How to get started with writing tests for contrib

Brent Gees

Dropsolid
The Digital Business Company

Diamond Sponsor

Acquia®

Dropsolid

The Digital Business Company

Platinum Sponsors

Gold Sponsors

Slides + example module

http://bit.ly/lissabon-testing

http://bit.ly/lissabon-testing-module

Dropsolid

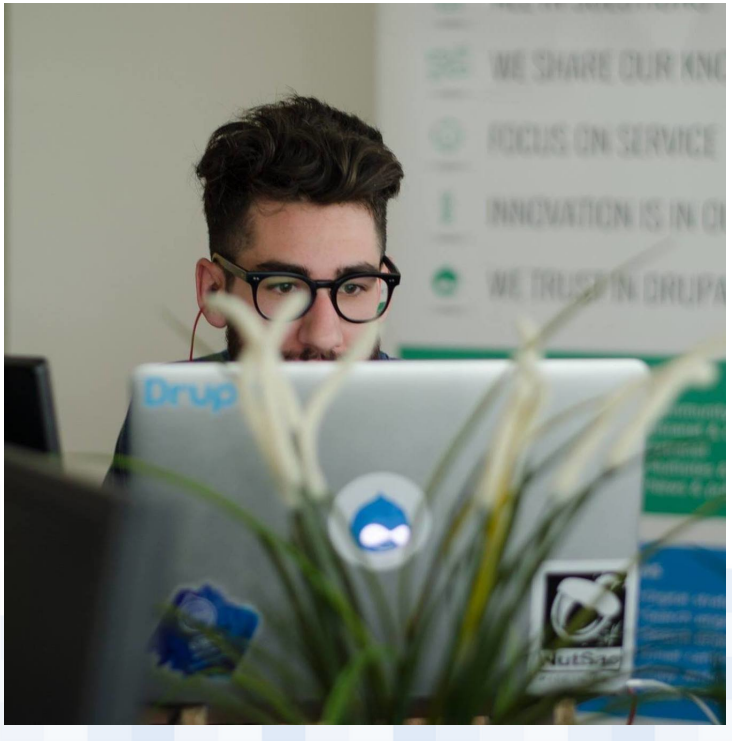
The Digital Business Company

Who am I?

Dropsolid
The Digital Business Company

Who am I?

- Brent
- Developer / Architect
- @brentgees



Dropsolid

The Digital Business Company


DRUPALCAMP


DRUPALCAMP

Who are you?

What is testing

"Software testing is a process used to identify the correctness, completeness, and quality of a developed computer software.

It includes a set of activities conducted with the intent of finding errors in software so that it could be corrected before the product is released to the end user"

“In simple words, software testing is an activity to check that the software is defect free”

Why this session

Why this session?

- The first sprint for me was testing
- You'll learn a lot of new things
- Really helpful and enough issues
- You don't have to know the complete module

Dropsolid
The Digital Business Company

Own modules vs other modules

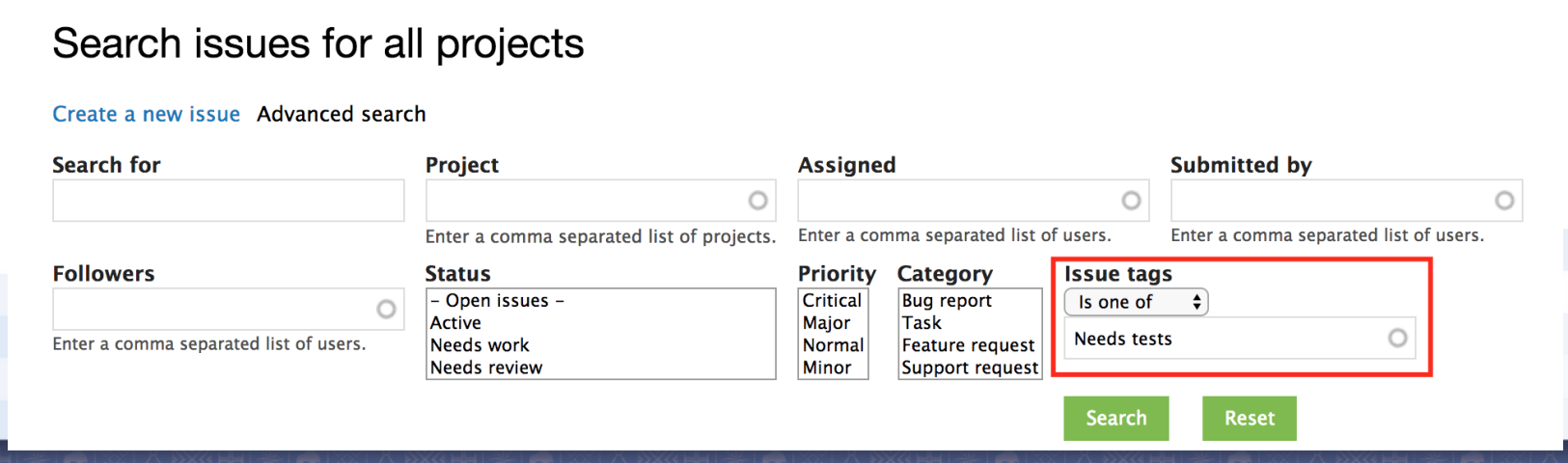
Own modules

- Easy tests are still available
- You'll have to set up everything yourself

Dropsolid
The Digital Business Company

Other modules

- Will have a lot of functions already created
- Easy parts will be gone most likely
- Best if you can find a maintainer



