TEST WITH KARMA AND JASMINE

For those unfamiliar with Karma, it is a JavaScript test runner that is test framework agnostic. The Angular generator has two included test frameworks: ngScenario and Jasmine. When we ran `yo angular` earlier in this codelab the generator scaffolded a ***test*** directory in the root of the ***mytodo*** folder, created a ***karma.conf*** file, and pulled in the Node modules for Karma. We'll be editing a Jasmine script to describe our tests soon but let's see how we can run tests first.

RUN UNIT TESTS

Let's go back to the command line and kill our Grunt server using Ctrl + C. There is already a Grunt task scaffolded out in our ***Gruntfile.js*** for running tests. It can be run as follows:

```
$ grunt test
```

When you run `grunt test`, you will see a new browser window quickly open and close, plus some warnings in the Yeoman console. Don't worry, that's to be expected right now since our tests are currently failing for two reasons. Let's fix that.

UPDATE UNIT TESTS

First, we need to update the Karma configuration to load the the new Bower components we installed in the Step 7.

Open ***karma.conf.js*** and replace the `files` array with:

```
 files: [
  'app/bower_components/jquery/jquery.js',
  'app/bower_components/jquery-ui/ui/jquery-ui.js',
  'app/bower_components/angular/angular.js',
  'app/bower_components/angular-ui-sortable/sortable.js',
  'app/bower_components/angular-mocks/angular-mocks.js',
  'app/bower_components/angular-local-storage/angular-local-storage.js',
  'app/scripts/*.js',
  'app/scripts/**/*.js',
  'test/mock/**/*.js',
  'test/spec/**/*.js',
  'app/bower_components/angular-resource/angular-resource.js',
  'app/bower_components/angular-cookies/angular-cookies.js',
  'app/bower_components/angular-sanitize/angular-sanitize.js',
  'app/bower_components/angular-route/angular-route.js'
],
```

Second, we haven't updated the boilerplate test which still references `awesomeThings`.

You'll find the tests scaffolded out in the ***test*** folder, so open up **test/spec/controllers/main.js**. This is the unit test for your Angular MainCtrl controller that we need to modify.

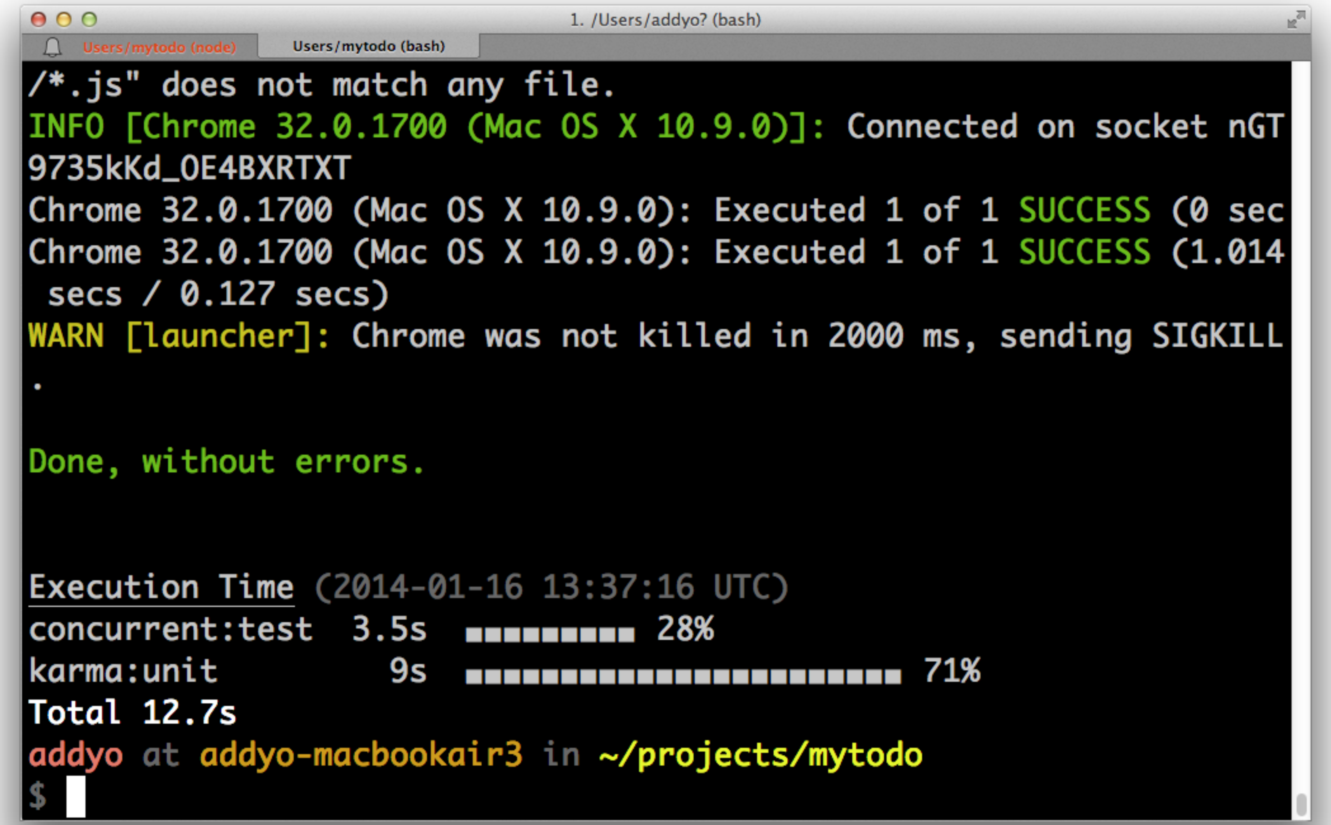
Delete the following:

```
it('should attach a list of awesomeThings to the scope', function () {
  expect(scope.awesomeThings.length).toBe(3);
});
```

