



Fork me on GitHub

# PRACTICAL

# AngularJS

Based on blog posts by

Dinis Cruz

beta version

# Practical AngularJS

Real-world examples of using AngularJS in multiple scenarios and platforms (KarmaJS, Firebase, Eclipse, WebStorm, O2 Platform)

Dinis Cruz

This book is for sale at [http://leanpub.com/Practical\\_AngularJS](http://leanpub.com/Practical_AngularJS)

This version was published on 2014-03-29



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a [Creative Commons Attribution 3.0 Unported License](#)

## **Tweet This Book!**

Please help Dinis Cruz by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#PracticalAngularJS](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

<https://twitter.com/search?q=#PracticalAngularJS>

# Contents

<b>Introduction</b> . . . . .	<b>i</b>
Change log: . . . . .	ii
<b>1 Using AngularJS</b> . . . . .	<b>1</b>
1.1 A really SIMPLE and clean AngularJS+Firebase example . . . . .	2
1.2 Using AngularJS in Eclipse, Part 1) The Basics . . . . .	8
1.3 Using AngularJS in Eclipse, Part 2) Add Some Control . . . . .	21
1.4 Using AngularJS in Eclipse, Part 3) Wire up a Backend . . . . .	36
1.5 Using AngularJS in Eclipse, Part 4) Create Components . . . . .	50
1.6 AngularJS code editor using UI-Bootstrap and CodeMirror (done without using jQuery) . . . . .	58
<b>2 KarmaJS</b> . . . . .	<b>62</b>
2.1 A small AngularJS Jasmine test executed by KarmaJS . . . . .	63
2.2 Creating an Eclipse UI to run AngularJS e2e tests using Karma . . . . .	72
2.3 Running KarmaJS's AngularJS example test/e2e/angular-scenario (on Chrome) . . . . .	82
<b>3 Firebase</b> . . . . .	<b>90</b>
3.1 First PoC of sending TeamMentor's server-side request URLs to Firebase (and seeing it in realtime in an AngularJS page) . . . . .	91
3.2 Trying out Firebase (Beta) hosting solution and good example of Firebase Security rules . . . . .	94
<b>4 Misc Tricks</b> . . . . .	<b>102</b>
4.1 Programatically changing an AngularJS scope variable and adding Firebug Lite to an AngularJS app . . . . .	103
4.2 Hubspot current.js code includes JQuery on it . . . . .	107
4.3 Submitting TM users to HubSpot via TBOT interface (using Angular JS) . . . . .	110
<b>5 IDEs</b> . . . . .	<b>112</b>
5.1 Eclipse Groovy REPL script to sync a Browser with file changes (with recursive folder search via Java's WatchService) . . . . .	113
5.2 Eclipse Groovy script to remove the 'busy' image from the WebBrowser Editor . . . . .	116
5.3 Using Chrome inside a native VisualStudio pane (using Window Handle Hijacking) . . . . .	122
5.4 Using WebStorm with Chrome and ChromeDriver (to view KarmaJS execution results) . . . . .	126
5.5 When the best way to automate Chrome is to use ... Chrome (with examples on Google search, direct AngularJS scope manipulation and ChromeDriver javascript access) . . . . .	129
5.6 Adding KarmaJS support to WebStorm to automagically run tests on file changes (and test UI with SublimeText, Chrome and Cmd.exe) . . . . .	137

## CONTENTS

<b>6 Troubleshooting . . . . .</b>	<b>144</b>
6.1 KarmaJS AngularJS Scenario Test Runner execution variations in IE 7,8,9 and 10 when using AngularJS . . . . .	145
6.2 If AngularJS doesn't work on your O2 Platform IE scripts (the fix is to change browser compatibility mode) . . . . .	149
6.3 Debugging a weird case of missing module in AngularJS and KarmaJS . . . . .	151
<b>Appendix Post's Details . . . . .</b>	<b>156</b>

# Introduction

This book contains the AngularJS related blog posts posted on Dinis Cruz' blog at <http://blog.diniscruz.com>.

This is the first draft release of this book, so please send your suggestions, criticisms, ideas or comments to [dinis.cruz@owasp.org](mailto:dinis.cruz@owasp.org)

## Notes about current structure

The first version of this book had the chapter order created by the original ‘import from blogger’ (i.e. by publish order).

In the current version, the posts are split into the following areas: “Using AngularJS”, “KarmaJS”, “Firebase”, “Misc Tricks”, “IDEs”, “Troubleshooting”, “Appendices”, which is a filter based on technology. Note that this can change based on reader feedback, so if you think the order should be different, please send your feedback and ideas.

## About the Author

Dinis Cruz is a Developer and Application Security Engineer focused on how to develop secure applications. A key drive is on ‘Automating Application Security Knowledge and Workflows’ which is the main concept behind the OWASP O2 Platform and Security Innovation’s TeamMentor (Dinis is the main developer and architect of both Applications). Current day job is with Security Innovation where Dinis tries to promote openness, quality and sharing as part a core tenet of TeamMentor’s application development environment.

After many years (and multiple roles) Dinis is still very active at OWASP, currently leading the O2 Platform project and helping out other projects and initiatives.

After failing to scale his own security knowledge, learned Git, created security vulnerabilities in code published to production servers, delivered training to developers, and building multiple CI (Continuous Integration) environments; Dinis had the epiphany that the key to application security is “Secure Continuous Delivery: Developer’s Immediate Connection to What They’re Creating”. This ‘Immediate Connection/Feedback’ concept is deep rooted in the development of the O2 Platform/TeamMentor, and is something that will keep Dinis busy for many years.

---

[Table of Contents](#) | [Code](#)

## Change log:

In 2014, here are the changes made (per version):

- **v0.20 (29 March)**
  - updated Readme.md file to point to existing articles.
  - renamed all content files to \*.md (which make sense since they are Markdown files, and they look much better at GitHub) and gave them a name that is related to its current chapter
  - new files: C0-Introduction/Table\_of\_contents.md
  - created a GitHub-only ‘table of contents’ which links to all available chapters. Each chapter intro page also has the links for its child articles. Added some navigation links to the end of each article (to allow easy navigation when browsing the content in GitHub)
- **v0.13 (16th March)**
  - added new cover to eBook version
  - lots of formating changes to most chapters/posts (specialy on current source code samples)
  - added leanpub attributes for: frontmatter, mainmatter, backmatter
  - major rename of markdown file names to help to quickly see which file belongs to what chapter (see github repo)
- **v0.12 (11th March)**
  - fixed spellings
  - fixed nested list in this change log (needs 4 spaces on line start)
  - fixed use of sections in ‘Appendix Posts Details’
  - created github repository for this book: [https://github.com/DinisCruz/Book\\_Practical\\_AngularJS](https://github.com/DinisCruz/Book_Practical_AngularJS)
- **v0.11 (11th March)**
  - Create the following main section: “Using AngularJS”, “KarmaJS”, “Firebase”, “Misc Tricks”, “IDEs”, “Troubleshooting”, “Appendices”
  - Changed the order of the chapters (using the new sections created, instead of the original import structure (by month))
  - Book configuration change to add page break for every section (i.e. blog post)
  - Added this change log
  - Created Git repo on local dropbox sync folder
  - Added Appendix “Post’s Details”
- **v0.10 (10th March)**
  - First release of book with raw import from blogger posts (no formatting or editing done)

# 1 Using AngularJS

This section has the following chapters:

- [A really SIMPLE and clean AngularJS+Firebase example](#)
  - [Using AngularJS in Eclipse, Part 1\) The Basics](#)
  - [Using AngularJS in Eclipse, Part 2\) Add Some Control](#)
  - [Using AngularJS in Eclipse, Part 3\) Wire up a Backend](#)
  - [Using AngularJS in Eclipse, Part 4\) Create Components](#)
  - [AngularJS code editor using UI-Bootstrap and CodeMirror \(done without using jQuery\)](#)
- 

GitHub Navigation: [Table of Contents](#) | [Code](#)

## 1.1 A really SIMPLE and clean AngularJS+Firebase example

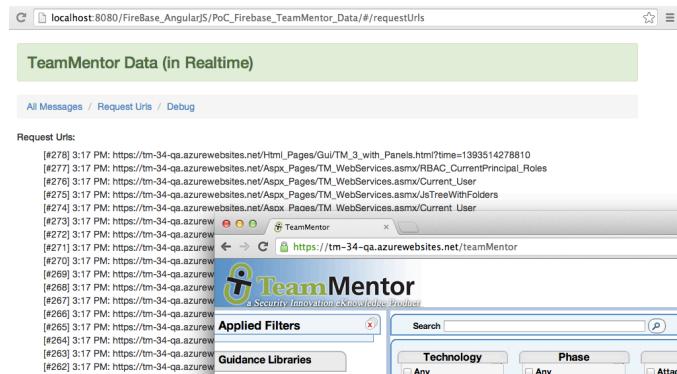
As seen on the [First PoC of sending TeamMentor's server-side request URLs to Firebase \(and seeing it in realtime in an AngularJS page\)](#) I created a *Simple* AngularJS website which I'm very happy with (and I mean *Simple* with a capital S).

The main reason I really like the solution shown below, is because it represents a number of really nice, clean and *Simple* solutions for common (complex) problems that exist while developing in Javascript.

The created application is an:

- AngularJS real-time viewer for HTTP requests,
- ... made to an ASP.NET web application ([TeamMentor](#)),
- ... captured by an custom [C# HttpHandler filter](#),
- ... submitted to [Firebase](#) using its REST API and
- ... pushed back to the AngularJS app using open HTML 5 WebSockets.