- 2000 tagged with 'needs tests' in core
- 4000 tagged with 'needs tests' overall

Drupal core	Core oembed support can't parse soundcloud XML	Needs work	Normal	Bug report	8.6.x-dev	3	6 days 23 hours	6 days 23 hours	1 week 19 min
Drupal core	Views Date Filter Datetime Granularity Option	Needs work	Major	Feature request	8.6.x-dev	41	1 week 1 hour	3 months 2 weeks	1 year 2 months
Drupal core	Handle computed fields in entity queries: throwing a helpful exception is better than a PHP fatal error	Needs work	Normal	Bug report	8.6.x-dev	22	1 week 1 hour	1 week 1 hour	2 months 2 weeks
Drupal core	Formatter settings lost when moving fields between regions in 'Manage display'	Needs work	Normal	Bug report	8.6.x-dev	16	1 week 6 hours	1 month 4 weeks	11 months 2 weeks
Drupal core	DatabaseSchema_mysql::createTableSql() can't set table collation	Needs work	Major	Bug report	8.5.x-dev	29	1 week 15 hours	1 week 15 hours	4 years 4 months
Drupal core	Object of class Drupal\\Core\\Session\\AccountProxy could not be converted to string	Needs work	Normal	Bug report	8.6.x-dev	3	1 week 19 hours	1 week 19 hours	2 weeks 20 hours

1 2 3 4 5 6 7 8 9 … next › last »

Why you should write tests

Dropsolid
The Digital Business Company


FIXES ISSUE ON THE NEWS PAGE
CONTACT PAGE IS BROKEN
imgflip.com

The China airline Airbus A300 crashed due to a software bug in 1994 killing 264 people.

Software bug caused the failure of a military satellite launch, causing a loss of $1.2 billion

responsibility

When your module / patch is used by multiple people, you should make sure that it will work with every update, otherwise you'll break other peoples websites.

Types of testing

- Manual testing
- Automated testing
  - Unit testing
  - Kernel testing
  - Functional testing

Happens way too often...

Manual testing

Dropsolid
The Digital Business Company

Manual testing

- Done by developer, tester, client and/or project manager
- Most primitive of all testing types
- Mostly used for short-term projects
- Easy to forget use cases
- Doesn't require a lot of time
- Becomes very boring if you have to execute the same test multiple times.

Manual testing

- In the browser
- With xdebug, var_dump, dsm, dpm, kint
- Reading code

Manual testing

Issues for all projects

Create a new issue Advanced search

Search for

Project

Enter a comma separated list of projects.

Status Needs review

Priority – Any –

Category – Any –

Search Reset

Displaying 50+ issues.

Hide irrelevant issues Mark all as read

Project	Title	Status	Priority	Category	Version	Replies	Last updated	Assigned to	Created
Drupal core	Module install doesn't invalidate render cache new	Needs review	Normal	Bug report	8.6.x-dev	41 2 new	33 sec		1 year 10 months
Drupal core	"Add media" button missing in the media library table display new	Needs review	Normal	Bug report	8.6.x-dev	18 18 new	3 min 15 sec		2 days 1 hour
Drupal core	Redirect back to media list after creating a media entity new	Needs review	Normal	Task	8.6.x-dev	15 12 new	15 min 10 sec		1 month 2 weeks
Drupal core	Add static cache to \Drupal\Core\Entity\ContentEntityStorageBase::getLatest*RevisionId() new	Needs review	Major	Task	8.6.x-dev	4 4 new	15 min 23 sec		2 hours 29 min

https://www.drupal.org/project/issues?projects=&status=8

Dropsolid
The Digital Business Company

Unit testing

Unit testing

- Tests on functions, methods, classes
- Extends on the class UnitTestCase

Advantages of unit testing

- Verify individual parts
- Quickly find problems in code
- Fast execution
- No system setup for the test run

Disadvantages of unit testing

- Refactoring your code might require tests to be rewritten
- Complicated mocking
- No guarantee that the whole system actually works

Folder structure

```
../modules/custom
--lissabon
----lissabon.info.yml
----lissabon.module
----src
------Lissabon.php
----tests
------src
--------Unit
----------LissabonSumTest.php
```

Lissabon.php

```php
<?php

namespace Drupal\lissabon;

/**
 * Defines a Lissabon class.
 */
class Lissabon {
  private $total = 0;

  /**
   * Returns the total amount.
   *
   * @return int
   *   Returns the total
   */
  public function getTotal() {
    return $this->total;
  }

  /**
   * Sets the total.
```

LissabonSumTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\Tests\UnitTestCase;

/**
 * Defines a Unit class.
 *
 * @group lissabon
 */
class LissabonSumTest extends UnitTestCase {
  protected $lissabon;

  /**
   * Before a test method is run, setUp() is invoked.
   *
   * We create a new object of the class Lissabon.
   */
  public function setUp() {
    $this->lissabon = new Lissabon();
  }

}
```

LissabonSumTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\Tests\UnitTestCase;

/**
 * Defines a Unit class.
 *
 * @group lissabon
 */
class LissabonSumTest extends UnitTestCase {

  /**
   * We unset the lissabon object.
   *
   *  Once test method has finished running, whether it succeeded or
   *  failed, tearDown() will be invoked.
   */
  public function tearDown() {
    unset($this->lissabon);
  }

}
```

LissabonSumTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\Tests\UnitTestCase;

/**
 * Defines a Unit class.
 *
 * @group lissabon
 */
class LissabonSumTest extends UnitTestCase {

