



# DRUPAL CON VIENNA



26-29 SEPTEMBER 2017





# DRUPALCON VIENNA



## Test all the things!





Get productive with automated testing in Drupal 8

Sam Becker





# WHO AM I?

-  Sam152 on [drupal.org](https://drupal.org)
-  Back-end Drupal dev for PreviousNext
-  Core contributor
-  Author of 50+ contributed projects





# WHO ARE YOU?

Developers seeking information on the  
"big picture" in Drupal testing.

May not have written many tests before.



DRUPALCON



VIENNA

# First Principles



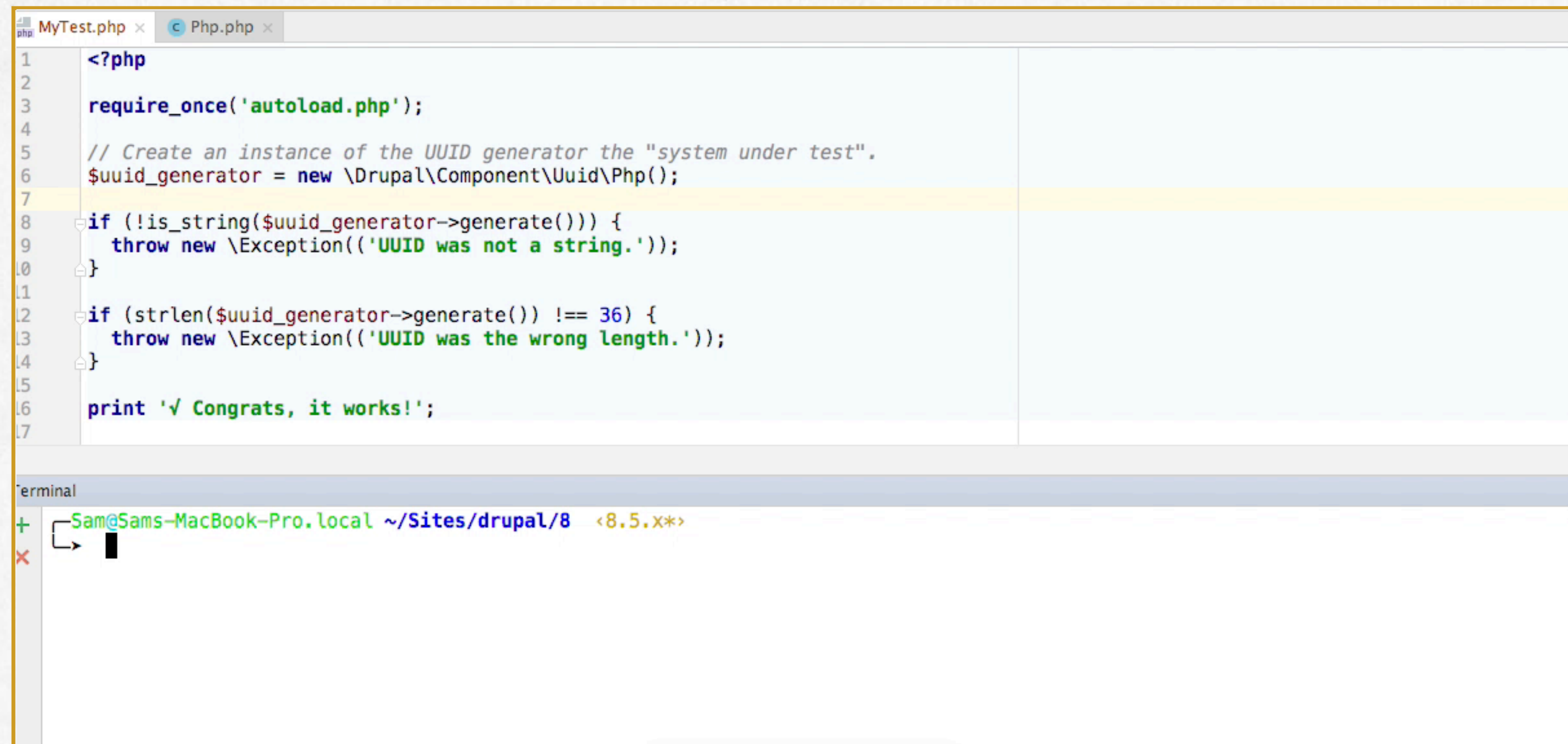


# What is a test?

- ✦ Code written to assert a series of constraints about some other bit of code.
- ✦ Essential for complex software.
- ✦ There are many forms of testing (we'll be covering this later).



# Simple Example






```
MyTest.php x  Php.php x
1  <?php
2
3  require_once('autoload.php');
4
5  // Create an instance of the UUID generator the "system under test".
6  $uuid_generator = new \Drupal\Component\Uuid\Php();
7
8  if (!is_string($uuid_generator->generate())) {
9      throw new \Exception('UUID was not a string.');
```

```
terminal
+ Sam@Sams-MacBook-Pro.local ~/Sites/drupal/8 <8.5.x*>
x
```

The UUID generator is our "system under test".



# Why?

-  Catch bugs.
-  Confidence to refactor.
-  Proof to reviewers that something works as advertised.





