

Integration Testing PHP Applications

OSCON 2008
Mike Naberezny



<http://maintainable.com>

About Me

- <http://mikenaberezny.com>
- <http://maintainable.com>
- <http://ohloh.net/accounts/mnaberez>

Later Today

- Supervisor as a Platform
(UNIX Process Control System in Python)
- Room D138 at 4:30pm

About You

- Web application developer, using PHP
- Already writing unit tests
- Ready for the next level

Integration Tests

- Testing the application or its components on a higher level than unit tests
- Overlap with functional, acceptance tests. Don't get hung up on terminology.
- Making HTTP Requests (real or fake) to the application and testing the responses

Agenda

- Markup and Testability
- CSS Selector Basics
- PHPUnit SeleniumTestCase
- Roll Your Own
- Extras
- Q & A

Markup and Testability

Testability

- Unit testing can be very difficult if the application does not cleanly separate concerns
- Tests that look at response HTML are more forgiving about the underlying implementation
- Separating concerns on both the server side and the client side greatly enhances testability and maintainability

Yesterday's Application

- Big files, all mixed up
 - PHP and SQL
 - HTML with `` tags
 - JavaScript
- Unit testing next to impossible
- Integration testing possible, probably not fun

Today's Application

- Small, well-separated files
 - PHP projects became more organized
 - Basic object orientation greatly helped
 - Templating systems separated application logic from presentation logic
- Unit testing possible, Maintenance easier
- Hopefully you are here already

Application Developer

- Server-side technologies you need to know:
 - PHP, Basic OOP, Basic Testing
 - Relational Databases
 - HTTP and Best Practices
 - Separation of Concerns