And replace that test with the following:

```
it('should have no items to start', function () {
    expect(scope.todos.length).toBe(0);
});
```

Re-running our tests with `grunt test` should see our tests now passing:



```
/*.js" does not match any file.
INFO [Chrome 32.0.1700 (Mac OS X 10.9.0)]: Connected on socket nGT
9735kKd_OE4BXRTXT
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 1 of 1 SUCCESS (0 sec
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 1 of 1 SUCCESS (1.014
 secs / 0.127 secs)
WARN [launcher]: Chrome was not killed in 2000 ms, sending SIGKILL
.

Done, without errors.


Execution Time (2014-01-16 13:37:16 UTC)
concurrent:test  3.5s  ■■■■■■■■■ 28%
karma:unit         9s  ■■■■■■■■■■■■■■■■■■■■■■■ 71%
Total 12.7s
addyo at addyo-macbookair3 in ~/projects/mytodo
$
```

The above test only covers part of our app's functionality so let’s add a couple of more tests to test the adding and removing of items:

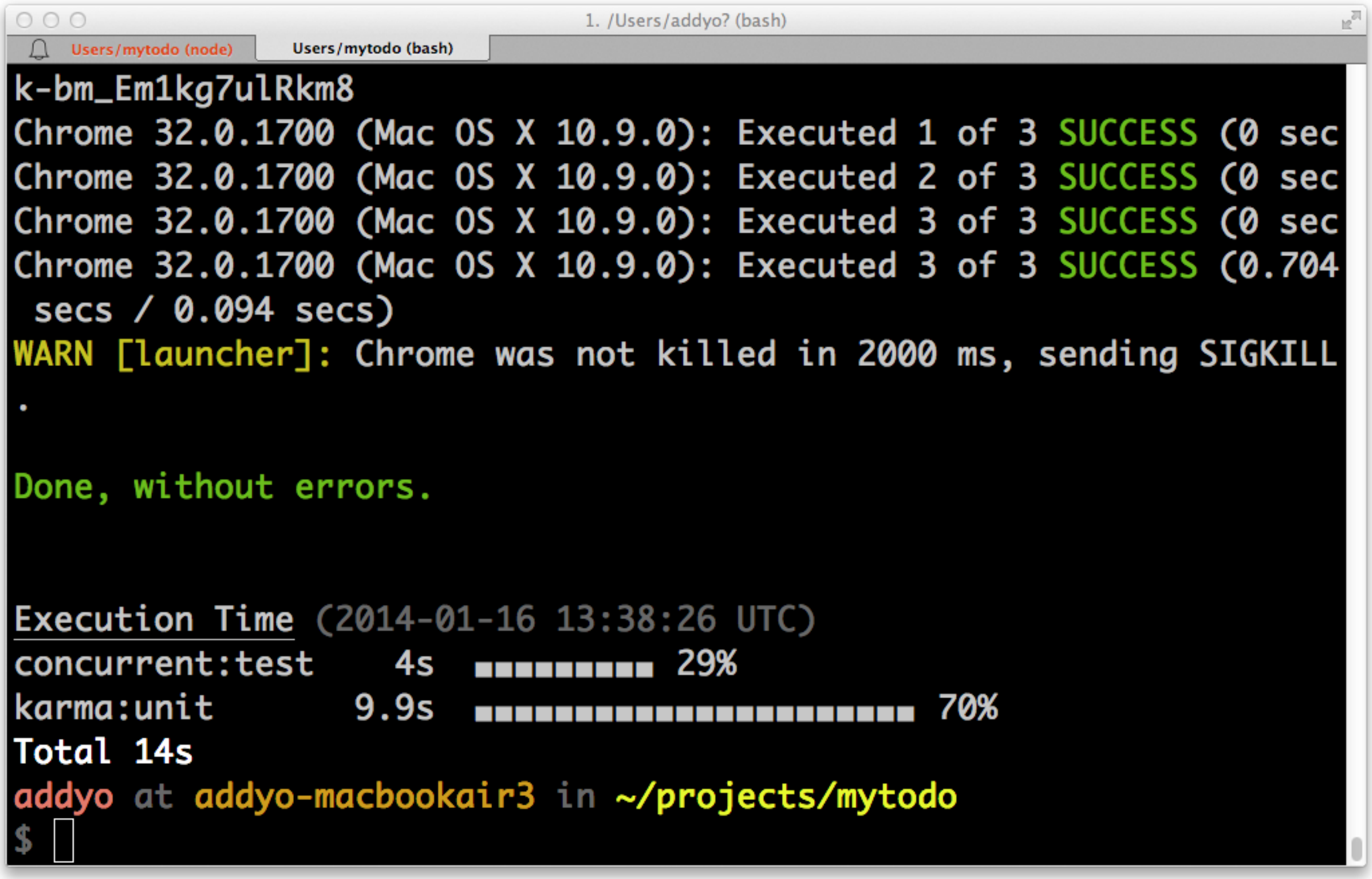
```
it('should add items to the list', function () {
    scope.todo = 'Test 1';
    scope.addTodo();
    expect(scope.todos.length).toBe(1);
  });

it('should add items to the list', function () {
    scope.todo = 'Test 1';
    scope.addTodo();
    scope.removeTodo(0);
    expect(scope.todos.length).toBe(0);
  });
```

Your full MainCtrl test script (*test/spec/controllers/main.js*) should now look like this:

```
'use strict';
```

```
describe('Controller: MainCtrl', function () {

  // load the controller's module
  beforeEach(module('mytodoApp'));

  var MainCtrl,
    scope;

  // Initialize the controller and a mock scope