DRUPALCON



VIENNA

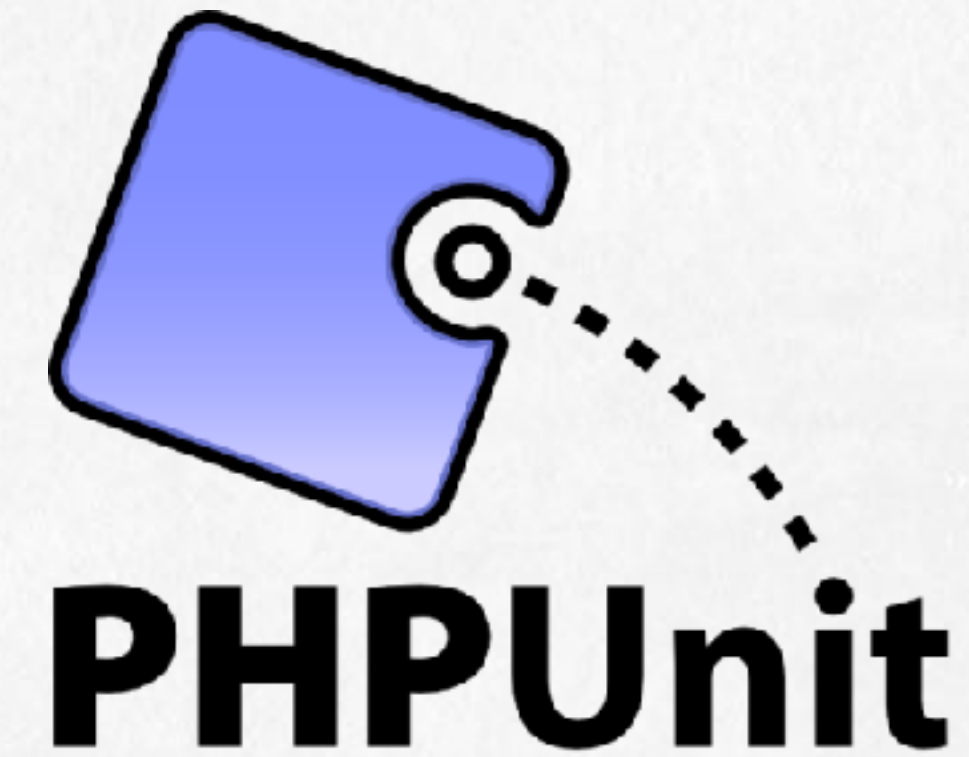
# PHPUnit





# PHPUnit

- Used for all testing in Drupal core.
- Trusted by many frameworks and projects in the PHP community.
- Lots of tools, extensions and support.



# PHPUnit Test Anatomy

```
1 <?php
2
3 namespace Drupal\Tests\workflows\Kernel;
4
5 use Drupal\KernelTests\KernelTestBase;
6
7 /**
8  * @coversDefaultClass \Drupal\workflows\EventSubscriber\ConfigImportSubscriber
9  * @group workflows
10 */
11 class ConfigImportSubscriberTest extends KernelTestBase {
12
13     /**
14      * {@inheritdoc}
15     */
16     public static $modules = [
17         'system',
18         'workflows',
19     ];
20
21     /**
22      * {@inheritdoc}
23     */
24     protected function setUp() {
25         parent::setUp();
26         // Setup the test.
27     }
28
29     /**
30      * @covers ::onConfigImporterValidate
31     */
32     public function testDeletingStates() {
33         // Test code here.
34     }
35
36 }
```

Annotation based metadata

Extends some core testing base class

Modules to enable

Sets up the preconditions for each test method.

Your test code goes in here.



# @dataProvider

- Use the same test method, but with different variables.
- Allows you to quickly add more test cases.
- Requires tests to be written in a more generic fashion.

```
10  /**
11   * @group video_embed_field
12   */
13  class ProviderUrlParseTest extends UnitTestCase {
14
15   /**
16    * Test URL parsing works as expected.
17    *
18    * @dataProvider urlsWithExpectedIds
19    */
20   public function testUrlParsing($provider, $url, $expected) {
21     $this->assertEquals($expected, $provider::getIdFromInput($url));
22   }
23
24   /**
25    * A data provider for URL parsing test cases.
26    *
27    * @return array
28    *   An array of test cases.
29    */
30   public function urlsWithExpectedIds() {
31     return [
32       // Youtube passing cases.
33       'YouTube: Standard URL' => [
34         'Drupal\video_embed_field\Plugin\video_embed_field\Provider\YouTube',
35         'https://www.youtube.com/watch?v=fdbFVWupSsw',
36         'fdbFVWupSsw',
37       ],
38       'YouTube: Non HTTPS' => [
39         'Drupal\video_embed_field\Plugin\video_embed_field\Provider\YouTube'
```

An example of the @dataProvider syntax



# @see Thursday

## PHPUnit for Drupal Developers

[sebastian\\_bergmann](#)

### SESSION TIME SLOT(S)

**Time:**

09/28/2017 - 14:15 to 09/28/2017 - 15:15

**Room:**

Lehar 3



DRUPALCON



VIENNA

# A Recent History



# Simpletest

- Tool introduced into Drupal 7.
- Did not gain widespread adoption.
- Still used to test D7 contrib.
  
- Superseded by PHPUnit in D8.
- Still simpletest tests in D8 core.

Home » Administration » Configuration » Development » Testing

## Test result

The test run finished in 39 sec.

**ACTIONS**

Filter Fal (1) [Run tests](#) [Return to list](#)

**RESULTS**

34 passes, 4 fails, 0 exceptions, and 13 debug messages

**WEBFORM SCHEDULED ANALYSIS DELIVERY TEST**

Test: the scheduled analysis delivery.  
34 passes, 4 fails, 0 exceptions, and 13 debug messages

| MESSAGE   | GROUP   | FILENAME                               |
|---|---------|--|
| Enabled modules: <i>webform, webform_scheduled_analysis</i>   | Other   | WebformScheduledAnalysisEmailDeliveryT |
| GET http://7.dev/cron.php?cron_key=yrrjPCizdGz_nRjEL5W35ZRXPDoqhQwBlgWojLZfK4 returned 200 (0 bytes). | Browser | WebformScheduledAnalysisEmailDeliveryT |
| Verbose message   | Debug   | WebformScheduledAnalysisEmailDeliveryT |

simpletest UI





# The Future!

- ✎ See the "phpunit initiative" tag in [drupal.org](https://drupal.org).
- ✎ Help port simpletests to PHPUnit.
- ✎ 300 tests left to port on last count.
  
- ✎ Porting these tests will mean consistency across core.







---

# High Level


## Unit Testing

- ✦ Tests one individual unit in isolation.
- ✦ Low level of abstraction, interacting directly with classes/functions.

