads-management.ddev.site"/>
    <!-- Example SIMPLETEST_DB value: mysql://username:password@localhost/databasename#table_prefix -->
    <env name="SIMPLETEST_DB" value="mysql://db:db@localhost/db"/>
    <!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /path/to/webroot/sites/simpletest/browser_output -->
    <env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/var/www/html/docroot/sites/simpletest/browser_output"/>
  </php>
</phpunit>
```

4. Or follow Running PHPUnit tests within PhpStorm instructions to setup with PHPStorm

File structure

```
../modules/custom
--stuff_utility
---src
----StuffUtilityService.php

--stuff_ad_exclusion
----stuff_ad_exclusion.info.yml
----stuff_ad_exclusion.module
----src
------Unit.php
----tests
------src
--------Unit
----------AdExclusionUnitTest.php
```

Sample code

`modules/custom/stuff_utility/src/StuffUtilityService.php`

```php
<?php

namespace Drupal\stuff_utility;

use Drupal\Component\Plugin\Exception\InvalidPluginDefinitionException;
use Drupal\Component\Plugin\Exception\PluginNotFoundException;
use Drupal\Core\Access\AccessResult;
use Drupal\Core\Entity\EntityInterface;
use Drupal\Core\Entity\EntityTypeManagerInterface;
use Drupal\Core\Session\AccountInterface;
use Drupal\Core\Site\Settings;
use GuzzleHttp\ClientInterface;
use GuzzleHttp\Exception\GuzzleException;

/**
 * Provides common functions at one place.
 */
class StuffUtilityService {

  /**
   * The entity type manager service.
   *
   * @var \Drupal\Core\Entity\EntityTypeManagerInterface
   */
  protected $entityTypeManager;

  /**
   * The HTTP client to fetch the feed data with.
   *
   * @var \GuzzleHttp\ClientInterface
   */
  protected $httpClient;

  /**
   * Constructs a StuffUtilityService object.
   *
   * @param \Drupal\Core\Entity\EntityTypeManagerInterface $entity_type_manager
   *   The entity type manager service.
   * @param \GuzzleHttp\ClientInterface $http_client
   *   The http client service.
   */
  public function __construct(EntityTypeManagerInterface $entity_type_manager, ClientInterface $http_client) {
    $this->entityTypeManager = $entity_type_manager;
    $this->httpClient = $http_client;
  }

  /**
   * Validate the article asset_id from Api.
   *
   * @param string $type
   *   Type of page (i.e. article)
   * @param string $uuid
   *   Uuid of article.
   * @param string $assetId
   *   Asset id of article.
```

```
   * @param bool $result
   *   Whether data return or not.
   *
   * @return bool|string|null
   *   Return the data.
   */
  public function validateAssetId(string $type, string $uuid, string $assetId, bool $result = FALSE) {
    $key = 'stuff_asset_id';
    $url_entity = 'node/' . $type;
    if ($type === 'video') {
      $key = 'video_id';
      $url_entity = 'brightcove_video/brightcove_video';
    }
    $url = Settings::get('platform_api_base_url') . $url_entity . '/' . $uuid;
    $response = $this->handlePlatformApiRequest($url);
    $res = !empty($response) && $response['attributes'][$key] === $assetId;
    if ($res && $result) {
      $res = $this->buildAutocompleteString($response, $type);
    }
    elseif (!$res && $result) {
      $res = 'Broken/invalid link to ' . $type;
    }
    return $res;
  }

  /**
   * Handle the request of platform Api.
   *
   * @param string $url
   *   Url to fetch data including filter.
   *
   * @return false|mixed|void
   *   Return the response.
   */
  public function handlePlatformApiRequest(string $url) {
    try {
      $options = [
        'headers' => [
          'Accept' => 'application/vnd.api+json',
          'Authorization' => 'Basic ' . base64_encode(Settings::get('platform_api_user') . ':' . Settings::get(
        ],
      ];
      $response = $this->httpClient->request('GET', $url, $options);
      if ($response->getStatusCode() === 200) {
        $result = json_decode($response->getBody(), TRUE);
        return $result['data'];
      }
    }
    catch (GuzzleException $exception) {
      return FALSE;
    }
  }