  /**
   * Covers setTotal.
   */
  public function testSetTotal() {
    $this->assertEquals('0', $this->lissabon->getTotal());
    $this->lissabon->setTotal(366);
    $this->assertEquals(366, $this->lissabon->getTotal());
  }

  /**
   * Covers getTotal.
   */
  public function testGetTotal() {
```

Setting it up for the first time

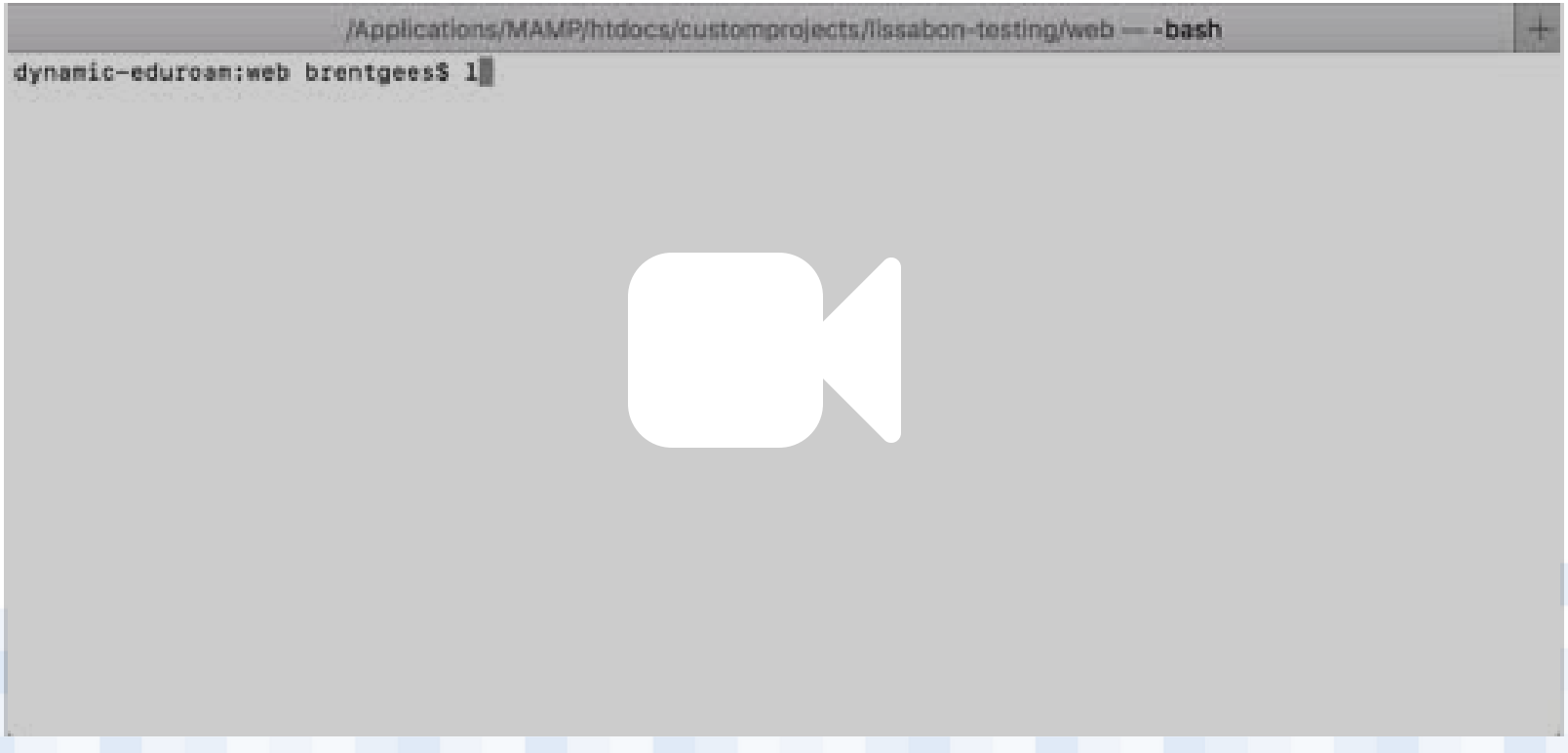
```
# Download a drupal installation file (you can also use git clone here)
composer create-project drupal-composer/drupal-project:8.x-dev lissabon-testing
--stability dev --no-interaction --prefer-source


# Install the module you're working on (unless you're working on core,
# here you can alsou use git clone if you want)
composer require drupal/example --prefer-source


# go to the core folder in the web directory
cd lissabon-testing/web/core

# Copy the phpunit.xml.dist file to phpunit.xml
cp phpunit.xml.dist phpunit.xml
```

Now to the testing part

```
# Go to the root of your website (web folder)
cd ..

# start the test (change example for the module you're using or leave it empty to test core)
../vendor/bin/phpunit -c core modules/example