The image below shows what the AngularJS+Firebase application looks like, with the urls shown in the background browser being the ones requested when the TeamMentor website is loaded or navigated (note that the latency between '*request made*' \_and\_ '*request listed*' is really small (about ~ 50 milliseconds)):



What I also like about the AngularJS structure that I ended up with, is that it represents a great way to learn AngularJS' architecture and capabilities (and yes I know and agree that for bigger AngularJS apps it is better to [organise by feature](#) and group the multiple files under a dedicated folder (for example the login controller, service, factory, view and tests should all go under the same logical folder))

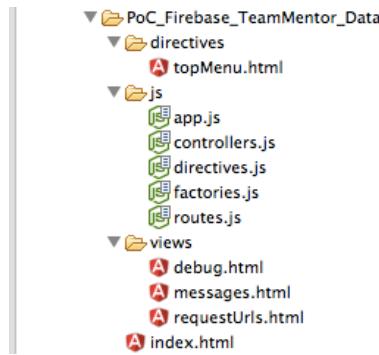
This post contains a tour of the multiple files created ([gist here](#)) which were developed/refactored while in Eclipse using the [Eclipse Groovy REPL script to sync a Browser with file changes \(with recursive folder search via Java's WatchService\)](#)

### Folder/File Structure:

Inside an Eclipse Web Static project, I created a file structure with:

- `_index.html` as the main AngularJS file (i.e. this is *the* single-page application file)

- all javascript files were placed in the `_js` \_folder (with the file names providing a clue on what they are doing/providing)
- 1 directive created in the `directives` folder
- 3 views placed on the `views` folder



## 1) Index.html

This is the file loaded directly by the browser, which is made of:

- External/Framework javascript includes: `angular.js`, `angular-route.js`, `firebase.js`, `angularfire.js`, `bootstrap.min.css`
- AngularJS javascript includes (for the current application): `app.js`, `factories.js`, `controllers.js`, `directives.js`, `routes.js`
- CSS link to `bootstrap.min.css`
- Html body (containing the AngularJS `_ng-app` \_directive) and:
  - div tag with `_container` \_css class
  - h3 tag with `alert-success` css class (which creates that nice green top banner)
  - the custom `**top-menu **directive` (using `_attribute` instead of `element`)
  - the AngularJS `ng-view` directive

```

<!doctype html>
<html ng-app="project">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.js" type="text/javascript"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-route.js" type="text/javascript"></script>
    <script src="https://cdn.firebaseio.com/0.firebaseio.js" type="text/javascript"></script>
    <script src="https://cdn.firebaseio.com/0.5.0/angularfire.js" type="text/javascript"></script>
    <link href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css" rel="stylesheet" >

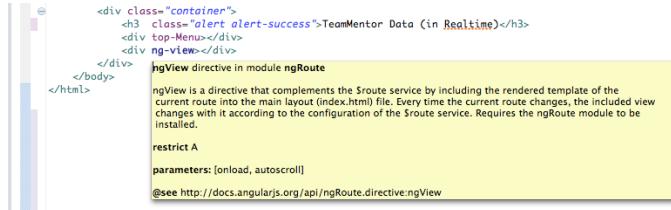
    <script src="js/app.js" ></script>
    <script src="js/factories.js" ></script>
    <script src="js/controllers.js" ></script>
    <script src="js/directives.js" ></script>
    <script src="js/routes.js" ></script>
  </head>
  <body>
    <div class="container">
      <h3 class="alert alert-success">TeamMentor Data (in Realtime)</h3>
      <div top-Menu></div>
      <div ng-view></div>
    </div>
  </body>
</html>
  
```

Since I'm using the [AngularJS Eclipse](#) plugin, hovering the mouse op top of an AngularJS directive provides a nice description of they do.

Here is the `_ng-app` \_directive



... and here is the `ng-view` directive:



## 2) app.js

This is where the `project` module is created (with two dependencies `ngRoute` and `firebase`).

Since I moved the controllers, factories, directives and routes into their own separate js file, there wasn't much to do here, apart from creating global values for the firebase URL and auth token (which will be dependency injected into the controllers and factories)



## 3) controllers.js

This file contains 3 controllers: `DebugCtrl`, `MessagesCtrl` and `_RequestsUrlCtrl` (each will be used on a specific view)

Note that each controller has services injected into them (the AngularJS `$scope` ***\*\*and the custom \*\* - fbDebugMsg, fbRequestUrl, fbaDebugUrl***)

The `DebugCtrl` *is currently just adding the injected \_fbDebugMsg and \*\*fbRequestUrl \*\*services into the \$scope so that we can see them in the view (this is a nice trick to get an inside view of AngularJS objects)*

The `MessagesCtrl` is using the Firebase [AngularFire API](#), which makes it really easy to create the firebase-real-time update view (my only problem with this was that there didn't seem to be an easy way to re-order the new elements (which in the current AngularFire implementation are added at the end of the provided array))

The `_RequestsUrlsCtrl` uses the default Firebase [Javascript API](#) (i.e not the AngularFire one) which gives us more control on how to handle the data received by Firebase. The `_$scope.urls` array is used to store the data received from the Firebase `_child_added` event (one note here to say that the Firebase `_child_added` will also provide the entire data-set on first load, which is okish, but I would prefer that the `_child_added` only fired for new events)



```

controllers.js ✘
var project = angular.module('project');

project.controller('DebugCtrl' , function($scope, fbDebugMsg, fbRequestUrl)
{
    $scope.fbDebugMsg = fbDebugMsg;
    $scope.fbRequestUrl = fbRequestUrl;
});

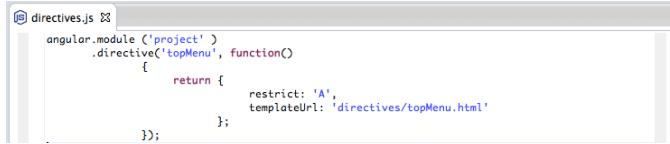
project.controller('MessagesCtrl' , function($scope, fbaDebugMsg)
{
    $scope.messages = fbaDebugMsg;
});

project.controller('RequestUrlsCtrl' , function($scope, fbRequestUrl)
{
    $scope.urls = [".... Loading data ..."];
    $scope.count = 0;
    fbRequestUrl.on("child_added", function(childSnapshot, prevChildName)
    {
        var text = "#[" + ++$scope.count + "] " + childSnapshot.val();
        $scope.urls.splice(0,0,text);
        $scope.$apply();
    });
    scope = $scope;
});

```

#### \*\*4) directives.js \*\*

This is a simple directive used by **index.html**, that will display a top menu, created by the content of the **topMenu.html** file (directives are AngularJS way to creating/defining new HTML tags/attributes)



```

directives.js ✘
angular.module('project')
.directive('topMenu', function()
{
    return {
        restrict: 'A',
        templateUrl: 'directives/topMenu.html'
    };
});

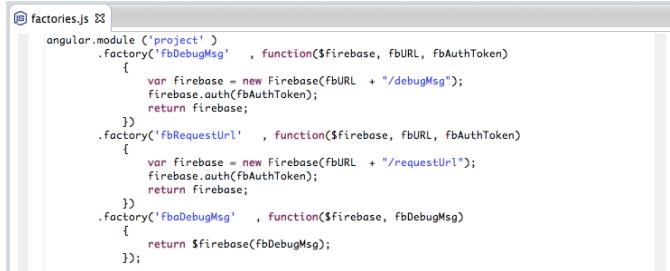
```

#### 5) factories.js

These factories create the Firebase mappings, namely they define the ‘area’ (or namespace/object) that the data will be read from.

The first two (***fbDebugMsg*** and ***fbRequestUrl***) use the Firebase [Javascript API](#). I needed to do them this way so that I could add the Firebase auth token (that said, I’m sure there is a better way to do this in Angular, since ideally I would have an Angular service that took the two variables that need to be set: the target Firebase area and auth code)

The ***\_fbaDebugMsg*** is just a simple service/factory to return an [AngularFire API](#) object based on the (dependency injected) ***fbDebugMsg*** service



```

factories.js ✘
angular.module('project')
.factory('fbDebugMsg' , function($firebase, fbURL, fbAuthToken)
{
    var firebase = new Firebase(fbURL + "/debugMsg");
    firebase.auth(fbAuthToken);
    return firebase;
})
.factory('fbRequestUrl' , function($firebase, fbURL, fbAuthToken)
{
    var firebase = new Firebase(fbURL + "/requestUrl");
    firebase.auth(fbAuthToken);
    return firebase;
})
.factory('fbaDebugMsg' , function($firebase, fbDebugMsg)
{
    return $firebase(fbDebugMsg);
});

```

#### 6) routes.js

The routes configuration is basically defining the 3 available views (each with a different controller mapped to it)

```

routes.js ✘
angular.module('project')
.config(function($routeProvider)
{
    $routeProvider.when('/messages', { templateUrl : 'views/messages.html',
                                      controller : 'MessagesCtrl' })
        .when('/requestUrls', { templateUrl : 'views/requestUrls.html',
                               controller : 'RequestUrlsCtrl' })
        .when('/debug', { templateUrl : 'views/debug.html',
                          controller : 'DebugCtrl' })
        .otherwise({ redirectTo : '/requestUrls' });
});

```

### \*\*7) requestUrls.html \*\* (view)

Since the *RequestsUrlsCtrl* \_is responsible for updating the `$scope.urls` \*\*array, all we need to do here is to use Angular's `ng-repeat` directive to create a list with all items (the `_list-unstyled_` class hides the bullet usually shown in HTML `<li>` tags).

