
new target

Automated Testing with Drupal: What Tests Should We Write?

...

Kevin McCulloch
Administrative Office (AO) of the US Courts/New Target

Why test?

1. To define requirements and guide implementation
2. To guard against regressions

Behavior-Driven Development

- We write *acceptance* tests
- We write them in a business-domain language shared between stakeholders and developers
- They follow a "Context-Action-Outcome" structure
- They are declarative, not imperative
- Taken together, they comprise "living documentation" for the application

For example

```
Scenario: Successful sign up
  New users should get a confirmation email and be greeted
  personally by the site once signed in.

  Given I have chosen to sign up
  When I sign up with valid details
  Then I should receive a confirmation email
  And I should see a personalized greeting message
```

[Wynne & Hellesøy, The Cucumber Book (2012), p. 6]


Dodd-Frank Act Report
Kevin
www.uscourts.gov/statistics-reports/publications/dodd-frank-act-report
Email Updates
Court Locator
Careers
News
Search uscourts.gov
UNITED STATES COURTS
About Federal Courts
Judges & Judgeships
Services & Forms
Court Records
Statistics & Reports
Rules & Policies
Statistics & Reports
Publications
Civil Litigation Management Manual
Courtroom Technology Manual
Dodd Frank Act Report
Federal Court System in the U.S.
Federal Probation Journal
Journalist's Guide to the Federal Courts
Judiciary Conferences That Cost More Than $100,000
Dodd-Frank Act Report
The Dodd Frank Report studies the resolution of financial institutions as required by the Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010.
In response to the global economic turmoil that began in late 2007, the Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010, Pub. L. No. 111-203 (2010), introduced a broad array of regulatory reforms in the financial sector. The reforms in Title II of the Dodd-Frank Wall Street Reform Act are intended to mitigate risks posed by the failure of systemically important financial institutions. Title II provisions direct the Administrative Office of the United States Courts (AO) to study the resolution of these institutions and report on its findings.
The AO submitted its first three annual reports to Congress pursuant to 12 U.S.C. § 5382(e) on July 21, 2011 (First Report), July 17, 2012 (Second Report), and July 19, 2013 (Third Report). Beginning in July 2015, the AO is required to submit reports every five years.
2011 Dodd-Frank Report
2012 Dodd-Frank Report
Download the Report
2015 Dodd Frank Report
(PDF, 465.58 KB)
2013 Dodd Frank Report
(PDF, 1.91 MB)
2012 Dodd Frank Report
(PDF, 1.21 MB)
2011 Dodd Frank Report
(PDF, 521.77 KB)

"Related Downloads Block" requirements

- Title and 1-4 file downloads are required
- Must allow an optional description, limited to 140 characters
- Each file should show an icon, the file name, the document type and the file size
- Content creators must be able to create one of these when creating or editing a page
- They must be able to reuse one that was created earlier for a different page
- They must be able to upload and provide metadata for files when creating/editing
- They must be able to reuse previously-uploaded files when creating/editing
- File types limited to txt, doc, pdf

What will we use to build it?

- A bunch of contrib modules (bean, entityreference, inline_entity_form, file_entity, media, maxlength) to define the Drupal entity and make it editable
- Some custom code to extract the file type and size from the file entity and add it to the markup
- Component-oriented CSS to style the block
- Positioning of the block inside our responsive panels node_view layout

Write a test in Gherkin

```
Feature: Related Downloads Block
As a site visitor
I want to know what file downloads are available on a page
So that I can decide whether or not I want to download them

  Scenario: I visit the page
    Given a page with a Related Downloads Block
    When I visit that page
    Then I should see the Related Downloads Block in the
    right sidebar
```

The same test, using Drupal Extension step definitions

```
Given "page" content:
| title     | url          | sidebar_block_id |
| Test Page | 'test-page' | 1                |
And "sidebar block" bean:
| id | Title                | File Name              |
| 1  | Download the Report | 2015 Dodd Frank Report |
When I visit 'test-page'
Then I should see "Download the Report" in the sidebar title
And I should see "2015 Dodd Frank Report" in the sidebar
region
```

The Behat Drupal Extension: Reasons to be wary

- Behat is slow, so a lot of tests will bog down our continuous integration process
- If we're committed to true BDD acceptance testing as a design strategy, we'll need to write higher-level step definitions to make less imperative tests
- If we try to use Behat to provide full regression coverage, we're going to wind up with brittle tests that are hard to maintain and we're still going to miss bugs

How *should* we use the Behat Drupal Extension?

- Sparingly
- Think of it as a "test script runner," not a BDD design tool
- Focus on areas of user interaction, like forms and AJAX responses
- Only test page rendering ("When I visit...") if the rendering should differ based on user conditions (roles, permissions) or entity conditions (published/not published)

Going back to our sidebar block, what should we do?

- Skip Behat
- For the tiny bit of code we're writing ourselves, use PHPSpec
- For general regression test coverage, use a visual regression testing tool

Continuing Conversation...

Thu 1PM: I'll be at the **mentoring table** to talk Behat/PHPSpec (downstairs)

Thu 2PM: **Behat and Drupal for Absolute Beginners** (Balcony C)

Thu 3PM: **BDD Strategies for Rock-Solid Automated Testing in Drupal** (Room B)

Fri Lunch: **Birds-of-a-Feather** session on testing (location TBD)

Fri 2PM: **Ensuring Quality through Automated Visual Regression Testing** (Balcony C)

How I used to write Drupal hooks

```
/**
 * Implements hook_something().
 */
function mymodule_something($input_from_drupal) {
  if ($input_from_drupal['foo'] == 'bar') {
    $input_from_drupal['foo'] = 'baz';
  }
  return $input_from_drupal;
}
```

How I write them so that I can run PHPSpec

```
use Drupal\mymodule\MyProcessor;

/**
 * Implements hook_something().
 */
function mymodule_something($input_from_drupal) {
  $processor = new MyProcessor();

  return $processor->doYourThing($input_from_drupal);
}
```

My processing object

```
namespace Drupal\mymodule;

class MyProcessor() {
  public function doYourThing($input_from_drupal) {
    if ($input_from_drupal['foo'] == 'bar') {
      $input_from_drupal['foo'] = 'baz';
    }
    return $input_from_drupal;
  }
}
```

My PHPSpec test

```
namespace spec\Drupal\mymodule;
class MyProcessorSpec extends PhpSpec\ObjectBehavior {
  public function it_replaces_bar_with_baz() {
    $this->doYourThing(array('foo' => 'bar'))
      ->shouldReturn(array('foo' => 'baz'));
  }
  public function it_leaves_other_kinds_of_foo_alone() {
    $this->doYourThing(array('foo' => 'anything else'))
      ->shouldReturn(array('foo' => 'anything else'));
  }
}
```