## Integration Testing

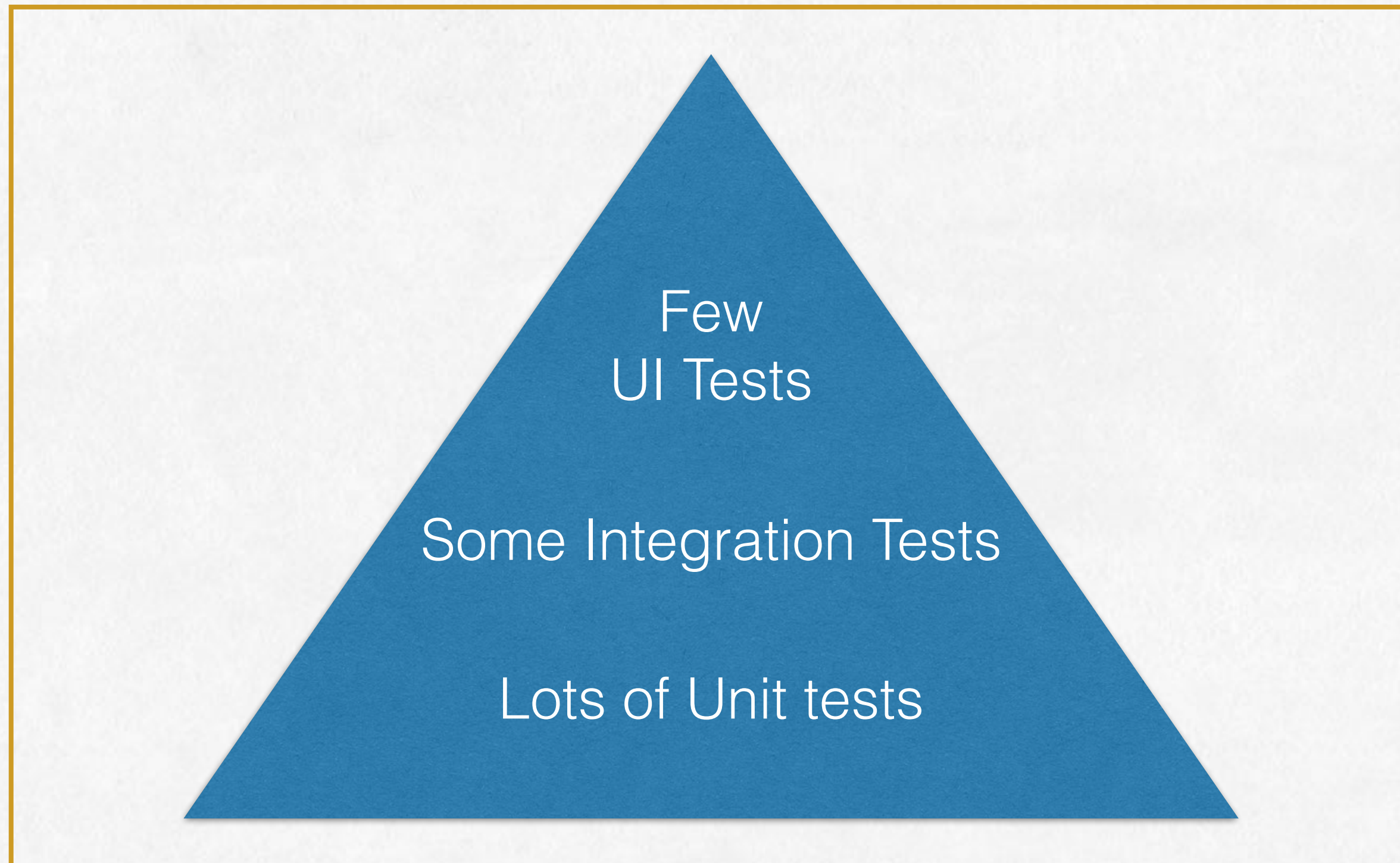
- ✦ Tests multiple units working together.
- ✦ Mid level of abstraction, working within a bootstrap.

## Functional (UI) Testing

- ✦ Tests system as a whole, like a user would.
  - ✦ High level of abstraction, knows nothing about the internals of the application.
- 



# The Pyramid



The testing pyramid

- Well tested applications have a mixture of these kinds of tests.





# An Example

**Scenario:** Are shipping prices correct.

## UI Test

- ✦ Create a user with access to checkout.
  - ✦ Create a product with weight 12KG.
  - ✦ Log in the test user.
  - ✦ Visit the product page and add to cart.
  - ✦ Visit the checkout.
  - ✦ Fill in address information and submit.
  - ✦ Assert the shipping page contains "\$14.95".
- 
- ✦ Runs in about 45 seconds.
  - ✦ Must be updated every time login, product pages and checkout changes.

## Unit Test

- ✦ Create an instance of ShippingPriceCalculator.
  - ✦ Verify return of "calculate" is 14.95 with postcode "6160" and "12KG".
- 
- ✦ Runs in 10ms.
  - ✦ Resilient to changes outside the unit.
  - ✦ Encourages testing many scenarios.





# A Counter Example

**Scenario:** Can the user checkout?

- ✦ Impossible to unit test, no single unit is responsible for checkout.
- ✦ Hugely valuable, we need to make sure this always works.



DRUPALCON



VIENNA

# JavaScriptTestBase





# JavaScriptTestBase

- ✦ Highest level of abstraction.
- ✦ UI testing with a browser, executing JavaScript.
- ✦ New capability in Drupal 8 testing.
- ✦ Very slow.



# Mink: Pluggable Drivers

```
7  /**
8  * JavaScript testing for the integration of the colorbox library.
9  *
10 * @group colorbox
11 */
12 class ColorboxJavascriptTest extends JavascriptTestBase {
13
14     /**
15     * {@inheritdoc}
16     */
17     public static $modules = [
18         'colorbox',
19         'colorbox_library_test',
20         'node',
21     ];
22
23     /**
24     * Test the colorbox launches when a gallery is clicked.
25     */
26     public function testColorboxLaunches() {
27         $this->drupalGet('node/1');
28         $this->getSession()->getPage()->find('css', 'img')->click();
29         $this->getSession()->wait(500);
30         $this->assertSession()->elementContains('css', '#colorbox', 'test.png');
31     }
32
33
34
```

Example of a JavaScript test

- Learn one API.
- Pick the tool for the job.
- Swap drivers without rewriting your test code.