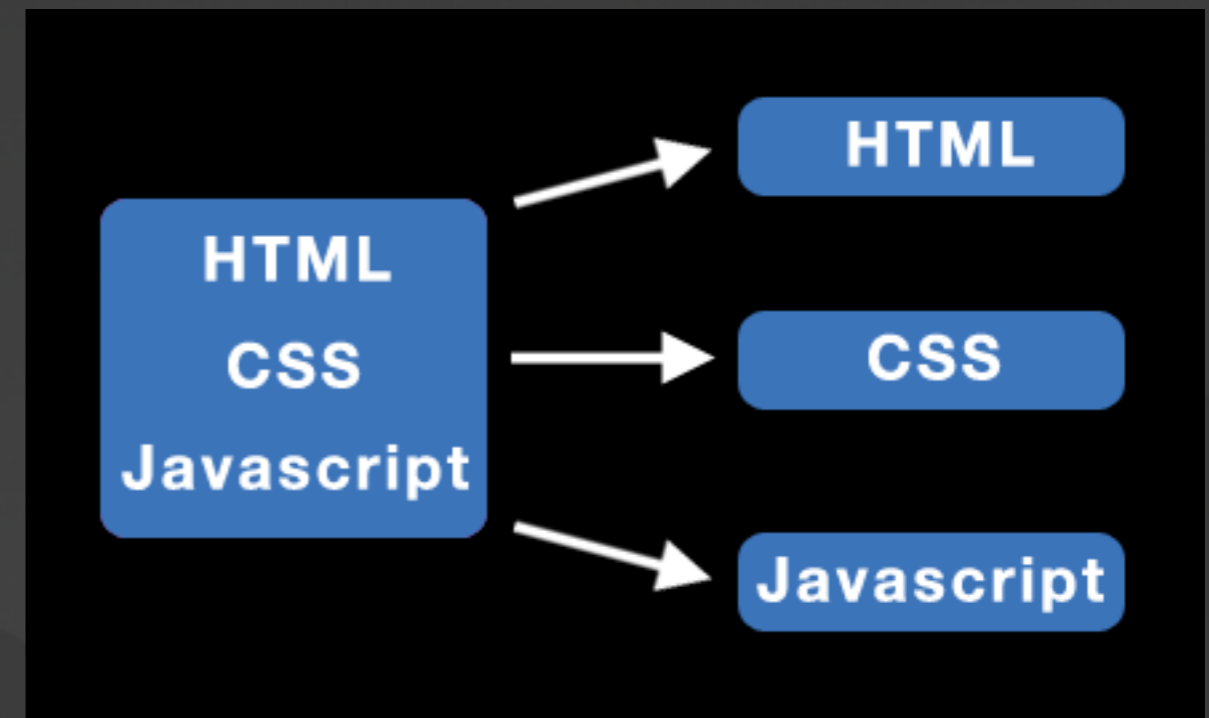
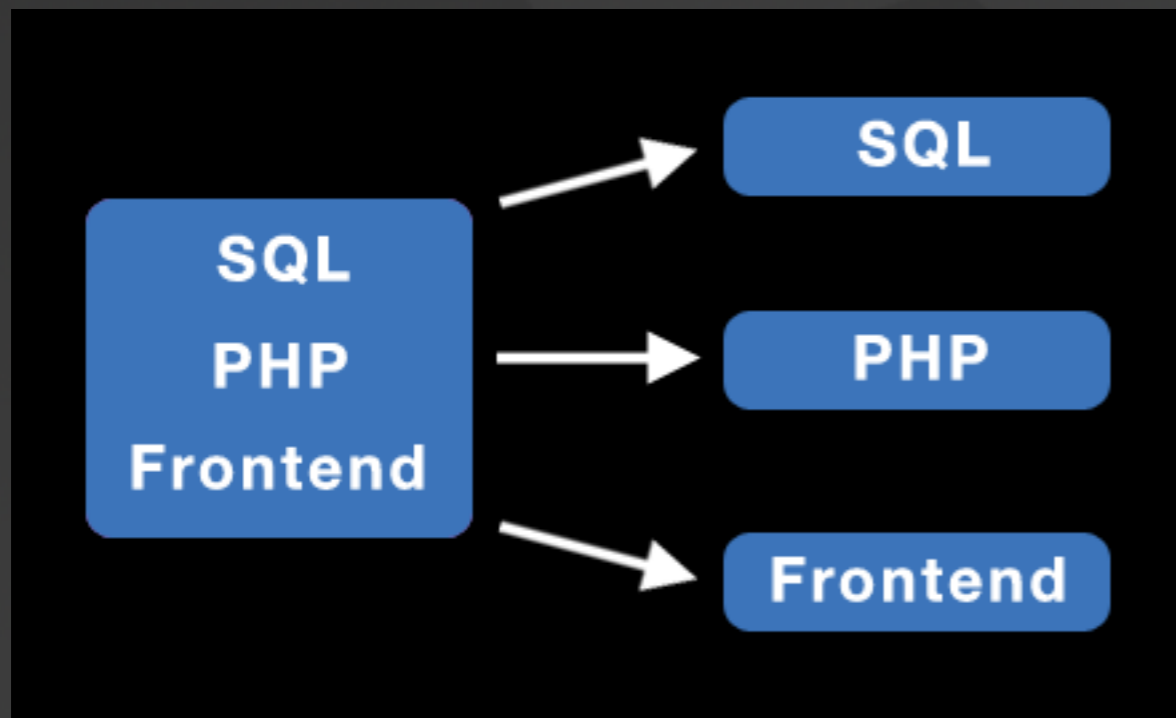
Application Developer

- Client-side techniques you need to know:
 - XHTML and CSS
 - DOM
 - JavaScript, JSON, XML
 - Separation of Concerns

Application Developer

- Client-side techniques have evolved along with the server side.
- You need to understand them regardless of your specific role as a web developer.

Separating Concerns



Separate both server-side and client-side code by type and responsibility.

Today's Application

- XHTML, CSS, Javascript cleanly separated into different files.
- The DOM lets all of these client-side technologies hang together cleanly.
- CSS Selectors and newer JavaScript libraries make this much easier

Testability

- Your markup determines how much fun you will have doing your testing.
- You need to understand markup, DOM, selectors, and then design for testability.
- The DOM is an integration point for your tests, through selectors.