Note that since the *RequestsUrlsCtrl* controller is using the Firebase [Javascript API](#) `child_added` event, we will see new entries shown in real time (ie. no browser refresh needed), but any changes made to existing items will not be reflected on the UI (unless the entire page is refreshed and the data is reloaded)

```

requestUrls.html ✘
<h5>Request Urls:</h5>
@<ul>
@<li ng-repeat="url in urls" class="list-unstyled">
@{url}
@</li>
</ul>

```

### 8) messages.html (view)

In this view the `$scope.messages` (used in the `ng-repeat`) is populated by the \*\*MessagesCtrl \*\*controller which is using the [AngularFire API](#). This means that data will be updated in real time (on both add and change events)

```

messages.html ✘
<h5>Current messages:</h5>
@<table class="table table-striped">
@<thead>
@<tr>
@<th>message</th>
@</tr>
@</thead>
@<tbody>
@<tr ng-repeat="message in messages" >
@<td>{message}</td>
@</tr>
@</tbody>
</table>

```

... which look like this:

The screenshot shows a web browser window titled "Synced WebBrowser". The main content area has a green header bar with the text "TeamMentor Data (in Realtime)". Below this, there is a navigation bar with links: "All Messages" / "RequestUrls" / "Debug". The main content area is labeled "Current messages:" and contains a list of messages. The messages are displayed in a table-like structure with a striped background. The messages listed are:

- mapping TMEvents.OnApplication\_BeginRequest to send url to Firebase
- 2:55 PM: aaaaa TMEvents.OnApplication\_BeginRequest to send url to Firebase
- 3:16 PM: mapping TMEvents.OnApplication\_BeginRequest to send url to Firebase
- 6:05 PM: mapping TMEvents.OnApplication\_BeginRequest to send url to Firebase

## 9) debug.html (view)

This view just shows a json representation of the `fbDebugMsg` and `_fbRequestUrl_`



```
<h4>Debug (Firebase objects)</h4>
<b>$fbDebugMsg:</b>


```

{{fbDebugMsg | json}}

```


<b>_fbRequestUrl:</b>

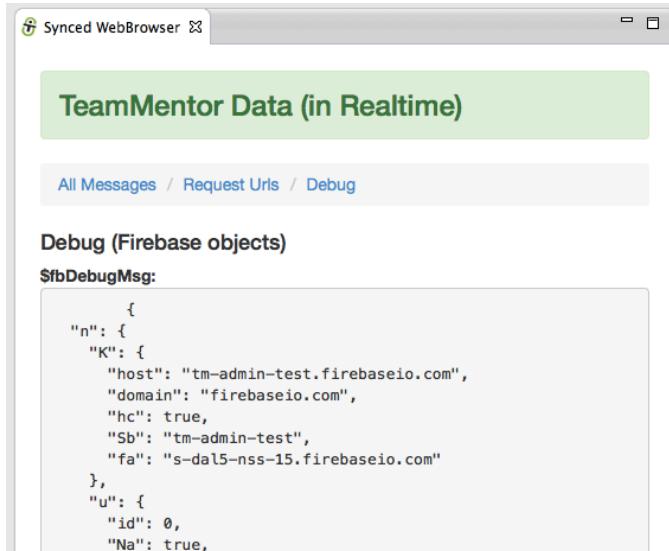

```

{{fbRequestUrl | json}}

```


```

... which looks like this:

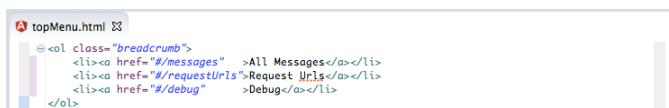


The browser window title is "Synced WebBrowser". The main content area has a green header bar with the text "TeamMentor Data (in Realtime)". Below it is a navigation bar with links: "All Messages" / "RequestUrls" / "Debug". The main content area is titled "Debug (Firebase objects)" and contains the JSON representation of \$fbDebugMsg:

```
{
  "n": {
    "K": {
      "host": "tm-admin-test.firebaseio.com",
      "domain": "firebase.com",
      "hc": true,
      "Sb": "tm-admin-test",
      "fa": "s-dal5-nss-15.firebaseio.com"
    },
    "u": {
      "id": 0,
      "Na": true,
    }
  }
}
```

## 10) topMenu.html (directive templateUrl)

Finally this is the html that creates the top menu (this could be improved/refactored by having the url and titles being provided as a **ng-model** object)



```
<ol class="breadcrumb">
  <li><a href="#/messages" >All Messages</a></li>
  <li><a href="#/requestUrls">Request Urls</a></li>
  <li><a href="#/debug" >Debug</a></li>
</ol>
```

All code:

For reference here is the entire source code ([gist here](#)) of the source code files shown above:

Btw, did you noticed anything different in the formatting of the code samples above?

Is it easier to read?

If you are interested in the topic of code formatting also see [On Java code formatting](#) and [Formatting code for readability](#)

## 1.2 Using AngularJS in Eclipse, Part 1) The Basics

This is the first of four posts on how to run (inside Eclipse) the examples provided in [AngularJS's home page](#):

- Using AngularJS in Eclipse, Part 1) The Basics
- Using AngularJS in Eclipse, Part 2) Add Some Control
- Using AngularJS in Eclipse, Part 3) Wire up a Backend
- Using AngularJS in Eclipse, Part 4) Create Components

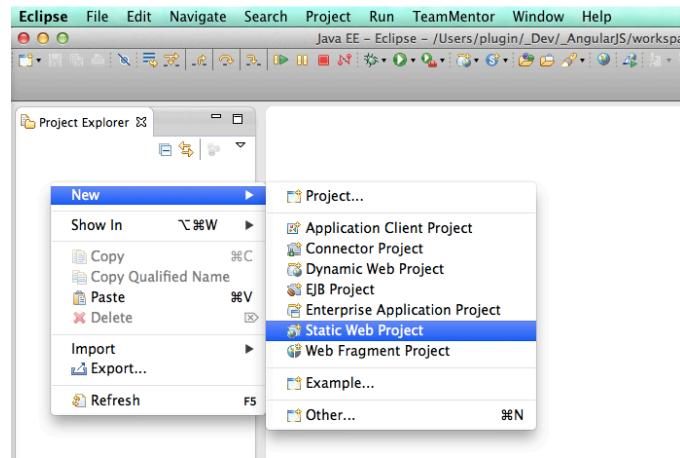
The example covered on this post is the *The Basics*.

I'm doing this on an OSX laptop and the first step was to download and unzip ([eclipse-standard-kepler-SR1-macosx-cocoa.tar.gz](#) (32bit version of Eclipse's Kepler) into the `~/_Dev/_AngularJS` folder.

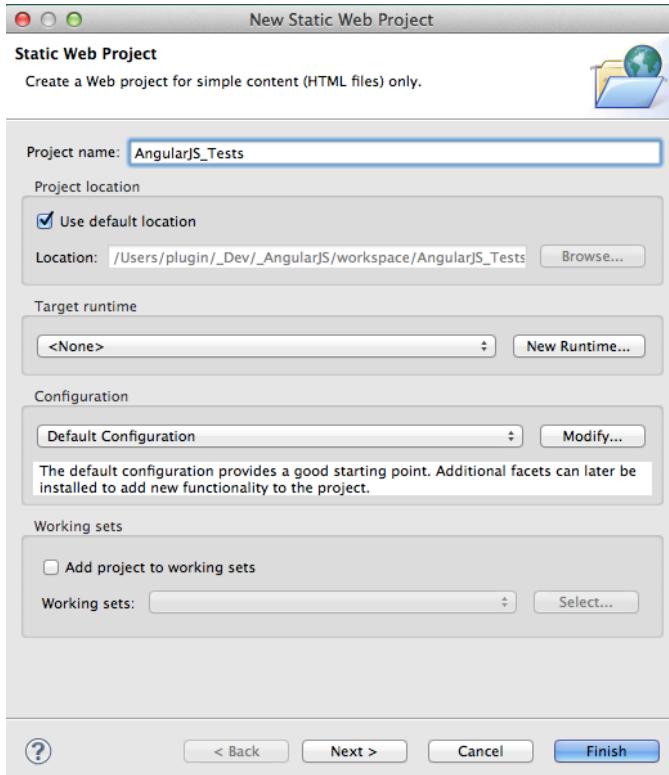
I fired up eclipse, chose the `~/_Dev/_AngularJS/workspace` as the workspace root and installed the [Eclipse Groovy REPL Scripting Environment 1.6.0](#) (update site) and [Angular-JS Eclipse Tooling](#) (update site) plugins.

### 1) Creating an Angular JS Project

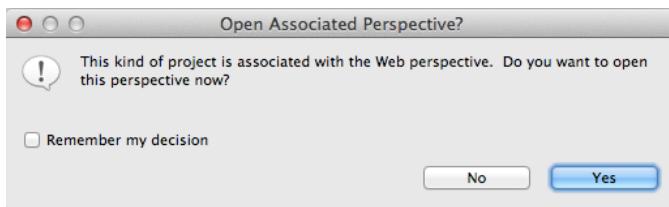
After restarting eclipse, I right-clicked on the *Project Explorer* view and chose the *New -> Static Web Project* menu item



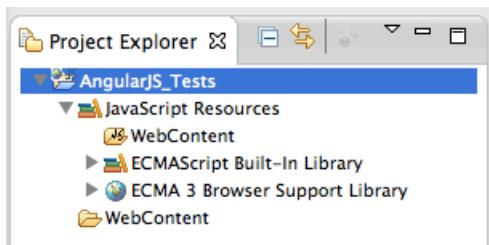
... set *AngularJS\_Tests* as the project name and clicked *Finish*



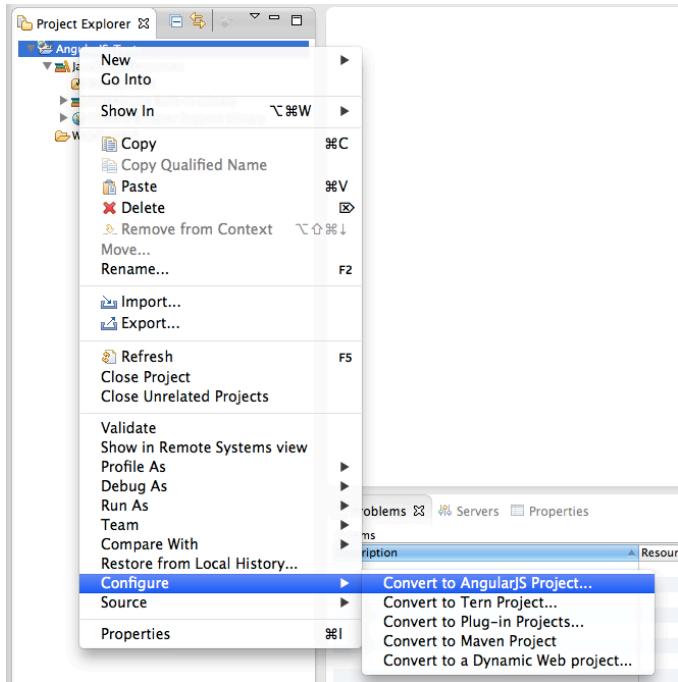
... switched to the *Web Perspective*



... with the *Project Explorer* view now looking like this:

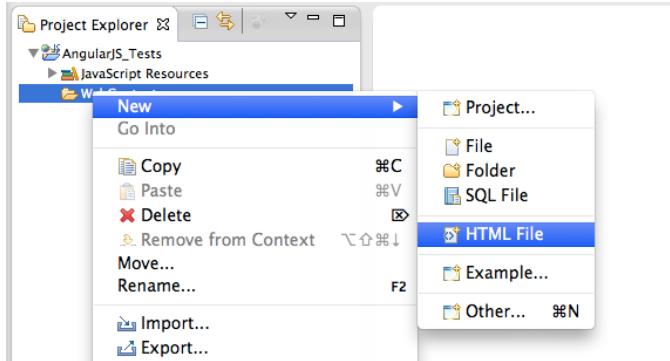


With the final setup being the conversion into an *AngularJS Project*

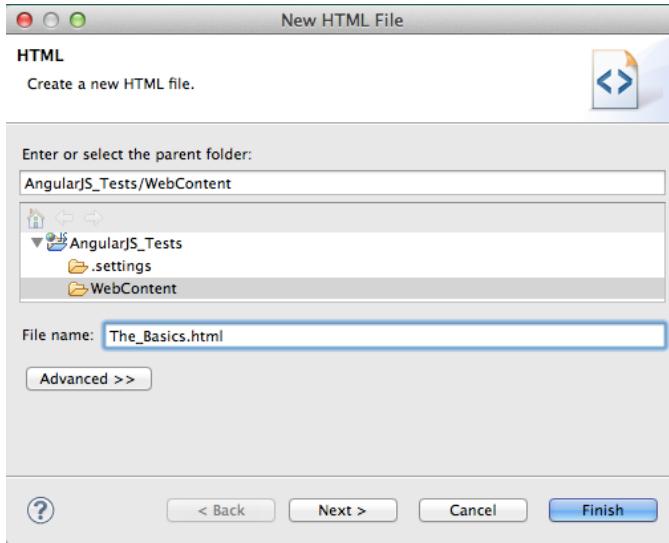


## 2) Creating the The\_Basics.html file

To create the first test file, I right-clicked on the *Web Content* folder, and chose the *New -> \_Html File* menu option:



... set the file name to *The\_Basics.html* and click on *Finish*



**\*\*NOTE:** \*\*The reason this first file is called *The\_Basics.html* is because I'm going to be using the examples from AngularJS' home page <http://angularjs.org/>

Once the \_The\_Basics.html \_file opens up in eclipse

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
</body>
</html>
```

... I change its contents to the code sample from <http://angularjs.org/>

```
<!doctype html>
<html ng-app>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13,
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name !>
<hr>
<h1>Hello {{yourName}}!</h1>
</div>
</body>
</html>
```

Note how the AngularJS Eclipse plugin successfully detects the Angular attributes and showed relevant information about it.

Here is **ng-app**:

```
<!doctype html>
<html ng-app>
<head>
<ngApp directive in module ng
<use this directive to auto-bootstrap an AngularJS application. The ngApp directive designates the root
<body element of the application and is typically placed near the root element of the page - e.g. on the or
<d tags.
<restrict A
<see http://docs.angularjs.org/api/ng.directive:ngApp
<div>Hello {{yourName}}</div>
</body>
</html>
```

Here is **ng-model**:

```
<!doctype html>
<html ng-app>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13,
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name !>
<hr>
<h1>Hello {{yourName}}</h1>
</div>
</body>
</html>
```

### 3) Fixing the undefined Attribute name issue

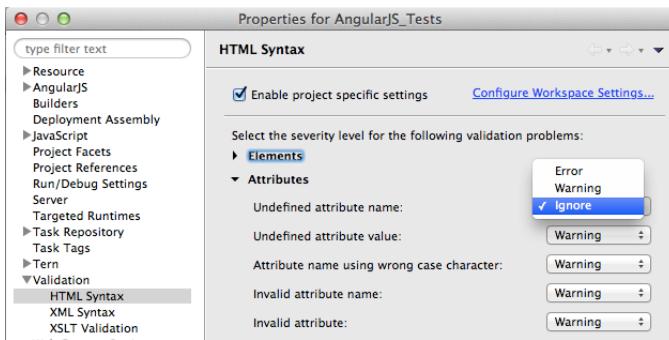
Since I chose to the Eclipse's **Static Web Project**, when we save the `_The_Basics.html` file (and if the Eclipse's Html validation settings are the default ones), the following error will show:

```
<!doctype html>
⚠ Undefined attribute name (ng-app).
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13,
</head>
<body>
<div>
<label>Name:</label>
```

... which basically means that Eclipse is not recognising the AngularJS Html attributes:

```
<!doctype html>
<html ng-app>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
</head>
<body>
<div ng-controller="HelloController">
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here">