# Getting Productive

- getSession() like "window" in JS.
- getSession()->getPage() like "document" in JS.
- assertSession()
- ...and a bunch of Drupal helpers:  
drupalPostForm(), drupalGet(), drupalLogin().

```
57  /**
58   * Test the colorbox gallery feature works.
59   */
60  public function testColorboxGallery() {
61      $this->drupalLogin($this->drupalCreateUser());
62
63      // Visit the test node and open the colorbox gallery.
64      $this->drupalGet('node/1');
65      $this->assertSession()->pageTextContains('Colorbox field');
66      $this->getSession()->getPage()->find('css', 'img')->click();
67
68      // Ensure the caption appears inside the colorbox.
69      $this->assertSession()->elementContains('css', '#cboxTitle', 'Image title 1');
70
71      // Clicking 'Next' should change the caption and the image to the second
72      // image.
73      $this->clickLink('Next');
74      $this->assertSession()->elementContains('css', '#cboxTitle', 'Image title 2');
75
76      // Closing the modal will hide the colorbox element.
77      $this->clickLink('Close');
78      $this->assertSession()->elementNotExists('css', '#colorbox');
79
80      // Ensure the image on the page is still visible.
81      $this->assertSession()->elementExists('css', 'img');
82
83
84
```

Mink in action





# Example Output

**Log in**

Username \*   
Enter your Drupal username.

Password \*   
Enter the password that accompanies your username.

Screenshots generated from a browser test





# Nondeterminism

Also known as the "random fail"

- Assume you need to wait for anything asynchronous.
- AJAX requests are the repeat offender.

```
$this->getSession()->wait(500, 'some js condition');  
$this->assertSession()->assertWaitOnAjaxRequest();  
$this->assertSession()->waitForElement('css', '.some-element');  
$this->assertSession()->waitForElementVisible('css', '.some-element');
```



DRUPALCON



VIENNA

# BrowserTestBase








# BrowserTestBase

- ✦ Same API as JavaScript test base.
  - ✦ Doesn't execute JavaScript during the test run.
  - ✦ Runs faster, less prone to random fails.
  - ✦ No external dependency on PhantomJS.
- 
- ✦ Should be used for all UI testing that doesn't require JS to execute.





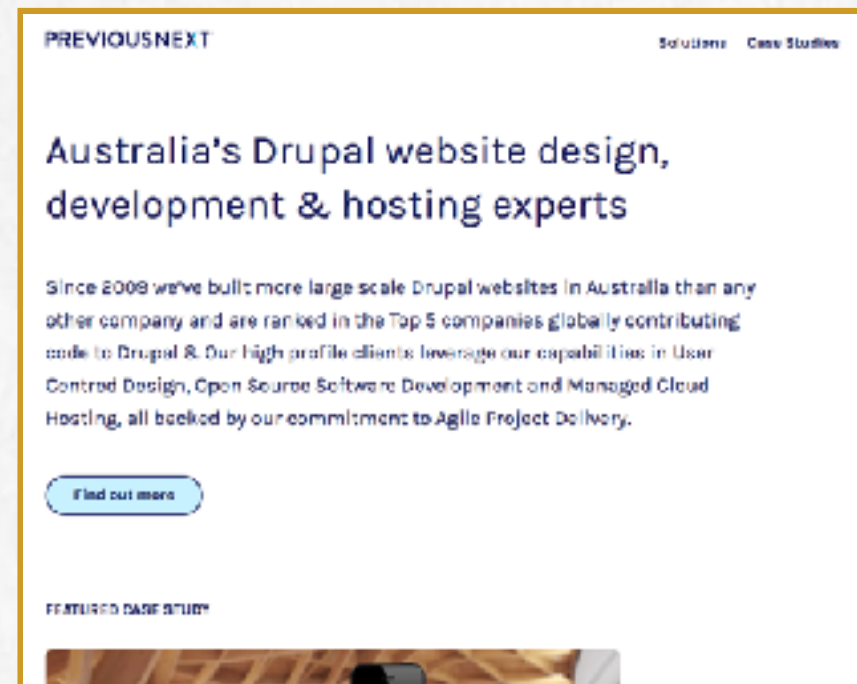
# setUp

-  BTB/JTB start from scratch.
-  Try to isolate tests, make them totally deterministic.
-  Other testing tools just point to an installed site with no fuss.



# Pre-provisioned Environments

DB Sync/  
Site Installation





BTB/JTB: Skipped setup  
for existing site

- Both BTB/JTB can be adapted to test pre-provisioned installed sites.
- Skip creating production-like conditions in setUp.
- Useful for testing heavily interdependent components:
  - Site building.
  - Complexities in a theme.
  - Intersection of modules and custom code.
- See <https://www.drupal.org/node/2793445> for work in progress.







## Why?

-  Lends itself to bespoke or custom site builds.
-  Use one set of tools for testing.

## Cons

-  Testing without isolation can be brittle.
-  Harder to maintain, state bleeds between test-runs.