CSS Selector Basics

CSS Selectors

```
#users li.user {  
  color: #000;  
}
```

- Simple, convenient
- Have become the standard for accessing DOM, driven by CSS
- You don't need to be a designer, you do need be able to read this selector and write others like it.

CSS Selectors

- All inlined CSS and JS has been removed.
- Readable by human or machine.
- IDs must begin with alpha character and may only be used once.
- A class can be used multiple times.

```
<ul id="users">  
  <li class="user" id="user_1">  
    <a href="/users/1">Jeff</a>  
  </li>  
  <li class="user" id="user_2">  
    <a href="/users/2">Walter</a>  
  </li>  
</ul>
```

CSS Selectors

- Strategically placed IDs and class names to easily reference elements individually or as a group.
- Specific elements get an ID.
- Elements you're looping over or that you'll reference as a collection can use a class.

```
<ul id="users">
  <li class="user" id="user_1">
    <a href="/users/1">Jeff</a>
  </li>
  <li class="user" id="user_2">
    <a href="/users/2">Walter</a>
  </li>
</ul>
```

CSS Selectors

```
<ul id="users">
  <li class="user" id="user_1">
    <a href="/users/1">Jeff</a>
  </li>
  <li class="user" id="user_2">
    <a href="/users/2">Walter</a>
  </li>
</ul>
```

```
// JQuery:
$("#users li.user").addClass("highlight");

// Prototype:
$$('#users li.user').invoke("addClassName", "highlight");
```

- CSS selectors and newer JS libraries make life much easier.
- Your PHP code can use the same selectors for testing!

PHPUnit SeleniumTestCase

Selenium RC

- Browser-based testing tool
 - Launches a web browser
 - Retrieves URL
 - Inspects Results
- PHPUnit Integration is Simple to Use

Selenium RC

- Download and install Selenium Server
- Launch Selenium Server on Command Line
- Run PHPUnit Tests utilizing Selenium
- Shut down Selenium Server

First Selenium Test

```
class WebTest extends PHPUnit_Extensions_SeleniumTestCase
{
    protected function setUp()
    {
        $this->setBrowser('*firefox');
        $this->setBrowserUrl('http://www.example.com/');
    }

    public function testTitle()
    {
        $this->open('http://www.example.com/');
        $this->assertTitleEquals('Example Web Page');
    }
}
```