<br>
<h1>Hello {{yourName}}!</h1>
</div>
</body>
</html>
```

To fix this, I went to the *AngularJS\_Test* project's Properties, opened the HTML Syntax page (from the Validation section) and set to false the Undefined attribute name setting (in the Attributes' options , not the Elements)



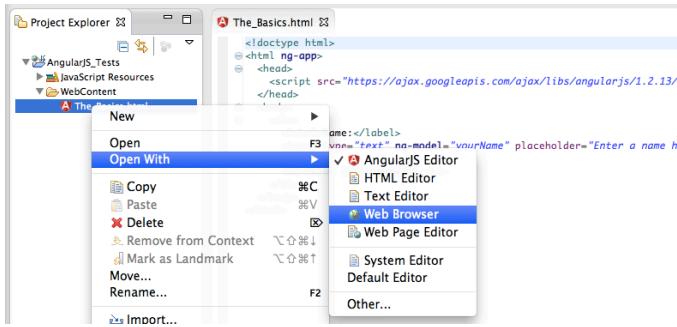
With that config change, there are no problems in this page, and hovering on top of one the AngularJS directives will show the correct tooltip:



#### \*\*4) Viewing and previewing the \*\*The\_Basics.html page

Since at the moment we only have one page, we can view it directly without needing a Web Server.

To do that, I clicked on the html file and chose the **Web Browser** option from the **Open With** menu:



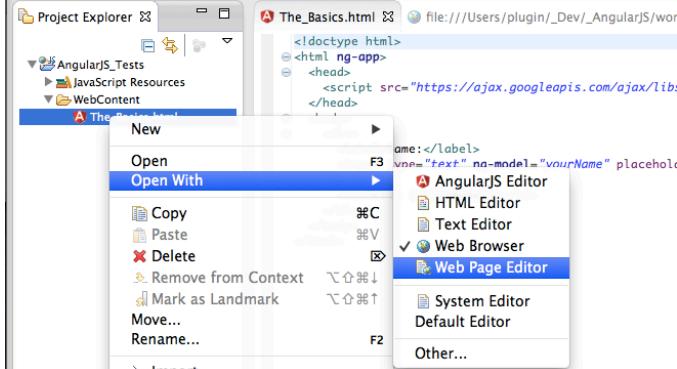
This will open the default Eclipse Browser



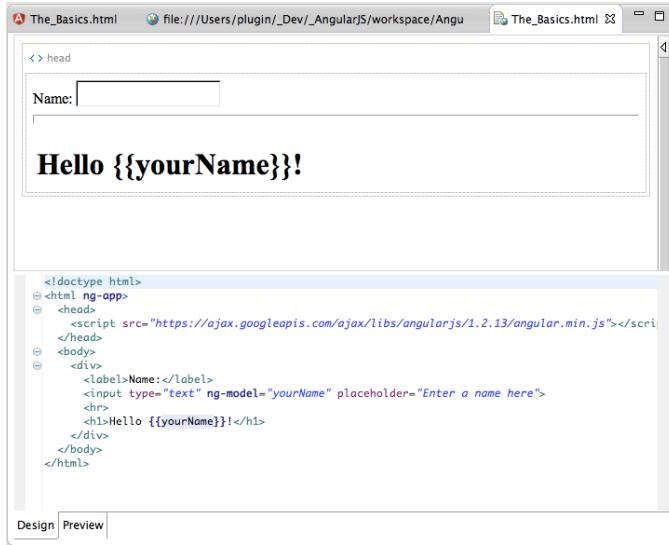
... with the AngularJS test working as expected (in this case any text typed in the **Name** TextBox will automatically be shown in the page):



We can also preview some of the changes in real time, by choosing the **Web Page Editor**:



... which will look like this (note the non-processed HTML at the top and the HTML code at the bottom):



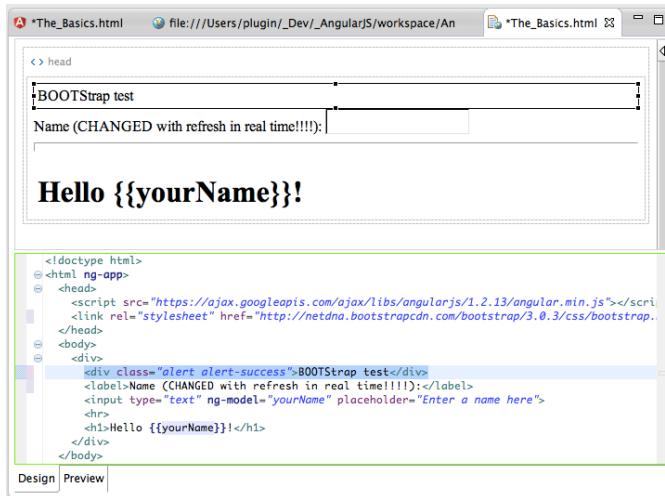
Opening up the **Preview** tab (from the default **Design** tab) will allow us to test the page currently being edited (note how Angular JS is working):



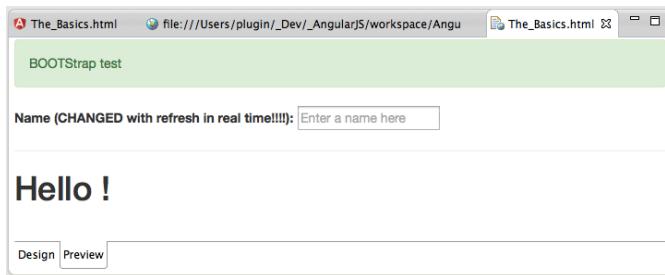
This means that (when in the **Design** tab) we can edit the AngularJS HTML page and see the changes immediately:



NOTE: This version of the Web Page Editor \*\*doesn't render the CSS while in that \*\***Design** mode, which means that if we add bootstrap to this project:



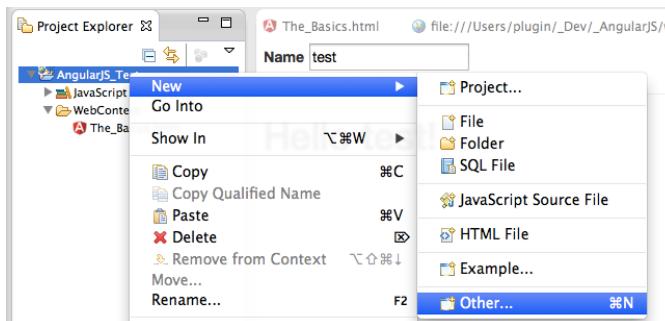
... the CSS will only be visible when in the *Preview* tab:



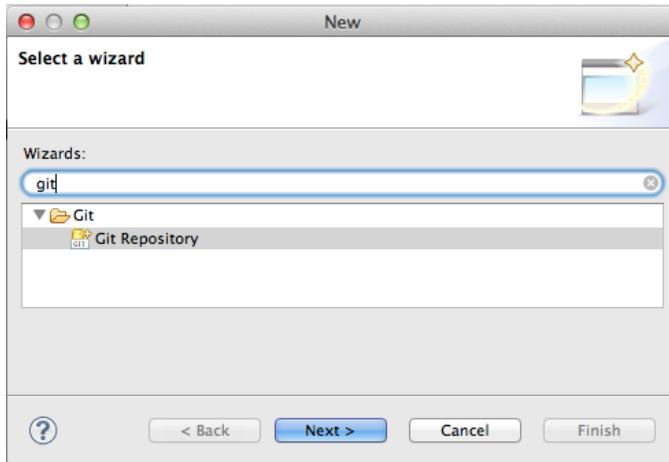
#### 4) Creating a Git Repository for the files created

The best way for me to share these files is via a Git Repository, so the final step of this post is to create one on the files we have already created.

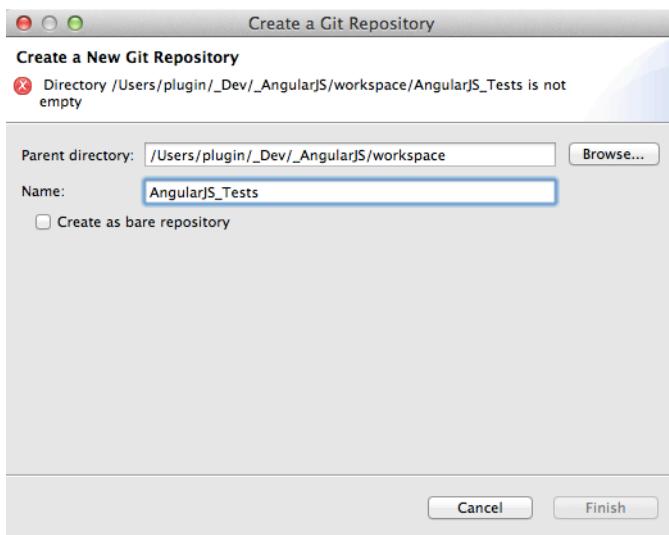
Since we are in eclipse I tried to create an Git Repo for the current project:



But the \_Git Repository \_wizard:



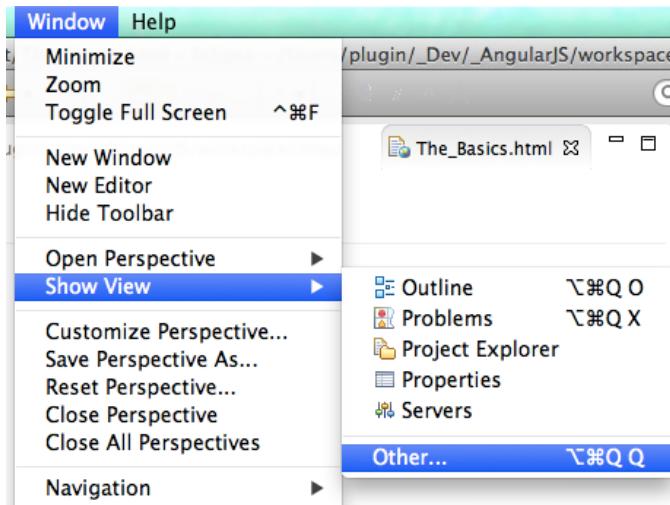
... didn't work, because it expects the target folder to not exist:



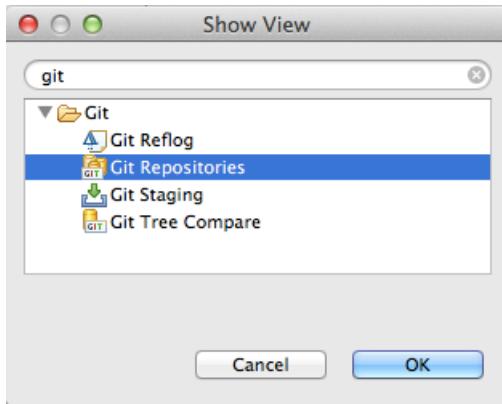
This was easily solved via the command line (by executing `$ git init` on the `_AngularJS_Tests` \_folder)