DRUPALCON



VIENNA

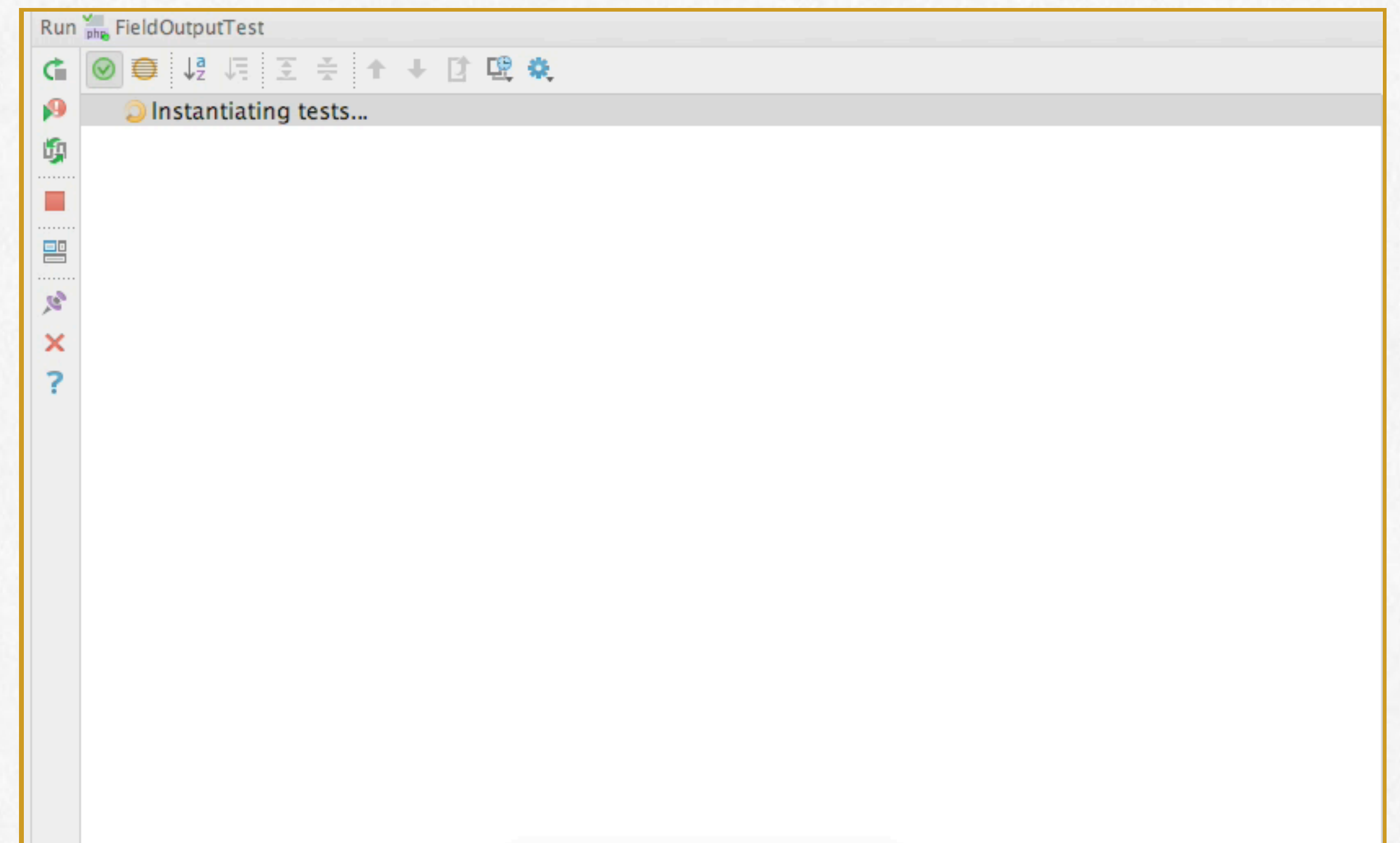
# KernelTestBase





# KernelTestBase

- ✦ Drops the notion of a web browser during testing.
- ✦ Starts with a minimal Drupal installation.
- ✦ Test must specify which parts of Drupal are installed.
- ✦ Fast compared to BTB/JTB.



KernelTestBase, look at that speed!

# Required Setup

```
0
7  /**
8   * An example KernelTestBase test.
9   */
10 abstract class ExampleKernelTest extends KernelTestBase {
11
12     /**
13      * Modules to enable.
14      *
15      * @var array
16      */
17     public static $modules = [
18         'user',
19         'system',
20         'node',
21     ];
22
23     /**
24      * {@inheritdoc}
25      */
26     protected function setUp() {
27         parent::setUp();
28
29         $this->installEntitySchema('node');
30         $this->installSchema('system', 'sequences');
31         $this->installConfig(['system']);
32     }
33
34 }
```

Typical KernelTestBase setup method

- Requires each module, module configuration, database table or entity type to be setup during the test.
- Using real dependencies, asserting the behaviour of one or more components working together.



# Simple Example

## Scenario

- ✦ A simple field formatter.
- ✦ Returns a `#theme => fancy_text` with the field value inside.

## Subsystems working together:

- ✦ Entity/field API (storing the data).
- ✦ Plugin system (for the formatter).
- ✦ Module system (does our fancy\_formatter module actually install etc).

```
7  /**
8   * Tests a field formatter.
9   *
10  * @group example
11  */
12  class FieldOutputTest extends KernelTestBase {
13
14  public static $modules = [
15    'entity_test',
16    'fancy_formatter'
17  ];
18
19  /**
20   * {@inheritdoc}
21  */
22  protected function setUp() {
23    parent::setUp();
24    $this->installEntitySchema('entity_test');
25  }
26
27  /**
28   * Tests a field formatter.
29  */
30  public function testFieldFormatter() {
31    $entity = EntityTest::create([
32      'test_field' => 'My test value',
33    ]);
34    $entity->save();
35
36    $output = $entity->test_field->view([]);
37
38    $this->assertEquals([
39      '#theme' => 'fancy_text',
40      '#text' => 'My test value',
41    ], $output);
42  }
43
44 }
```