Selenium Assertions

Assertion	Meaning
<code>void assertEquals()</code>	Reports an error if no alert is present.
<code>void assertEqualsNoAlertPresent()</code>	Reports an error if an alert is present.
<code>void assertEqualsChecked(string \$locator)</code>	Reports an error if the element identified by \$locator is not checked.
<code>void assertEqualsNotChecked(string \$locator)</code>	Reports an error if the element identified by \$locator is checked.
<code>void assertEqualsConfirmationPresent()</code>	Reports an error if no confirmation is present.
<code>void assertEqualsNoConfirmationPresent()</code>	Reports an error if a confirmation is present.
<code>void assertEqualsEditable(string \$locator)</code>	Reports an error if the element identified by \$locator is not editable.
<code>void assertEqualsNotEditable(string \$locator)</code>	Reports an error if the element identified by \$locator is editable.
<code>void assertEqualsElementValueEquals(string \$locator, string \$text)</code>	Reports an error if the value of the element identified by \$locator is not equal to the given \$text.
<code>void assertEqualsElementValueNotEquals(string \$locator, string \$text)</code>	Reports an error if the value of the element identified by \$locator is equal to the given \$text.
<code>void assertEqualsElementContainsText(string \$locator, string \$text)</code>	Reports an error if the element identified by \$locator does not contain the given \$text.
<code>void assertEqualsElementNotContainsText(string \$locator, string \$text)</code>	Reports an error if the element identified by \$locator contains the given \$text.
<code>void assertEqualsElementPresent(string \$locator)</code>	Reports an error if the element identified by \$locator is not present.
<code>void assertEqualsElementNotPresent(string \$locator)</code>	Reports an error if the element identified by \$locator is present.
<code>void assertEqualsLocationEquals(string \$location)</code>	Reports an error if the current location is not equal to the given \$location.
<code>void assertEqualsLocationNotEquals(string \$location)</code>	Reports an error if the current location is equal to the given \$location.
<code>void assertEqualsPromptPresent()</code>	Reports an error if no prompt is present.
<code>void assertEqualsNoPromptPresent()</code>	Reports an error if a prompt is present.
<code>void assertEqualsIsSelected(string \$selectLocator, string \$value)</code>	Reports an error if the given value is not selected.
<code>void assertEqualsIsNotSelected(string \$selectLocator, string \$value)</code>	Reports an error if the given value is selected.
<code>void assertEqualsSomethingSelected(string \$selectLocator)</code>	Reports an error if the option identified by \$selectLocator is not selected.
<code>void assertEqualsNothingSelected(string \$selectLocator)</code>	Reports an error if the option identified by \$selectLocator is selected.
<code>void assertEqualsTextPresent(string \$pattern)</code>	Reports an error if the given \$pattern is not present.
<code>void assertEqualsTextNotPresent(string \$pattern)</code>	Reports an error if the given \$pattern is present.
<code>void assertEqualsTitleEquals(string \$title)</code>	Reports an error if the current title is not equal to the given \$title.
<code>void assertEqualsTitleNotEquals(string \$title)</code>	Reports an error if the current title is equal to the given \$title.
<code>void assertEqualsVisible(string \$locator)</code>	Reports an error if the element identified by \$locator is not visible.
<code>void assertEqualsNotVisible(string \$locator)</code>	Reports an error if the element identified by \$locator is visible.

Selenium Assertions

- Fairly rich assertion vocabulary with specific assertions like `assertTitleEquals()`
- General purpose element assertions like `assertElementPresent()` take `$locator`
- Element locators can be a number of formats such as `XPath`.

Selenium Assertions

```
public function testTitle()  
{  
    $this->open('http://www.example.com/');  
    $this->assertElementValueEquals('css=title', 'Example Web Page');  
}
```

- Locators can be CSS selectors! “css=title”
- Use \$locator with CSS selectors where possible, keeping your test conventions congruent with your CSS and JavaScript