# If you want to test 1 specific test, you can add the following option
--filter testName
```

In action

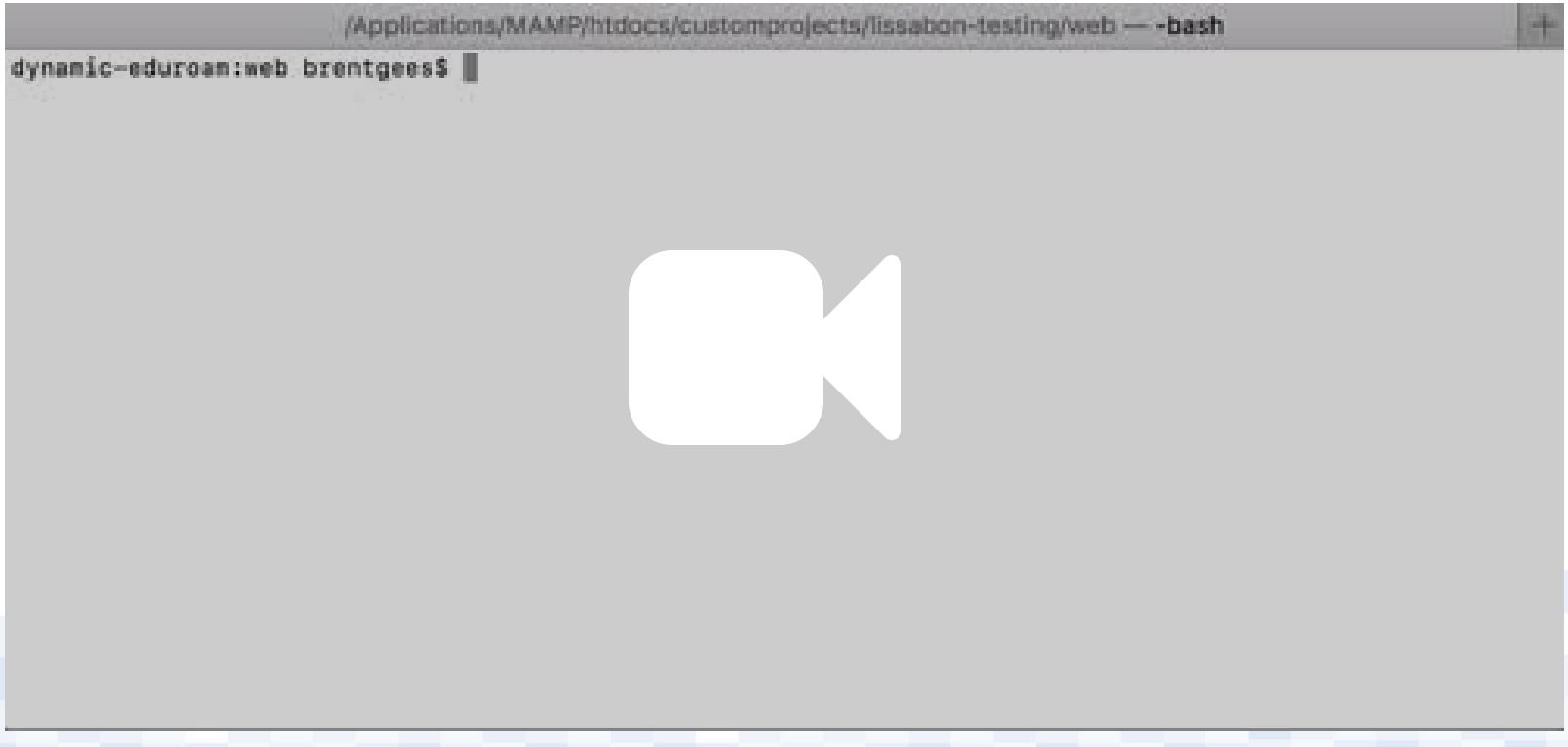
```
../vendor/bin/phpunit modules/custom/lissabon/tests/src/Unit/
```



Looks like it's not working yet

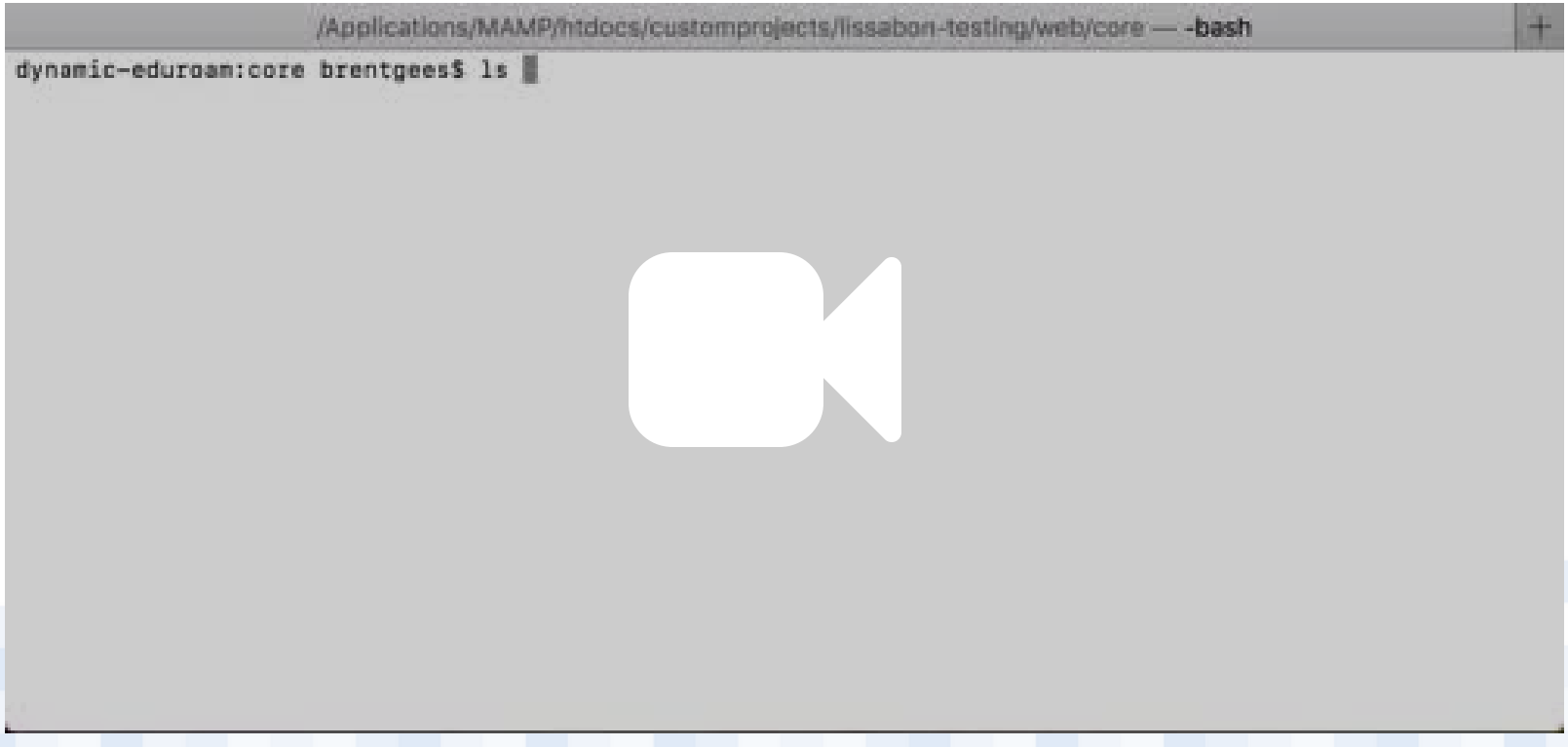
In action