  beforeEach(inject(function ($controller, $rootScope) {
    scope = $rootScope.$new();
    MainCtrl = $controller('MainCtrl', {
      $scope: scope
    });
  }));

  it('should have no items to start', function () {
    expect(scope.todos.length).toBe(0);
  });

  it('should add items to the list', function () {
    scope.todo = 'Test 1';
    scope.addTodo();
    expect(scope.todos.length).toBe(1);
  });

  it('should add items to the list', function () {
    scope.todo = 'Test 1';
    scope.addTodo();
    scope.removeTodo(0);
    expect(scope.todos.length).toBe(0);
  });

});
```

Running the tests again, we should see everything pass with 3 tests being executed:



```
k-bm_Em1kg7ulRkm8
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 1 of 3 SUCCESS (0 sec
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 2 of 3 SUCCESS (0 sec
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 3 of 3 SUCCESS (0 sec
Chrome 32.0.1700 (Mac OS X 10.9.0): Executed 3 of 3 SUCCESS (0.704
 secs / 0.094 secs)
WARN [launcher]: Chrome was not killed in 2000 ms, sending SIGKILL
.

Done, without errors.


Execution Time (2014-01-16 13:38:26 UTC)
concurrent:test    4s  ■■■■■■■■■ 29%
karma:unit       9.9s  ■■■■■■■■■■■■■■■■■■■■■■ 70%
Total 14s
addyo at addyo-macbookair3 in ~/projects/mytodo
$ 
```

Fantastic!

Writing unit tests make it easier to catch bugs as your app gets bigger and when more developers join your team. The scaffolding feature of Yeoman makes writing unit tests easier so no excuse for not writing your own tests! ;)

« Return to overview or **Go to the next step »**