Selenium Disadvantages

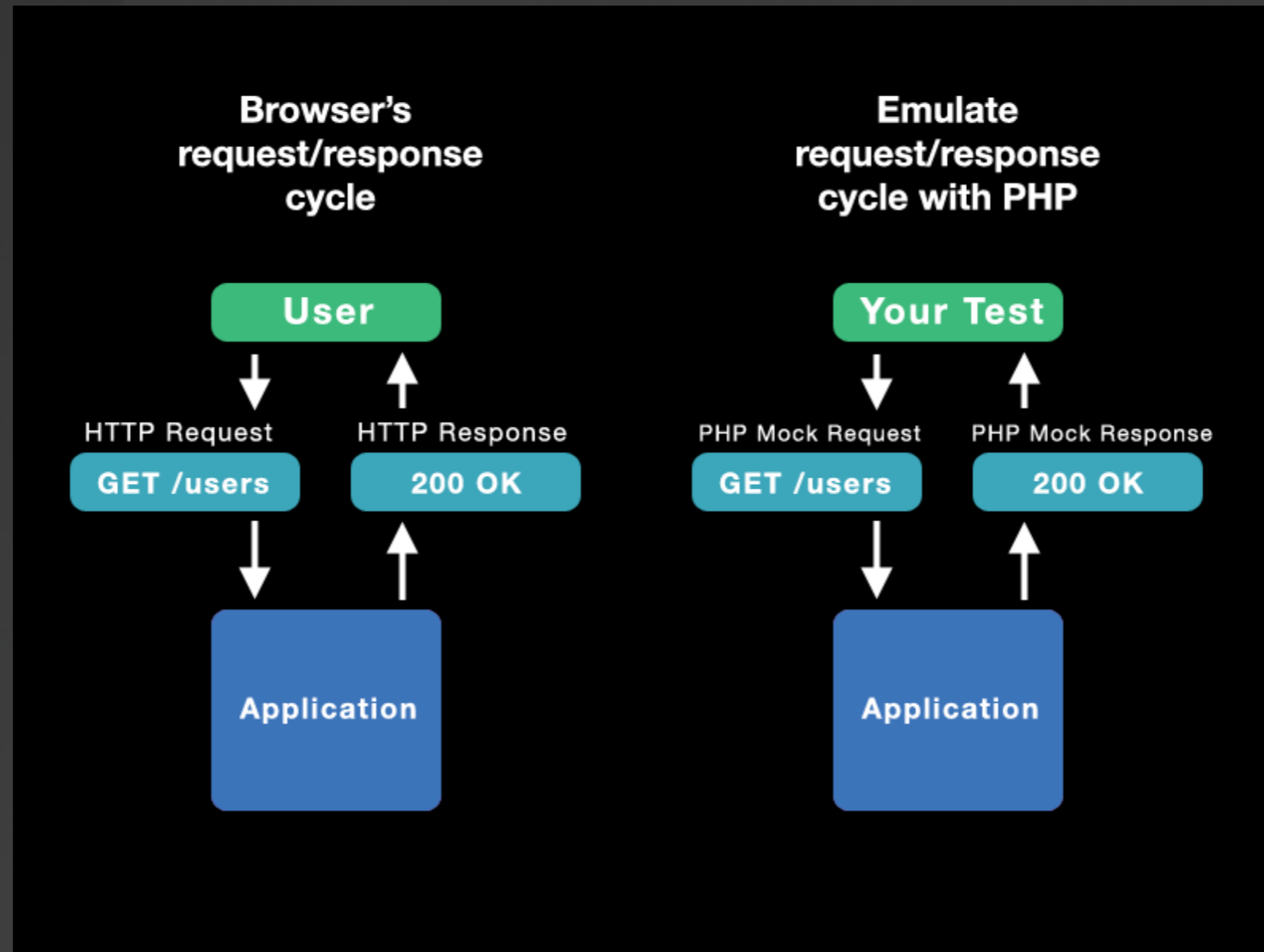
- Launching browser is too slow to be fun
- Somewhat fragile due to moving parts
- Falls down where browser falls down

Roll Your Own

Roll Your Own

- Functional or integration tests that don't depend on HTTP or the browser
- Some PHP frameworks may already have what you need, borrow from them
- More work to build the test harness, but testing is faster and more fun when done

Roll Your Own



Roll Your Own

- Your application probably needs some sort of request and response objects
- Subclass `PHPUnit_Framework_TestCase`

Roll Your Own

```
/**
 * @group functional
 */
class ItemsControllerTest extends Mad_Test_Functional {
  public function setUp()
  {
    $this->request = new Mad_Controller_Request_Mock();
    $this->response = new Mad_Controller_Response_Mock();

    $this->session = array('version' => Session::VERSION,
                          'user_id' => 1);

    $this->loadFixtures('Items');
  }

  public function testIndexDisplaysItems()
  {
    $this->get('/', array(), $this->session);

    $this->assertResponse(200);
    $this->assertSelect("#eng_items tr.eng_item", 4);

    // no clear filter button
    $this->assertSelect('h2#filters img', false);
  }
}
```



Roll Your Own

- Add methods to make fake GET, POST, PUT, DELETE requests
- Add assertions for response code and body
- <http://framework.maintainable.com> has test code you can use as a starting point
- PHPUnit 3.3 CSS Selector Assertions