```
../vendor/bin/phpunit -c core modules/custom/lissabon/tests/src/Unit/
```



In action

```
#or, go into the core folder
cd core
../../vendor/bin/phpunit ../modules/custom/lissabon/tests/src/Unit/
```



Kernel testing

Kernel testing

Kernel tests are integration tests that test on components.

You can install modules

Minimal Drupal, full api, without http

Extends from KernelTestBase or EntityKernelTestBase

Advantages of Kernel testing

- Verify that components actually work together
- Somewhat easy to locate bugs

Disadvantages of Kernel testing

- Slower execution
- System setup required
- No guarantee that end user features actually work

Folder structure

```
../modules/custom
--lissabon
----lissabon.info.yml
----lissabon.module
----config
------install
--------lissabon.settings.yml
------schema
--------lissabon.schema.yml
----src
------Form
--------LissabonConfigForm.php
------Lissabon.php
----tests
------src
--------Kernel
----------LissabonConfigTest.php
```

Dropsolid
The Digital Business Company

LissabonConfigForm.php

```php
<?php

namespace Drupal\lissabon\Form;

use Drupal\Core\Form\ConfigFormBase;
use Drupal\Core\Form\FormStateInterface;

/**
 * Defines a Configuration form.
 */
class LissabonConfigForm extends ConfigFormBase {

  /**
   * {@inheritdoc}
   */
  public function getFormId() {
    return 'lissabon_config_form';
  }

  /**
   * {@inheritdoc}
   */
  protected function getEditableConfigNames() {
    return [
      'lissabon.settings',
    ];
  }
```

lissabon.settings.yml

```
lissabon_name: 'Dev days is awesome!'
```

lissabon.schema.yml

```
lissabon.settings:
  type: config_object
  label: 'Lissabon testing'
  mapping:
    lissabon_name:
      type: string
      label: 'Lissabon name'
```

Dropsolid

The Digital Business Company

LissabonConfigTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\KernelTests\KernelTestBase;

/**
 * Tests the config for Lissabon.
 *
 * @package Drupal\lissabon
 */
class LissabonConfigTest extends KernelTestBase {


  /**
   * User for testing.
   *
   * @var \Drupal\user\UserInterface
   */
  protected $testUser;

  /**
   * Modules to enable.
   *
   * @var array
   */
  public static $modules = [
    'system',
```

LissabonConfigTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\KernelTests\KernelTestBase;

/**
 * Tests the config for Lissabon.
 *
 * @package Drupal\lissabon
 */
class LissabonConfigTest extends KernelTestBase {

  /**
   * Tests the default config.
   */
  public function testDefaultLissabonConfig() {
    $config = $this->config('lissabon.settings');
    $lissabon_name = $config->get('lissabon_name');
    $this->assertEquals('Dev days is awesome!', $lissabon_name);
  }