DRUPALCON



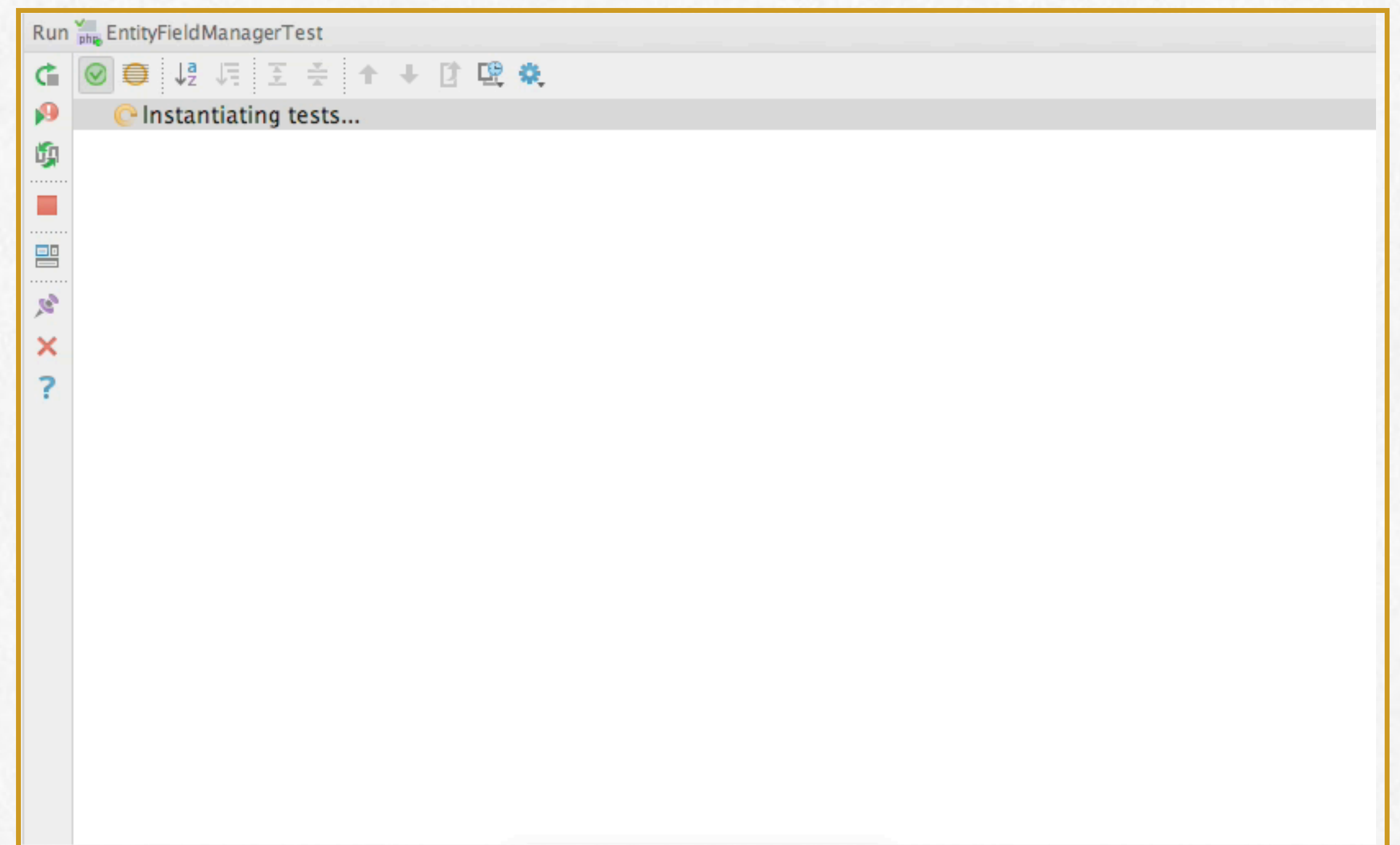
VIENNA

# UnitTestCase



# UnitTestCase

- Super fast!
- Lowest level of abstraction in testing.
- No access to a browser, database or bootstrap.
- Instantiate your system-under-test directly.
- Create exact pre-conditions.
- Test lots of scenarios in very little time.



The slowest unit test in core.



# An Example

- Test doubles for dependencies.
- Asserts the positive and negative test cases.
- A @dataProvider could be added with more test cases as required.

```
12  /**
13   * Tests the Rounder class.
14   *
15   * @coversDefaultClass \Drupal\commerce_price\Rounder
16   * @group commerce
17   */
18  class RounderTest extends UnitTestCase {
19
20   /**
21    * {@inheritdoc}
22    */
23   public function setUp() {
24     parent::setUp();
25
26     $usd_currency = $this->prophesize(CurrencyInterface::class);
27     $usd_currency->id()->willReturn('USD');
28     $usd_currency->getFractionDigits()->willReturn('2');
29
30     $storage = $this->prophesize(EntityStorageInterface::class);
31     $storage->load('USD')->willReturn($usd_currency->reveal());
32     $storage->load('EUR')->willReturn(NULL);
33
34     $entity_type_manager = $this->prophesize(EntityTypeManagerInterface::class);
35     $entity_type_manager->getStorage('commerce_currency')->willReturn($storage->reveal());
36
37     $this->rounder = new Rounder($entity_type_manager->reveal());
38   }
39
40   /**
41    * Tests rounding a price with an unknown currency.
42    */
43   public function testUnknownCurrency() {
44     $this->setExpectedException(\InvalidArgumentException::class);
45     $this->rounder->round(new Price('10', 'EUR'));
46   }
47
48   /**
49    * Tests rounding a valid price.
50    */
51   public function testValid() {
52     $rounded_price = $this->rounder->round(new Price('3.3698', 'USD'));
53     $this->assertEquals('3.37', $rounded_price->getNumber());
54     $this->assertEquals('USD', $rounded_price->getCurrencyCode());
55   }
56
57 }
```





# Testing and Design

- ✦ Clearly reveals all dependencies (you have to instantiate or mock them), encourages information hiding.
- ✦ Good reference for your public interface, clearly see how dependent classes will interact with your class. Does it make sense?



# Test Doubles: Know your lingo

```
// A dummy, NULL returned for all methods.
$mail_manager = $this->prophesize(MailManagerInterface::class);
$mail_manager->reveal();

// A stub, hard-codes the return value of mail.
$mail_manager = $this->prophesize(MailManagerInterface::class);
$mail_manager->mail()->willReturn(TRUE);
$mail_manager->reveal();

// A mock, asserts that the mail method will be called.
$mail_manager = $this->prophesize(MailManagerInterface::class);
$mail_manager->mail('Test Mail', 'foo@example.com')->shouldBeCalled();
$mail_manager->reveal();
```

All are test doubles, the notion of swapping your dependency for a test-specific implementation.

## Dummies

✎ Implements an interface but returns NULL for every method.

## Stubs

✎ Returns user-specified pre-canned responses to method calls.