PHPUnit 3.3

- Patch almost ready, based on our existing work
- Simple assertions for CSS selectors on strings:
 - `assertSelectCount()` / `assertSelectNotCount()`
 - `assertSelectEquals()` / `assertSelectNotEquals()`
 - `assertSelectRegexp()` / `assertSelectNotRegexp()`

Extras

JavaScript Unit Tests

- JavaScript can be unit-tested in a browser if it is sufficiently separated from HTML
- The unittest.js library from Scriptaculous is a nice solution for this

JavaScript Unit Tests

```
// cells.js

var Cell = Class.create({
  initialize: function(element) {
    this.element = $(element);
  },
  select: function() {
    this.element.addClassName('selected');
  },
  deselect: function() {
    this.element.removeClassName('selected');
  },
  selected: function() {
    return this.element.hasClassName('selected');
  }
});
```



Javascript Unit Tests

```
// inside cells_test.html

new Test.Unit.Runner({
  setup: function() {
  },
  teardown: function() {
  },

  testSelectCell: function() { with(this) {
    var cell = new Cell('cell_1');
    assert(!cell.selected());

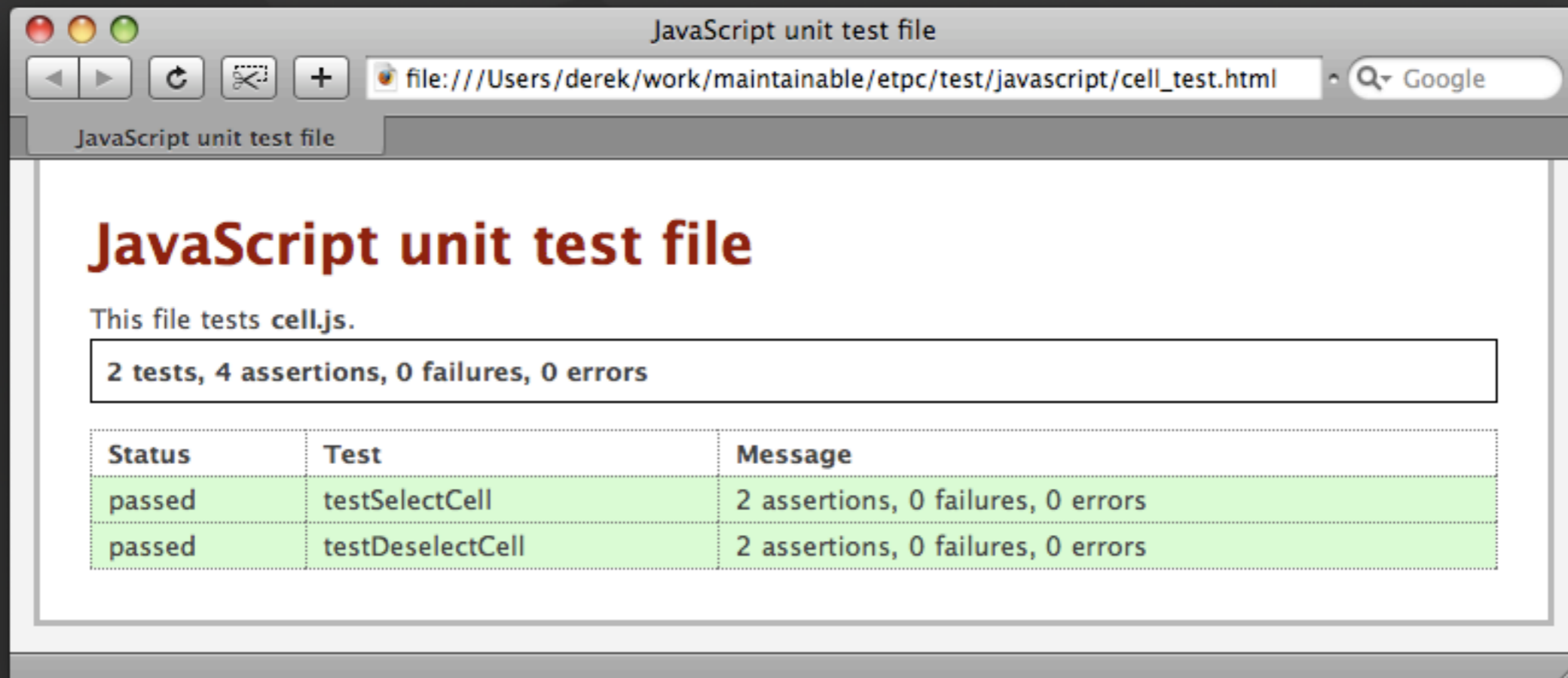
    cell.select();
    assert(cell.selected());
  }},

  // ...

}, "testlog");
```