  /**
   * Tests changing the config.
   */
  public function testChangeLissabonConfig() {
    // First we check if the config is the default one.
```

Setting it up for the first time

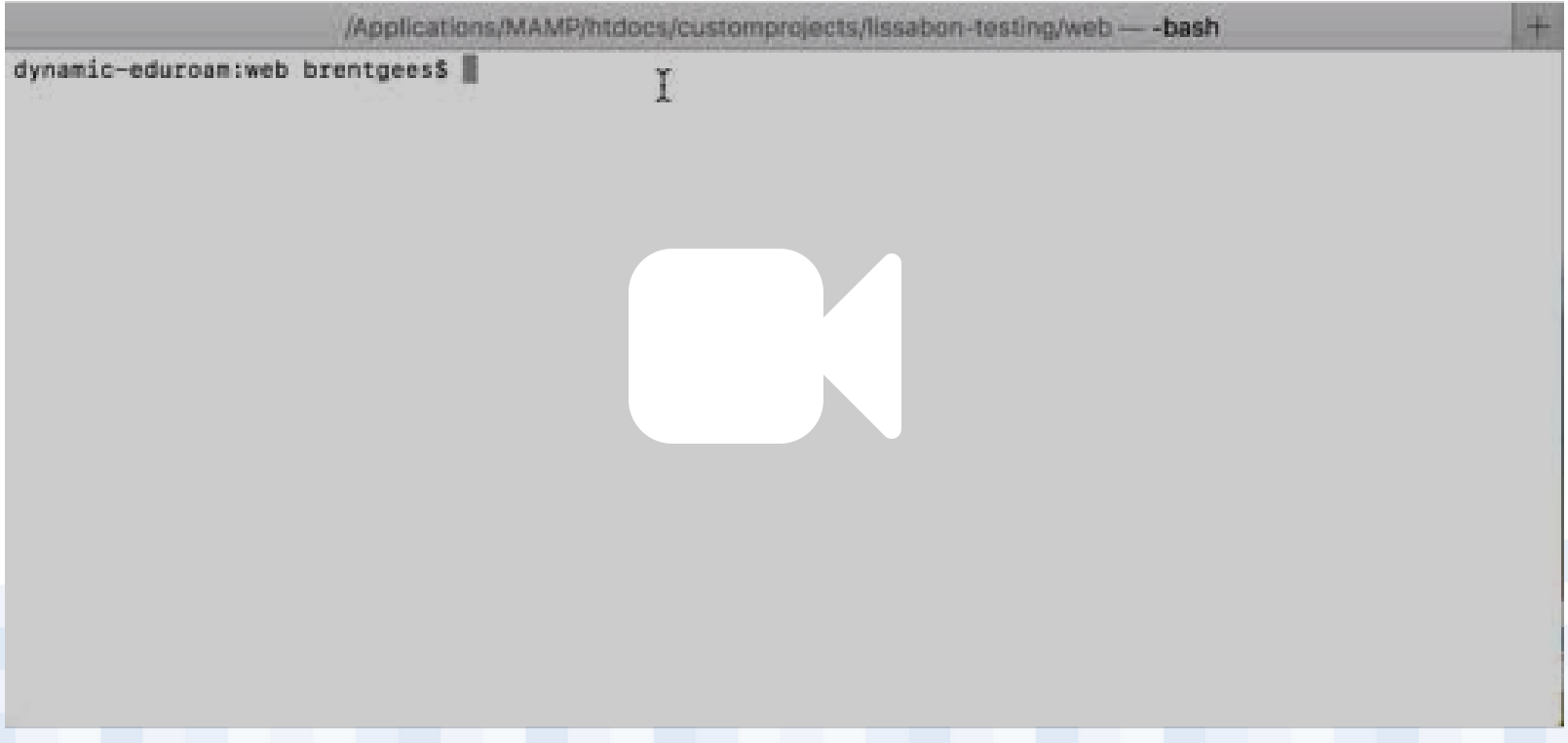
```
# Download a drupal installation file
composer create-project drupal-composer/drupal-project:8.x-dev lissabon-testing
--stability dev --no-interaction --prefer-source


# Install the module you're working on (unless you're working on core)
composer require drupal/example --prefer-source



# go to the core folder in the web directory
cd lissabon-testing/web/core


# Copy the phpunit.xml.dist file to phpunit.xml
cp phpunit.xml.dist phpunit.xml

# Create a database for your website (e.g. lissabon_testing).
# in the phpunit.xml file, update the following lines:

<env name="SIMPLETEST_DB" value=""/>
# to
<env name="SIMPLETEST_DB" value="mysql://root:root@localhost/lissabon_testing"/>

# where you change the value with your values (mysql://username:password@localhost/database_name)
```

Now to the testing part

```
# Go to the root of your website (web folder)
cd ..

# start the test (change example for the module you're using or leave it empty to test core)
../vendor/bin/phpunit -c core modules/example


# If you want to test 1 specific test, you can add the following option
--filter testName
```

In action

```
../vendor/bin/phpunit -c core modules/custom/lissabon/tests/src/Kernel/
```



Functional testing

Dropsolid
The Digital Business Company

Functional testing

2 types:

- BrowserTestBase
- JavascriptTestbase

Advantages of functional testing

- Verify that the system works as experienced by the user
- Verify that the system works when code is refactored

Disadvantages off functional testing

- Very slow execution
- Heavy system setup
- Hard to locate origins of bugs
- Prone to random test fails
- Hard to change

Folder structure

```
../modules/custom
--lissabon
-----lissabon.info.yml
-----lissabon.module
-----lissabon.routing.yml
-----config
-------install
---------lissabon.settings.yml
-------schema
---------lissabon.schema.yml
-----src
-------Form
---------LissabonConfigForm.php
-------Lissabon.php
-----tests
-------src
---------Functional
-----------LissabonConfigFormTest.php
-----------LoadTest.php
```

lissabon.routing.yml

```
# DropsolidHeaderDefaults routing definition
entity.lissabon_routing.collection:
  path: '/admin/lissabon'
  defaults:
    _form: '\Drupal\lissabon\Form\LissabonConfigForm'
    _title: 'Lissabon configuration'
  requirements:
    _permission: 'administer site configuration'
  options:
    _admin_route: TRUE
```

Dropsolid

The Digital Business Company

LissabonConfigForm.php

```php
<?php

namespace Drupal\lissabon\Form;

use Drupal\Core\Form\ConfigFormBase;
use Drupal\Core\Form\FormStateInterface;

/**
 * Defines a Configuration form.
 */
class LissabonConfigForm extends ConfigFormBase {

  /**
   * {@inheritdoc}
   */
  public function getFormId() {
    return 'lissabon_config_form';
  }