}

```

modules/custom/stuff_ad_exclusion/tests/src/Unit/AdExclusionUnitTest.php

```
<?php
```

```
2
3  namespace Drupal\Tests\stuff_ad_exclusion\Unit;
4
5  use Drupal\Core\Entity\EntityTypeManagerInterface;
6  use Drupal\Core\Site\Settings;
7  use Drupal\stuff_utility\StuffUtilityService;
8  use Drupal\Tests\UnitTestCase;
9  use GuzzleHttp\ClientInterface;
10 use GuzzleHttp\Psr7\Response;
11 use Prophecy\Argument;
12
13 /**
14  * Test generation of Ad exclusion.
15  *
16  * @group ad_exclusion
17  */
18 class AdExclusionUnitTest extends UnitTestCase {
19
20   /**
21    * The entity type manager service.
22    *
23    * @var \Drupal\Core\Entity\EntityTypeManagerInterface
24    */
25   protected $entityTypeManager;
26
27   /**
28    * The HTTP client to fetch the feed data with.
29    *
30    * @var \GuzzleHttp\ClientInterface
31    */
32   protected $httpClient;
33
34   /**
35    * The utility service.
36    *
37    * @var \Drupal\stuff_utility\StuffUtilityService
38    */
39   protected $utilityService;
40
41   /**
42    * {@inheritdoc}
43    */
44   public function setUp() : void {
45     parent::setUp();
46     $this->entityTypeManager = $this->prophesize(EntityTypeManagerInterface::class);
47     $this->httpClient = $this->prophesize(ClientInterface::class);
48     $this->utilityService = new StuffUtilityService($this->entityTypeManager->reveal(), $this->httpClient->reve
49   }
50
51   /**
52    * Check the service is availability.
53    */
54   public function testStuffUtilityService() {
55     $this->assertNotNull($this->utilityService);
56   }
57
58   /**
59    * Check article asset_id is matched.
```

```
   *
   * @dataProvider providerTestAssetVideoId
   */
  public function testAssetVideoId($type, $uuid, $asset) {
    $key = $type === 'video' ? 'video_id' : 'stuff_asset_id';
    $bundle = $type === 'video' ? 'brightcove_video/brightcove_video' : 'node/' . $type;
    $url = Settings::get('platform_api_base_url') . $bundle . '/' . $uuid;
    $body = [
      'data' => [
        'id' => $uuid,
        'attributes' => [
          $key => $asset,
        ],
      ],
    ];
    $this->httpClient->request('GET', $url, Argument::any())->willReturn(new Response(200, [], json_encode($bod
    $returnBoolean = $this->utilityService->validateAssetId($type, $uuid, $asset);
    $this->assertTrue($returnBoolean);
  }

  /**
   * Provide test cases for ::testAssetVideoId().
   *
   * @return array
   *   An array of tested data.
   */
  public function providerTestAssetVideoId() {
    $data[] = [
      'article', 'd737c199-7837-404d-8fc8-10d5bd9f0ffe', '350002529',
    ];

    $data[] = [
      'video', 'd9a9cee6-9865-4f18-a6f3-abc305f139f8', '5165982271001',
    ];

    return $data;
  }

  /**
   * Test platform api return result data.
   */
  public function testRequestApi() {
    $url = Settings::get('platform_api_base_url') . 'node/article';
    $this->httpClient->request('GET', $url, Argument::any())->will(function () {
      $body = [
        'data' => [
          'id' => 'd737c199-7837-404d-8fc8-10d5bd9f0ffe',
          'attributes' => [
            'status' => TRUE,
            'title' => 'Article Timestamp 5.10',
            'stuff_asset_id' => '350002529',
          ],
        ],
      ];
      return new Response(200, [], json_encode($body));
    });

    $returnArray = $this->utilityService->handlePlatformApiRequest($url);
```

```
118     $this->assertIsArray($returnArray);
119   }
120
121 }
```

Commands

- Normal execution

```
cd core
../vendor/bin/phpunit docroot/modules/custom/stuff_ad_exclusion/tests/src/Unit/AdExclusionUnitTest.php
```

- Using `ddev`:
  - ```
    ddev ssh
    cd docroot
    ../vendor/bin/phpunit docroot/modules/custom/stuff_ad_exclusion
    ```
  - ```
    ddev exec ./vendor/bin/phpunit -c docroot/core docroot/modules/custom/stuff_ad_exclusion
    ```

References

1. PHPUnit file structure, namespace, and required metadata
2. https://www.drupal.org/docs/automated-testing/phpunit-in-drupal/running-phpunit-tests
3. Running PHPUnit tests within PhpStorm
4. Mocking Entities and Services with PHPUnit and Mocks
5. Using Prophecy
6. GitHub - phpspec/prophecy-phpunit: Integrating Prophecy in PHPUnit test cases