JavaScript Unit Tests



The screenshot shows a web browser window titled "JavaScript unit test file". The address bar contains the file path: `file:///Users/derek/work/maintainable/etpc/test/javascript/cell_test.html`. The page content includes a title "JavaScript unit test file" and a subtitle "This file tests cell.js.". Below this, a summary box states "2 tests, 4 assertions, 0 failures, 0 errors". A table follows, detailing the test results.

Status	Test	Message
passed	testSelectCell	2 assertions, 0 failures, 0 errors
passed	testDeselectCell	2 assertions, 0 failures, 0 errors

Inline PHP Errors

Home Engineering ▾ Inventory ▾

Items

Notice: Undefined variable: foo in `/path/to/app/views/EngItems/index.html.php` on line 13

IN Item List

Search Filters

Item Number	<input type="text"/>	Inv Commodity Code	<input type="text"/>
Description	<input type="text"/>	Responsible Engineer	<input type="text"/>

Other Search Filters

1 - 4 of 4 Items

	Div	Item Number ▾	Description	Commodity Code	Type	Source	Stat
view	00001	COMPITEM1	Component Item 1		Standard	Purchased	Active
view	00001	COMPITEM2	Component Item 2		Standard	Purchased	Active
view	00001	PARENT2	Parent Item 1		Standard	Manufactured	Active
view	00001	PARENT2	Parent Item 2		Standard	Manufactured	Active

Inline PHP Errors

- Run your application with `display_errors=On` during tests, off in production.
- PHP errors output on a page are easily missed by tests if not explicitly checked.
- Test response body for “`Notice`” etc.

Errors as Exceptions

```
class YourPrefix_ErrorHandler
{
    public static function handle($errno, $errstr, $errfile, $errline)
    {
        if (ini_get('error_reporting') == 0) {
            // silence operator ("@") was used
            return;
        }

        throw new Exception($errstr, $errno);
    }
}

$callback = array('YourPrefix_ErrorHandler', 'handle');
set_error_handler($callback);
```



Errors as Exceptions

PHP Notice in ItemsController->index

Undefined variable: foo

```
app/views/EngItems/index.html(13) in include
vendor/Mad/View/Base.php(305): include
vendor/Mad/View/Base.php(215): Mad_View_Base->_template
vendor/Mad/Controller/Base.php(494): Mad_View_Base->render
vendor/Mad/Controller/Base.php(417): Mad_Controller_Base->renderAction
vendor/Mad/Controller/Base.php(244): Mad_Controller_Base->render
vendor/Mad/Controller/Dispatcher.php(100): Mad_Controller_Base->process
public/index.php(7): Mad_Controller_Dispatcher->dispatch
```

Extracted source (around [line #13](#))

```
10: </span>
11: </hl>
12:
13: <?=$foo ?>
14:
15: <div id="subnav" class="clr">
16: <ul id="breadcrumbs">
```

Request

Parameters:

None

[Show session dump](#)

Response

Headers:

array (

Q & A

Thank You

- <http://mikenaberezny.com>
- <http://maintainable.com>