  /**
   * {@inheritdoc}
   */
  protected function getEditableConfigNames() {
    return [
      'lissabon.settings',
    ];
  }
```

LissabonConfigFormTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\Tests\BrowserTestBase;

/**
 * Tests the config form.
 *
 * @package Drupal\lissabon
 */
class LissabonConfigFormTest extends BrowserTestBase {

  protected $user;
  protected $editForm;


  /**
   * Modules to enable.
   *
   * @var array
   */
  public static $modules = ['lissabon'];

  /**
   * {@inheritdoc}
   */
```

LissabonConfigFormTest.php

```php
<?php

namespace Drupal\lissabon;

use Drupal\Tests\BrowserTestBase;

/**
 * Tests the config form.
 *
 * @package Drupal\lissabon
 */
class LissabonConfigFormTest extends BrowserTestBase {

  /**
   * Tests the configuration form.
   */
  public function testConfigForm() {
    // We try to change the form on /admin/lissabon.
    $this->editForm = 'admin/lissabon';
    $form = [
      'edit-lissabon-name' => 'Lissabon is awesome',
    ];
    $this->drupalPostForm($this->editForm, $form, 'Save');

    // We check if our change went through.
    $this->drupalGet('admin/lissabon');
    $this->assertResponse(200);
```

Setting it up for the first time

```
# Download a drupal installation file
composer create-project drupal-composer/drupal-project:8.x-dev lissabon-testing
--stability dev --no-interaction --prefer-source


# Install the module you're working on (unless you're working on core)
composer require drupal/example --prefer-source

# go to the core folder in the web directory
cd lissabon-testing/web/core

# Copy the phpunit.xml.dist file to phpunit.xml
cp phpunit.xml.dist phpunit.xml

# Create a database for your website (e.g. lissabon_testing).
# in the phpunit.xml file, update the following lines:

<env name="SIMPLETEST_DB" value=""/>
# to
<env name="SIMPLETEST_DB" value="mysql://root:root@localhost/lissabon_testing"/>

# where you change the value with your values (mysql://username:password@localhost/database_name)

# Now you'll have to set up a base url as well, so change the following line
<env name="SIMPLETEST_BASE_URL" value=""/>
# to
<env name="SIMPLETEST_BASE_URL" value="http://lissabontesting.local"/>
# where you change the value with your values (http://lissabontesting.local)
```

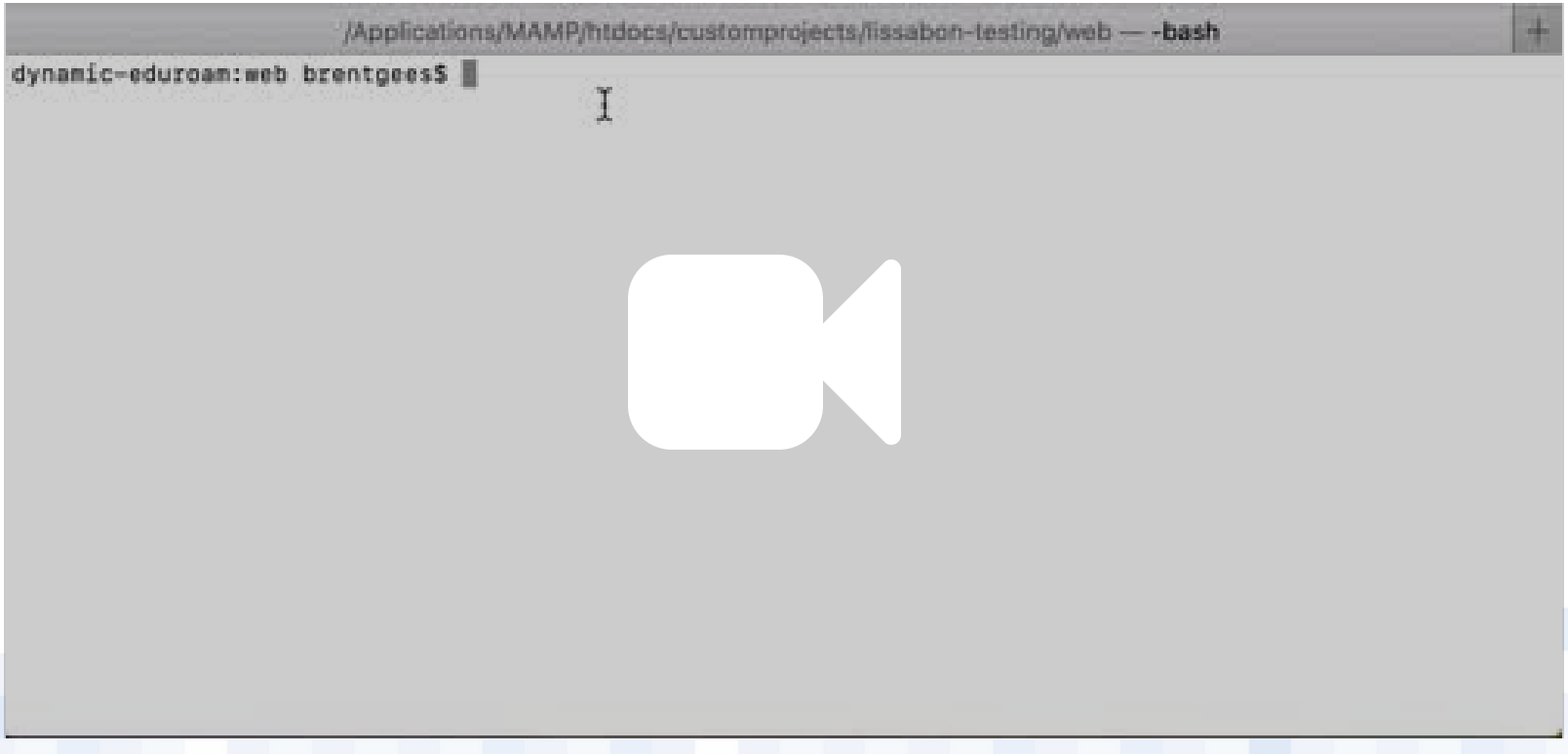
Now to the testing part

```
# Go to the root of your website (web folder)
cd ..

# start the test (change example for the module you're using or leave it empty to test core)
../vendor/bin/phpunit -c core modules/example


# If you want to test 1 specific test, you can add the following option
--filter testName
```