## Mocks

✎ Asserts specific method calls made to a dependency.







# Unit Test Code Smells

## **Too many test doubles**

- ✦ Perhaps class has too many responsibilities and needs to be spit up.
- ✦ Perhaps less information can be injected.

## **Testing the implementation**

- ✦ Avoid asserting very specific calls to dependencies unless necessary.

## **Stubbing the system under test**

- ✦ Why couldn't the state of the object be setup? Too complex?



DRUPALCON



VIENNA

# Tools & Runners



# PhpStorm Test Runner

```
9  /**
10 * Test the form that helps users setup their one time password.
11 *
12 * @coversDefaultClass \Drupal\one_time_password\Form>PasswordSetupForm
13 * @group one_time_password
14 */
15 class PasswordSetupFormTest extends BrowserTestBase {
16
17     use UserCreationTrait;
18
19     /**
20      * Modules to enable.
21      *
22      * @var array
23      */
24     public static $modules = [
25         'one_time_password',
26         'user',
27         'block',
28     ];
29
30     /**
31      * {@inheritdoc}
32      */
33     protected function setUp() {
34         parent::setUp();
35         $this->drupalPlaceBlock('local_tasks_block');
36     }
37
38     /**
39      * Test the password setup form.
40     */
```

PHPStorm test runner

DRUPALCON



VIENNA

# Limitations





# No JS Unit Testing?

- ✦ Lots of solutions out there for unit testing JavaScript.
- ✦ Helpful for any complex JS code.
- ✦ Not compatible with core or Drupal CI.
  
- ✦ One possible solution: Jest



# Three Easy Steps

```
webpack.config.js x
1  const webpack = require('webpack');
2
3  module.exports = {
4    context: __dirname + '/src/',
5    entry: './index.js',
6    output: {
7      path: __dirname,
8      filename: 'webform_date_restrictions.restrict-calendar.min.js'
9    }
10 };
11
```

Setting up web pack for JS unit testing

- Split your JS into individual units.
- Test those units.
- Integrate tested code into your Drupal-y JS.
- Webpack allows you to split your code into individually tested modules.



# Example JS Module + Test

```
1  /**
2   * JS module to calculate date restrictions.
3   */
4  var DateRestrictions = function (days, dates) {
5    this.days = days;
6    this.dates = dates;
7  };
8
9  /**
10   * Check if a given day falls within our date restrictions.
11   */
12  DateRestrictions.prototype.checkDay = function (day) {
13    var days = [
14      'sunday',
15      'monday',
16      'tuesday',
17      'wednesday',
18      'thursday',
19      'friday',
20      'saturday'
21    ];
22    if (this.days && this.days.indexOf(days[day.getDay()]) !== -1) {
23      return false;
24    }
25    var date_string = [day.getFullYear(), day.getMonth() + 1, day.getDate()].join('-');
26    if (this.dates && this.dates.indexOf(date_string) !== -1) {
27      return false;
28    }
29    return true;
30  };
31
32  module.exports = DateRestrictions;
33
```


```
1  var DateRestrictions = require('../DateRestrictions');
2
3  describe('date restrictions', function () {
4
5    test('no restrictions allow all days', function () {
6      var date = new DateRestrictions(null, null);
7      expect(date.checkDay(new Date('Jul 17 2017'))).toEqual(true);
8      expect(date.checkDay(new Date('Jul 18 2017'))).toEqual(true);
9      expect(date.checkDay(new Date('Jul 19 2017'))).toEqual(true);
10     expect(date.checkDay(new Date('Jul 20 2017'))).toEqual(true);
11     expect(date.checkDay(new Date('Jul 21 2017'))).toEqual(true);
12     expect(date.checkDay(new Date('Jul 22 2017'))).toEqual(true);
13     expect(date.checkDay(new Date('Jul 23 2017'))).toEqual(true);
14   });
15
16   test('restrict thursdays, fridays and 18th July 2017', function() {
17     var date = new DateRestrictions(['thursday', 'friday'], ['2017-7-18']);
18     expect(date.checkDay(new Date('Jul 17 2017'))).toEqual(true);
19     expect(date.checkDay(new Date('Jul 18 2017'))).toEqual(false);
20     expect(date.checkDay(new Date('Jul 19 2017'))).toEqual(true);
21     expect(date.checkDay(new Date('Jul 20 2017'))).toEqual(false);
22     expect(date.checkDay(new Date('Jul 21 2017'))).toEqual(false);
23     expect(date.checkDay(new Date('Jul 22 2017'))).toEqual(true);
24     expect(date.checkDay(new Date('Jul 23 2017'))).toEqual(true);
25   });
26
27 });
28
```

Test in D7 contrib: [d.org/project/webform\\_date\\_restrictions](https://d7.org/project/webform_date_restrictions)



# Using the Module

```
1 (function ($) {
2
3   var DateRestrictions = require('./DateRestrictions');
4
5   /**
6    * Attach to all webform calendars, to restrict the dates selected.
7    */
8   Drupal.behaviors.webformDateRestrictionsRestrictCalendar = {
9     attach: function (context, settings) {
10
11       $('webform-date-restrictions-restrict-dates', context).once('webform-date-rest
12         var $wrapper = $(this);
13         var $calendar = $wrapper.find('webform-calendar');
14
15         var date_restrictions = new DateRestrictions($wrapper.data('restricted-days')
16         if ($calendar.length) {
17           $calendar.datepicker('option', 'beforeShowDay', function(day) {
18             return [date_restrictions.checkDay(day), ''];
19           });
20         }
21       });
22     };
23   });
24 }
25
26 })(jQuery);
27
```

 We're sorry, the date selected is not available. Please choose another.