```
zen-MacBook-Air:~ plugin$ cd _Dev/
zen-MacBook-Air:_Dev plugin$ cd _AngularJS/
zen-MacBook-Air:_AngularJS plugin$ ls
eclipse      workspace
zen-MacBook-Air:_AngularJS plugin$ cd workspace/
zen-MacBook-Air:workspace plugin$ ls
AngularJS_Tests      RemoteSystemsTempFiles
zen-MacBook-Air:workspace plugin$ cd AngularJS_Tests/
zen-MacBook-Air:AngularJS_Tests plugin$ git init
Initialized empty Git repository in /Users/plugin/_Dev/_AngularJS/workspace/AngularJS_Tests/.git/
zen-MacBook-Air:AngularJS_Tests plugin$
```

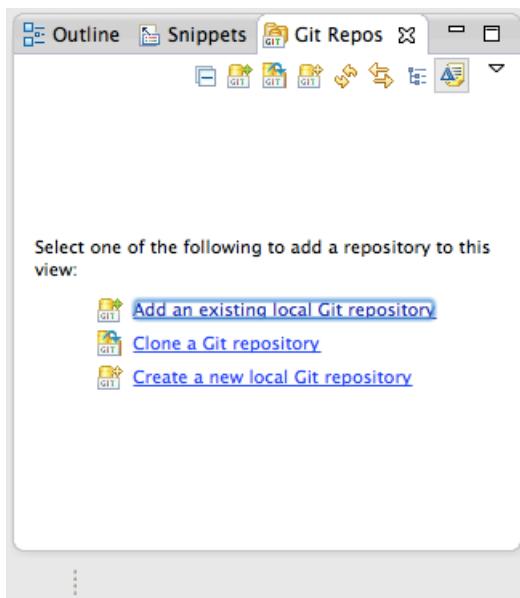
Now that we have a git repository, I was time to open it:



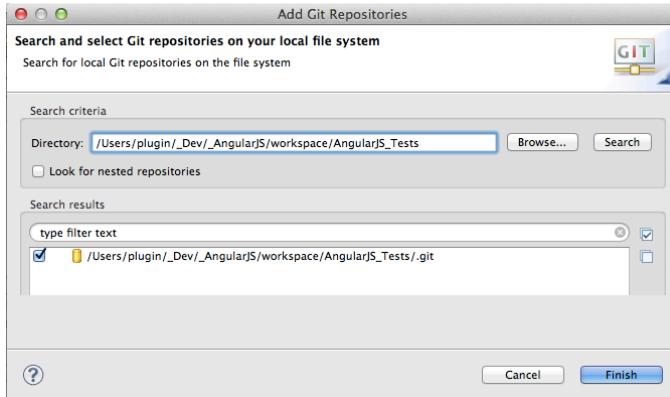
... using the \_Git Repositories\_ view



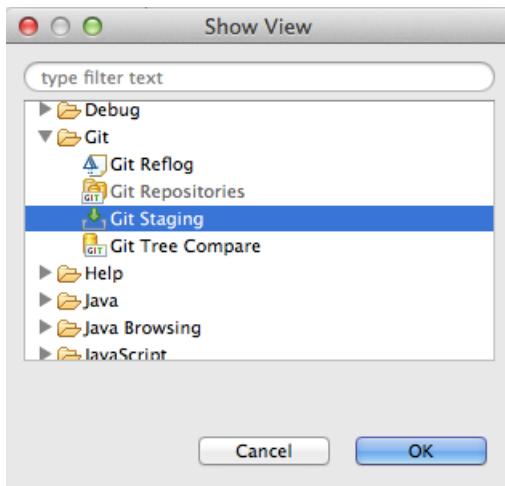
... where we can use the *Add an existing local Git repository* link:



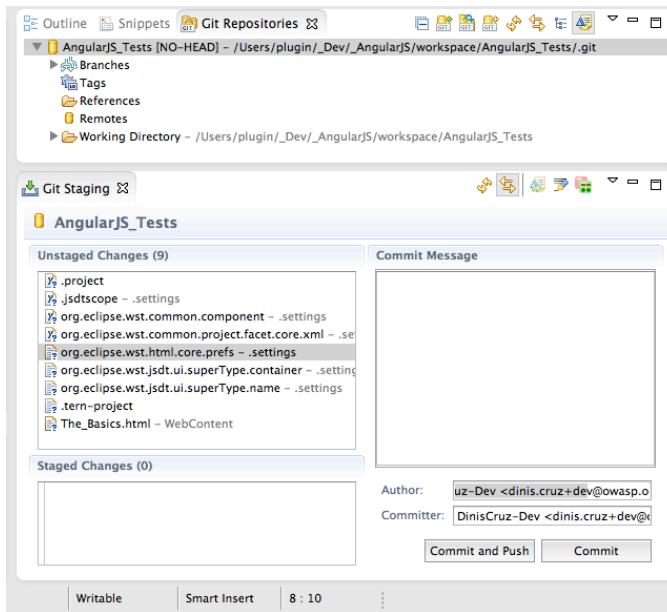
... to open the repository just created:



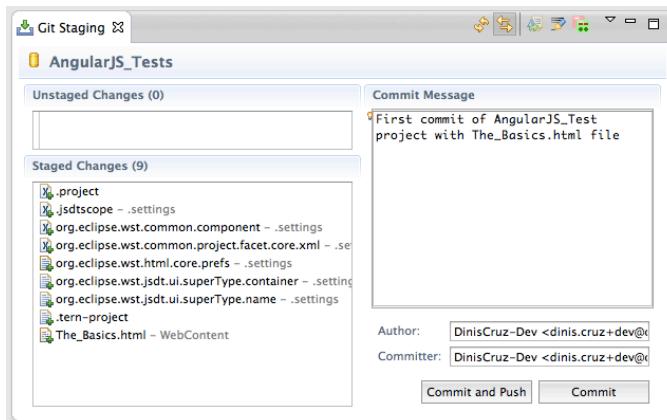
In order to create a git commit, I also opened the **Git Staging** view:



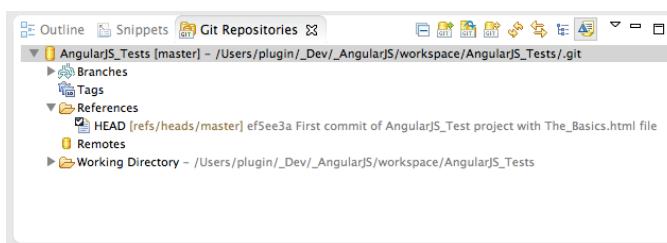
This is what these two git views look like (note that there are not commits/references and the list of new files in the \*\*Unstaged Changes \*\*list)



To commit the files drag-n-drop them from the *Unstaged Changes* to the *Staged Changes*, and write a commit message:



After clicking the **Commit** button the *\_Git Repositories\_* view will give a visual representation of the location current HEAD (which is the commit just done)



## 1.3 Using AngularJS in Eclipse, Part 2) Add Some Control

This is the second of four posts on how to run (inside Eclipse) the examples provided in [AngularJS's home page](#):

- [Using AngularJS in Eclipse, Part 1\) The Basics](#)
- [Using AngularJS in Eclipse, Part 2\) Add Some Control](#)
- [Using AngularJS in Eclipse, Part 3\) Wire up a Backend](#)
- [Using AngularJS in Eclipse, Part 4\) Create Components](#)

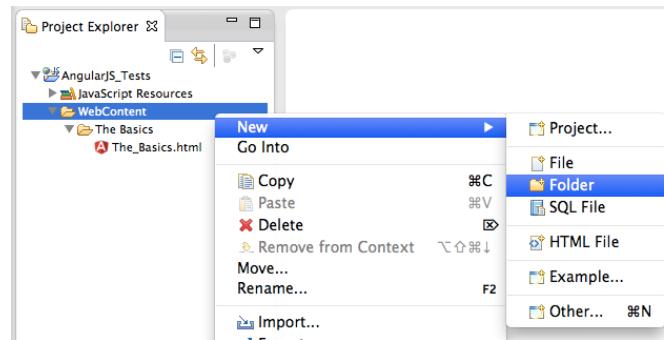
The example covered on this post is the *Add Some Control*:

The screenshot shows the AngularJS website with the 'Add Some Control' example selected. The page is divided into three main sections: 'Data Binding', 'Controller', and 'Plain JavaScript'. The 'Data Binding' section contains a brief explanation of what it is and how it works. The 'Controller' section provides a detailed explanation of controllers and their role in AngularJS. The 'Plain JavaScript' section compares AngularJS's controller behavior with other frameworks like Backbone.js. Below these sections are code snippets for 'index.html', 'todo.js', and 'todo.css'. To the right, there is a live preview of a 'Todo' application with a simple interface for adding tasks. A button labeled 'Edit Me' allows users to view or edit the code.

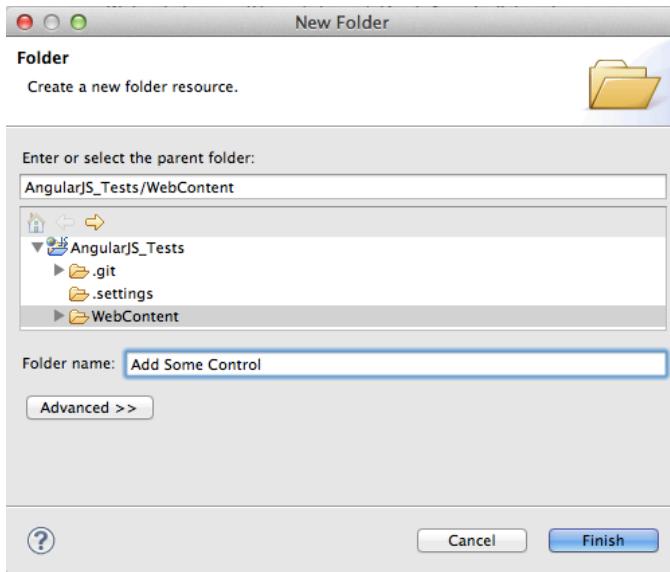
### 1) Creating the Html, CSS and JS Files

In order to keep the *AngularJS\_Tests* repository better organised, lets put each sample in its own folder.

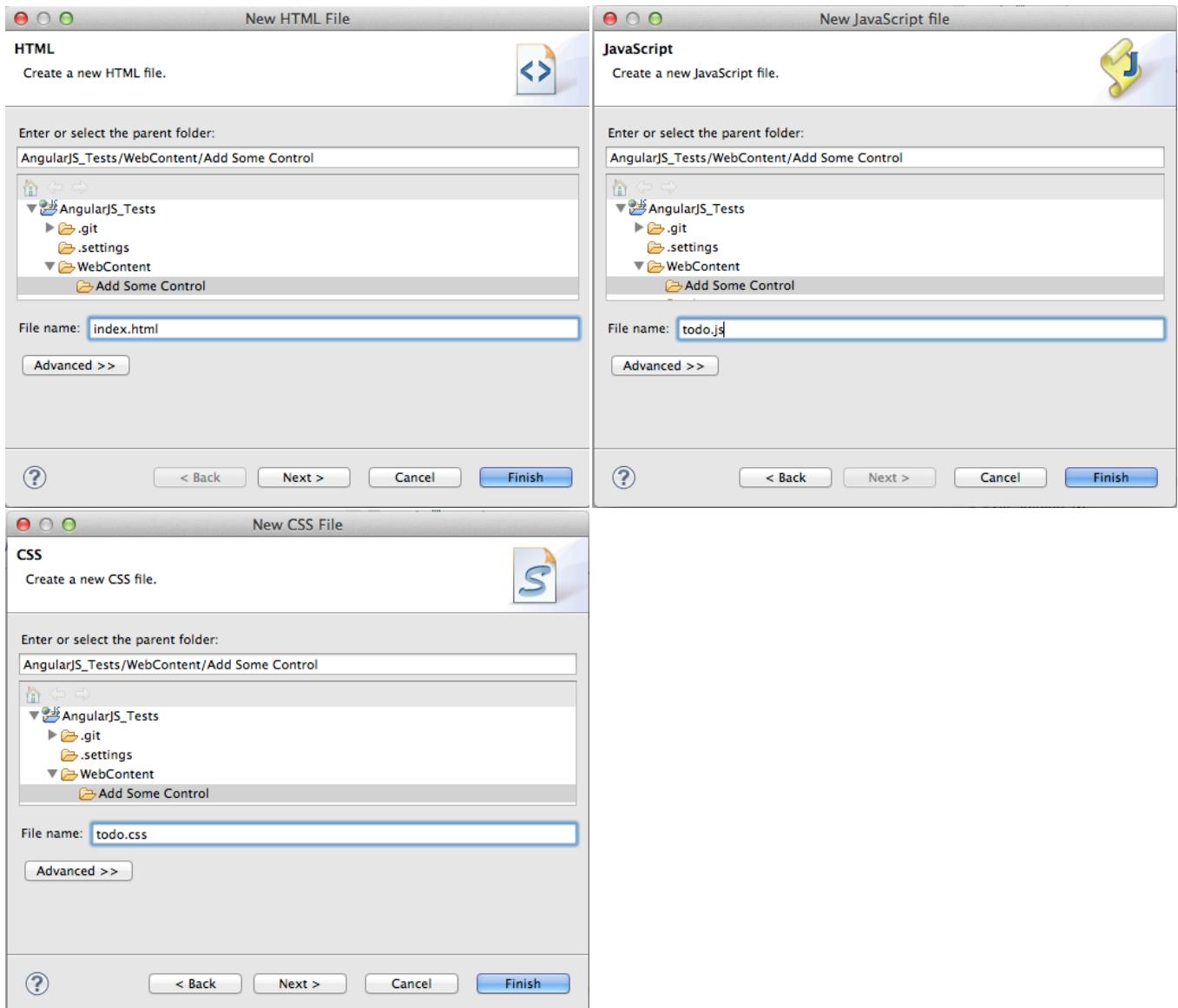
After moving the *The\_Basics.html* file into its own folder, I created a new folder:



... called *Add Some Control* to the \_WebContent \_folder of the *AngularJS\_Tests* project



And inside it, I created the `index.html`, `todo.js` and `todo.css` files (each using the Eclipse default template for the respective file type)



Here is what these 3 files look with the default content:

```

A index.html ✘
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

</body>
</html>

B todo.js ✘
/** 
 */
C todo.css ✘
@CHARSET "UTF-8";

```

Here is what they look like with the content from the [AngularJs.org \\_Add Some Control \\_sample](#):

```

A index.html ✘
<!doctype html>
<html ng-app>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
<script src="todo.js"></script>
<link rel="stylesheet" href="todo.css">
</head>
<body>
<h2>Todo</h2>
<div ng-controller="TodoCtrl">
<span>{{remaining()}} of {{todos.length}} remaining</span>
<a href="#" ng-click="archive()>archive</a> ]
<ul class="list-style">
<li ng-repeat="todo in todos">
<input type="checkbox" ng-model="todo.done">
<span class="done"{{todo.done}}>{{todo.text}}</span>
</li>
</ul>
<form ng-submit="addTodo()>addTodo()

```

```

B todo.js ✘
function TodoCtrl($scope) {
  $scope.todos = [
    {text:'learn angular', done:true},
    {text:'build an angular app', done:false};

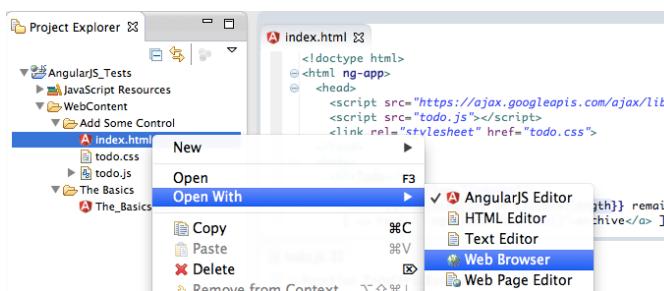
  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = '';
  };
}

C todo.css ✘
.done {
  text-decoration: line-through;
  color: grey;
}

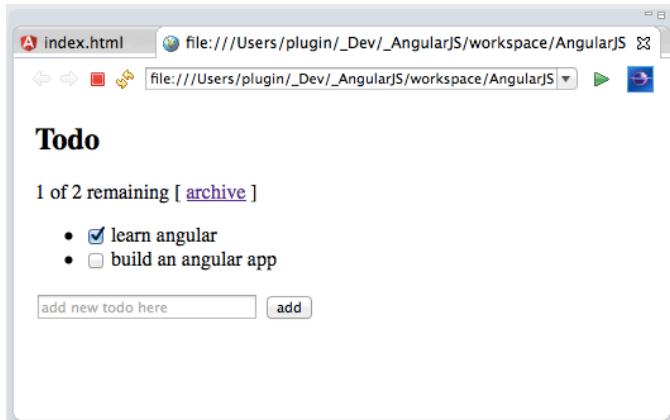
```

## 2) Opening up in Web Browser

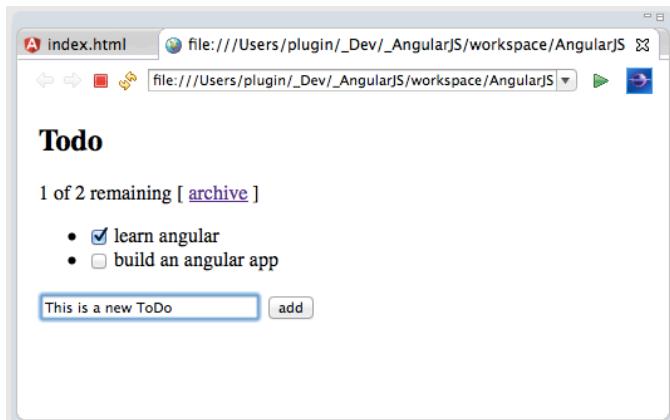
To see the example in action, right-click on the */Add Some Control/index.html* file, and chose the *Web Browser* option form the *Open With* menu



... which will look like this:



The objective of this sample is to add new \_Todos \_to the list shown, using the TextBox provided and the **add** button



So in the example shown above, after entering '*This is a new ToDo*' \_on the TextBox and clicking \_add a new item will be added to the list (see below)



We can also remove Todos by using the checkboxes (see above) and clicking on the **archive** link (see below for the result)



## 2) Making some changes to the code to see AngularJS in \$scope in action

To have a better understanding of what is going on, let's make some changes to the code sample.

Here is the original code

```
<!doctype html>


```

... which I'm going to modify by adding a new paragraph containing the value of the **todoText** variable (note that the `_todoText` \_variable is used on the `_input` \_HTML element, auto-wired to Angular by using the **ng-model** attribute).

```
<form ng-submit="addTodo()>
  <input type="text" ng-model="todoText" size="30"
         placeholder="add new todo here">
  <input class="btn-primary" type="submit" value="add">
  <p>
    New Todo text: <b>{{todoText}}</b>
  </p>
</form>
```

After refreshing the browser, we can see this in action by typing some text on the TextBox and seeing it show in real time after the "New Todo text: **text**" (one of the features of AngularJS is to keep all these variables in sync):



### 3) Changing default \*\*\*\*Todos

As another test, let's add a new default \_todo \_to the original version of the \*\*\$scope.todos \*\*array (see below).

What is happening is that the \_\$scope.todos\_ variable is populated when the \_TodoCtrl \_is executed (part of the page build), which is then inserted into the webpage using the \_<li ng-repeat="todo in todos"> \_HTML code (see screenshot of index.html above).

Here is the new **\$scope.todos** array entry added:

```

function TodoCtrl($scope) {
  $scope.todos = [
    {text:'Learn angular', done:true},
    {text:'Build an angular app', done:false},
    {text:'A NEW Angular TO DO!', done:true},
  ];

  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = '';
  };

  $scope.remaining = function() {
    var count = 0;
    angular.forEach($scope.todos, function(todo) {
      count += todo.done ? 0 : 1;
    });
    return count;
  };
}

```

... which can be seen in action by refreshing the browser (note that it is already checked by default, since I set the **done** variable to **true**)



#### 4) Changing default value of TextBox (after adding new Todo)

To show that we can control the `$scope` variables from anywhere in the `_TodoCtrl` controller, here is an example where I'm changing the `$scope.todoText` \*\*to have a specific value after the \*\*add\*\* button is clicked (which is wired to the `$scope.addTodo` using the `<form ng-submit="addTodo()">` HTML code)

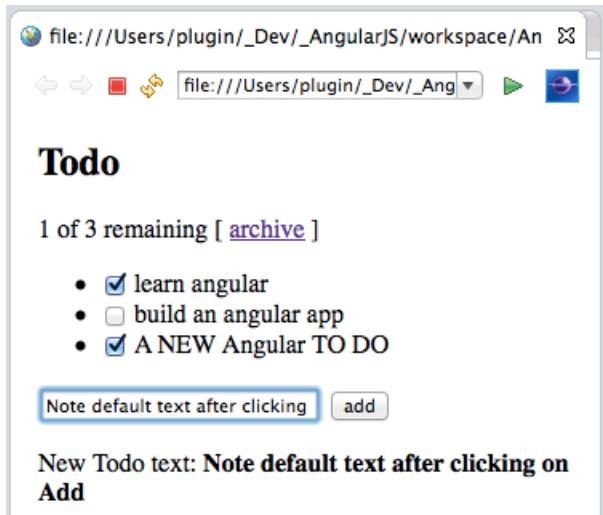
```

function TodoCtrl($scope) {
  $scope.todos = [
    {text:'learn angular', done:true},
    {text:'build an angular app', done:false},
    {text:'A NEW Angular TO DO', done:true},
  ];

  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = 'Default Text after a new Todo';
  };
}

```

After refreshing the web browser, we can see that if we add a new *todo*:



... the TextBox (and paragraph below) will have the value '\_Default value' \_(vs the original behaviour of being empty)

**Todo**

2 of 4 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A NEW Angular TO DO
- Note default text after clicking on Add

[add](#)

New Todo text: Default Text

#### 4) Creating an ‘impossible to archive’ Todo

An interesting variation is to change the default value of the `$_scope.todos` on the `$_scope.archive` method, which, instead of being empty:

```
$scope.archive = function() {
  var oldTodos = $scope.todos;
  $scope.todos = [];
  angular.forEach(oldTodos, function(todo) {
    if (!todo.done) $scope.todos.push(todo);
  });
};
```

... it contains one entry:

```
$scope.archive = function() {
  var oldTodos = $scope.todos;
  $scope.todos = [{text:'WILL be added on Archive', done:true}];
  angular.forEach(oldTodos, function(todo) {
    if (!todo.done) $scope.todos.push(todo);
  });
};
```

... which means that (after refresh) and clicking on the `archive` link

**Todo**

1 of 3 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A NEW Angular TO DO

... the \_WILL be added on Archive \_todo will now exist:

A screenshot of a browser window titled "file:///Users/plugin/\_Dev/\_AngularJS/workspace/An". The page content is a "Todo" list:

## Todo

1 of 2 remaining [[archive](#)]

- WILL be added on Archive
- build an angular app

... and even if we try to archive all:

A screenshot of a browser window titled "file:///Users/plugin/\_Dev/\_AngularJS/workspace/An". The page content is a "Todo" list:

## Todo

0 of 2 remaining [[archive](#)]

- WILL be added on Archive
- build an angular app

... the \_WILL be added on Archive \_todo will still be there:

A screenshot of a browser window titled "file:///Users/plugin/\_Dev/\_AngularJS/workspace/An". The page content is a "Todo" list:

## Todo

0 of 1 remaining [[archive](#)]

- WILL be added on Archive

## 5) Adding a Message label

As a final example, lets modify change the variable shown in the extra paragraph added to index.html to be \_message \_(which will be wired to `$scope.message`):

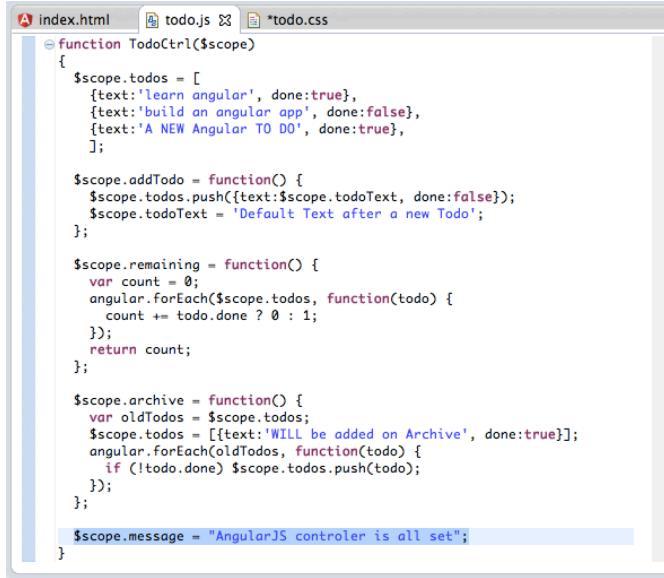