Dropsolid
The Digital Business Company

In action

```
../vendor/bin/phpunit -c core modules/custom/lissabon/tests/src/Unit/
```

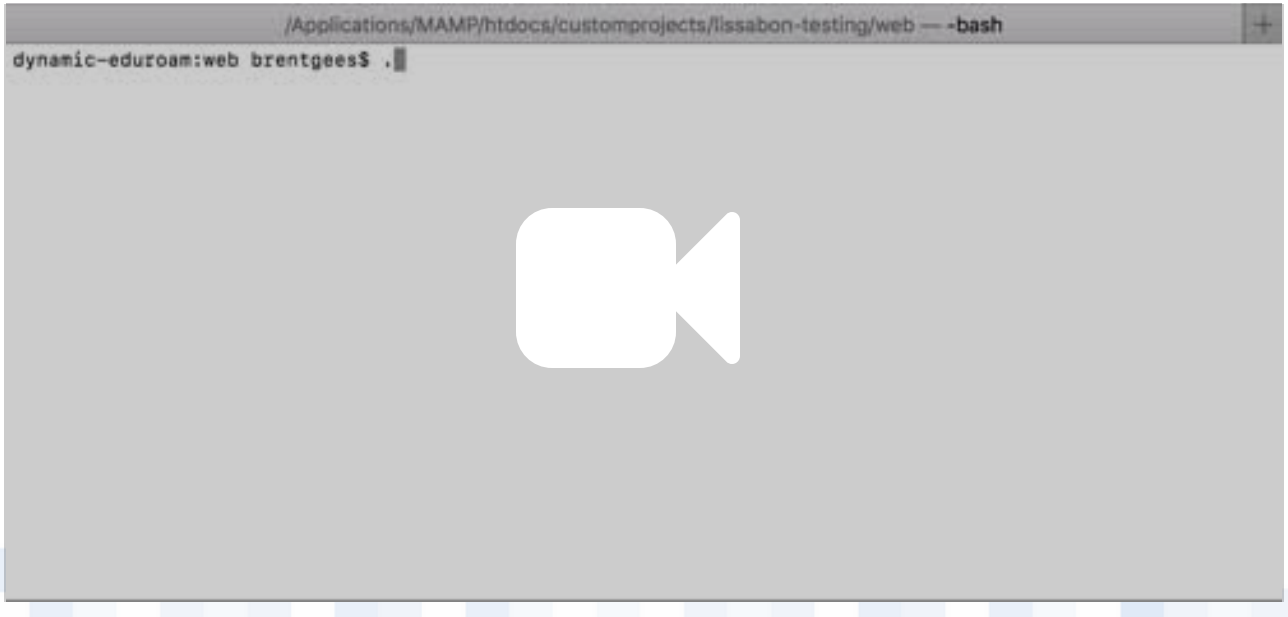


Debugging functional tests

```
# In the phpunit.xml file, change
<env name="BROWSERTEST_OUTPUT_DIRECTORY" value=""/>
# to
<env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/Applications/MAMP/htdocs/customprojects/lissabon

# where The value is a writable folder on your site
```

In action

```
# Execute test while printing html
../vendor/bin/phpunit -c core modules/custom/lissabon/tests/src/Functional/
--printer="\Drupal\Tests\Listeners\HtmlOutputPrinter"
```



To summarise

- If you want to test functions, Unit tests
- If you want to test module APIs without HTTP with a database, use Kernel tests
- If you want to test web interfaces, Use Functional tests

Tips and tricks

Dropsolid
The Digital Business Company

Start small

By starting with the easy task, you'll get more and more familiar with the code for testing.

Learn from others

A lot of tests are similar, use that to your advantage and read other tests of similar modules.

Find a maintainer

Try to find a maintainer/co-maintainer of a module you would like to help on.

They will be glad to help you and guide you through it.

Create a roadmap (own modules)

On your project page, create a roadmap with future steps of your module.

Add a segment tests that you'll update along the way

Dropsolid

The Digital Business Company

Use Drupal console

When creating a module with Drupal console, it asks if you want to create a test.

This is a very simple functional test to check if the front page is still working

Use Drupal console

```php
<?php

namespace Drupal\Tests\facets_autocomplete\Functional;

use Drupal\Core\Url;
use Drupal\Tests\BrowserTestBase;

/**
 * Simple test to ensure that main page loads with module enabled.
 *
 * @group facets_autocomplete
 */
class LoadTest extends BrowserTestBase {

  /**
   * Modules to enable.
   *
   * @var array
   */
  public static $modules = ['facets_autocomplete'];

  /**
   * A user with permission to administer site configuration.
   *
   * @var \Drupal\user\UserInterface
   */
  protected $user;
```

Questions?

@brentgees