[Home](#)

## Test Form

[View](#) [Edit](#) [Webform](#) [Results](#)

Submitted by [admin](#) on Thu, 07/20/2017 - 08:57

**Restricted Date**

Jul 12 2017 

| July 2017 |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|
| Su        | Mo | Tu | We | Th | Fr | Sa |
|           |    |    |    |    |    | 1  |
| 2         | 3  | 4  | 5  | 6  | 7  | 8  |
| 9         | 10 | 11 | 12 | 13 | 14 | 15 |
| 16        | 17 | 18 | 19 | 20 | 21 | 22 |
| 23        | 24 | 25 | 26 | 27 | 28 | 29 |
| 30        | 31 |    |    |    |    |    |

Powered by Drupal





# Snapshot Testing

```
(presso): jest --config=jest.config.json generator.test.js FunnelLinkGenerator.test.js.snap
1  /**
2   * Generate links to the sales funnel.
3   */
4   var FunnelLinkGenerator = function () {
5     };
6
7   /**
8    * Check if logging is enabled.
9    */
10  FunnelLinkGenerator.prototype.generateLink = function (address) {
11    if (typeof address !== 'object' || !address.hasOwnProperty('address')) {
12      throw new Error('Unable to generate a funnel link without an address.');
```


```
1  var FunnelLinkGenerator = require('../shared/FunnelLinkGenerator');
2
3  describe('funnel link generator', function () {
4
5    var funnel_link_generator = new FunnelLinkGenerator();
6
7    test('invalid address', function () {
8      expect(function () {
9        funnel_link_generator.generateLink()
10      }).toThrow(new Error('Unable to generate a funnel link without an address.');
```

Updating a snapshot with Jest





@see core

 <https://www.drupal.org/node/2702747> - Issue to track including JavaScript unit testing in core.



DRUPALCON



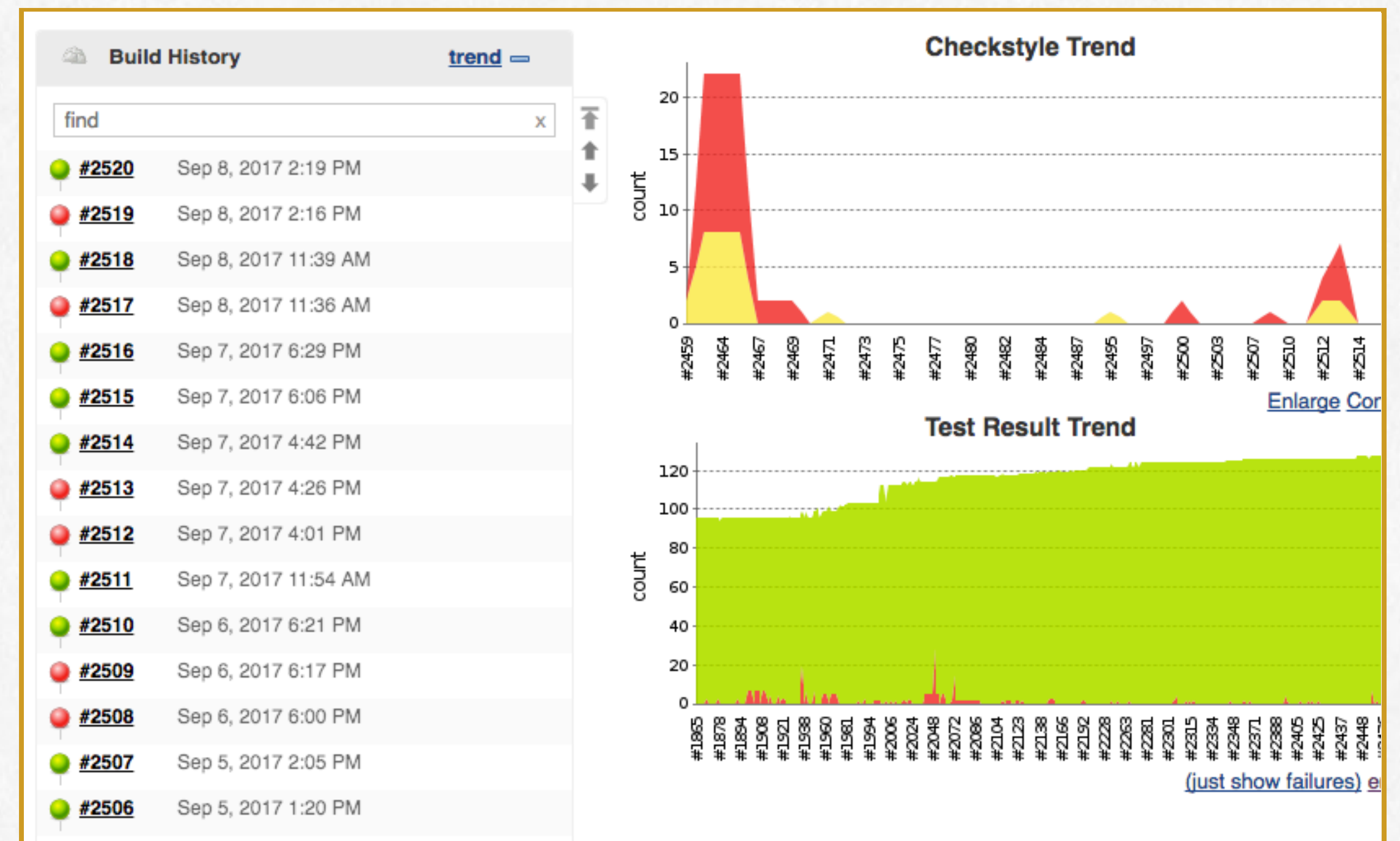
VIENNA

Bringing it all together



# Continuous Integration

- Invest in a CI for your private projects.
- CircleCI and Gitlab have great free tiers to dip your toe in.



Build history on a complex Drupal 8 project.



DRUPALCON



VIENNA

# Discussion & questions





# JOIN US FOR CONTRIBUTION SPRINT

Friday, September 29, 2017

Mentored  
Core Spint

9:00-12:00  
Room: Stolz 2

First time  
Sprinter Workshop

9:00-12:00  
Room: Lehar 1 - Lehar 2

General sprint

9:00-12:00  
Room: Mall

#drupalsprints





# WHAT DID YOU THINK?

Locate this session at the DrupalCon Vienna website:

<http://vienna2017.drupal.org/schedule>

Take the survey!

<https://www.surveymonkey.com/r/drupalconvienna>