```

<!doctype html>
<html ng-app>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
  <script src="todo.js"></script>
  <link rel="stylesheet" href="todo.css">
</head>
<body>
  <h2>Todo</h2>
  <div ng-controller="TodoCtrl">
    <span>{{remaining()}} of {{todos.length}} remaining</span>
    [<a href="#" ng-click="archive()>archive</a>]
    <ul class="unstyled">
      <li ng-repeat="todo in todos">
        <input type="checkbox" ng-model="todo.done">
        <span class="done-{{todo.done}}">{{todo.text}}</span>
      </li>
    </ul>
    <form ng-submit="addTodo()">
      <input type="text" ng-model="todoText" size="30"
             placeholder="Add new todo here">
      <input class="btn-primary" type="submit" value="Add">
    <p>
      Message: <b>{{message}}</b>
    </p>
  </form>
  </div>
</body>
</html>

```

... and let's use that variable to provide visual feedback to the user when the *TodoCtrl* has executed:



```

function TodoCtrl($scope) {
  $scope.todos = [
    {text:'Learn angular', done:true},
    {text:'Build an angular app', done:false},
    {text:'A NEW Angular TO DO', done:true},
  ];

  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = 'Default Text after a new Todo';
  };

  $scope.remaining = function() {
    var count = 0;
    angular.forEach($scope.todos, function(todo) {
      count += todo.done ? 0 : 1;
    });
    return count;
  };

  $scope.archive = function() {
    var oldTodos = $scope.todos;
    $scope.todos = [{text:'WILL be added on Archive', done:true}];
    angular.forEach(oldTodos, function(todo) {
      if (!todo.done) $scope.todos.push(todo);
    });
  };
}

$scope.message = "AngularJS controller is all set";
}

```

Now, we should see the message \_‘AngularJS controller is all set’ \_just after the page finishes reloading:

**Todo**

1 of 3 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A NEW Angular TO DO

Message: **AngularJS controller is all set**

To make it more relevant, let's add a `$_scope.message` to the `$_scope.addTodo` method:

```
function TodoCtrl($scope) {
  $scope.todos = [
    {text:'learn angular', done:true},
    {text:'build an angular app', done:false},
    {text:'A NEW Angular TO DO', done:true},
  ];

  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = 'Default Text after a new Todo';
    $scope.message = "$scope.addTodo was called";
  };
}
```

... and the `$_scope.archive` method:

```
$scope.archive = function() {
  var oldTodos = $scope.todos;
  $scope.todos = [{text:'WILL be added on Archive', done:true}];
  angular.forEach(oldTodos, function(todo) {
    if (!todo.done) $scope.todos.push(todo);
  });
  $scope.message = "$scope.archive was called";
};
```

Now we will get this message when the page reloads:

**Todo**

1 of 3 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A NEW Angular TO DO

Message: **AngularJS controller is all set**

... this message after clicking on the ***add*** button

file:///Users/plugin/\_Dev/\_AngularJS/workspace/An

**Todo**

2 of 4 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A NEW Angular TO DO
- Note Message Change

Default Text after a new Todo

Message: **\$scope.addTodo was called**

... this message after clicking on the ***archive link***

file:///Users/plugin/\_Dev/\_AngularJS/workspace/An

**Todo**

2 of 3 remaining [ [archive](#) ]

- WILL be added on Archive
- build an angular app
- Note Message Change

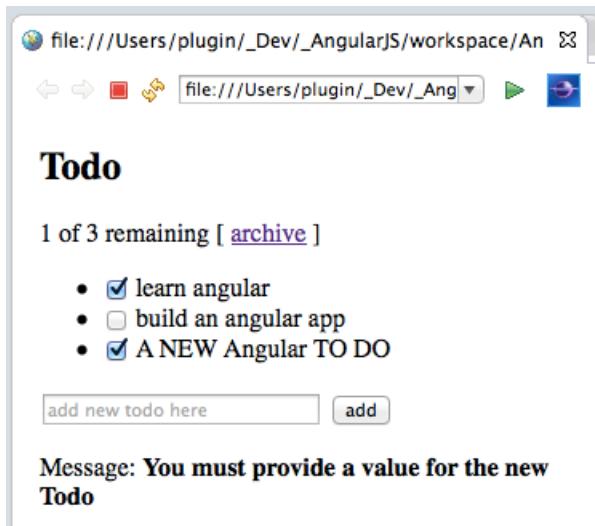
Default Text after a new Todo

Message: **\$scope.archive was called**

Finally let's refactor the `$scope.addTodo` method to so that it doesn't add a new ***Todo*** when no value is provided:

```
$scope.addTodo = function()
{
  if(!$scope.todoText)
  {
    $scope.message = "You must provide a value for the new Todo";
  }
  else
  {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.message = "New Todo added:" + $scope.todoText;
    $scope.todoText = '';
  }
};
```

After reloading the page, we can confirm that trying to click on the ***add*** button without entering first any text on the TextBox will show this message:



... and if we add a new one (in this case '*not an empty todo*') we will see this message:



#### Appendix: CSS not being rendered in Eclipse's browser.

One thing I noticed was that the style changes provided by `todo.css` :



... was not being shown inside eclipse (as you can see on multiple screenshots above), but worked ok in Chrome:

The screenshot shows a web browser window with the title bar "index.html". The address bar shows the URL "file:///Users/plugin/\_Dev/\_AngularJS/workspace/AngularJS\_Tests/WebContent/Add%20Some%20Control/index.html". The main content area displays a "Todo" list:

**Todo**

1 of 3 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A-NEW-Angular-TO-DO

Message: AngularJS controller is all set

... and stand-alone Safari (note that there is a *line-through* applied when an item is checked as done)

The screenshot shows a web browser window with the title bar "index.html". The address bar shows the URL "file:///Users/plugin/\_Dev/\_AngularJS/workspace/AngularJS\_Tests/WebContent/Add%20Some%20Control/index.html". The main content area displays a "Todo" list:

**Todo**

1 of 4 remaining [ [archive](#) ]

- learn angular
- build an angular app
- A-NEW-Angular-TO-DO
- Another added

Message: New Todo added:Another added

---

[Table of Contents](#) | [Code](#)

## 1.4 Using AngularJS in Eclipse, Part 3) Wire up a Backend

This is the third of four posts on how to run (inside Eclipse) the examples provided in [AngularJS's home page](#):

- [Using AngularJS in Eclipse, Part 1\) The Basics](#)
- [Using AngularJS in Eclipse, Part 2\) Add Some Control](#)
- Using AngularJS in Eclipse, Part 3) Wire up a Backend
- [Using AngularJS in Eclipse, Part 4\) Create Components](#)

The example covered on this post is the *Wire up a Backend*:

**Wire up a Backend**

**Deep Linking**  
A deep link reflects where the user is in the app, this is useful so users can bookmark and email links to locations within apps. Round trip apps get this automatically, but AJAX apps by their nature do not. AngularJS combines the benefits of deep linking with desktop app-like behavior.

**Form Validation**  
Client-side form validation is an important part of great user experience. AngularJS lets you declare the validation rules of the form without having to write JavaScript code. Write less code, go have beer sooner.

**Server Communication**  
AngularJS provides built-in services on top of XHR as well as various other backends using third party libraries. Promises further simplify your code by handling asynchronous return of data. In this example, we use the AngularFire library to wire up a Firebase backend to a simple Angular app.

```

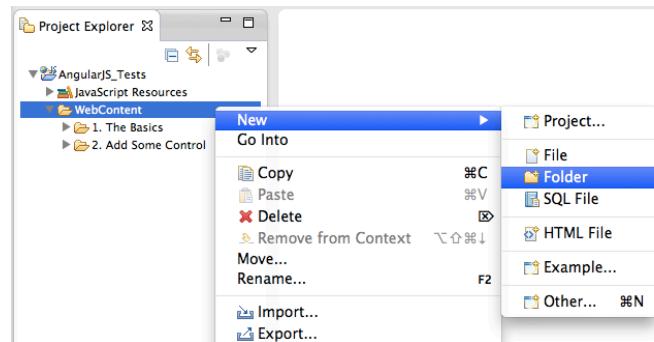
index.html project.js list.html detail.html

1. <!doctype html>
2. <html ng-app="project">
3.   <head>
4.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/
angular-min.js"></script>
5.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/
angular-resource.min.js">
6.     </script>
7.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/
angularfire-min.js">
8.     </script>
9.     <script src="https://cdn.firebaseio.com/0.firebaseio.js"></script>
10.    <script src="https://cdn.firebaseio.com/lib/angularfire/0.5.0/
angularfire-min.js"></script>
11.    <script src="project.js"></script>
12.  </head>
13.  <body>
14.    <h2>JavaScript Projects</h2>
15.    <div ng-view></div>
16.  </body>
17. </html>

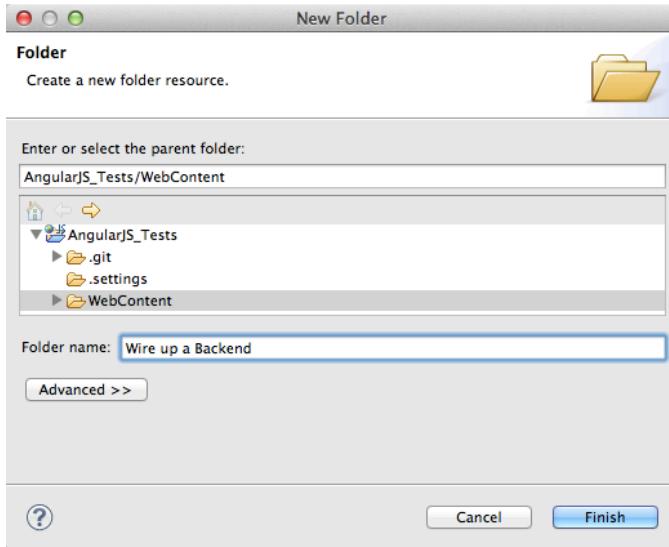
```

### 1) Creating the test files

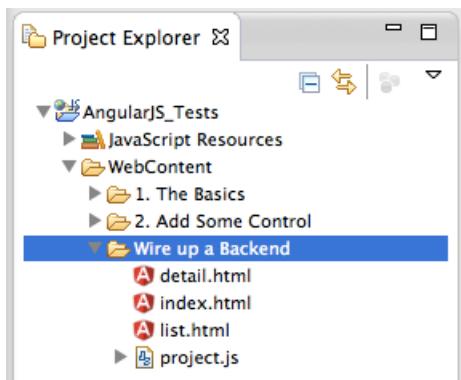
First step is to create a folder:



... called *Wire a Backend*



... to hold the 4 files required for this example: *index.html, detail.html, list.html and project.js*

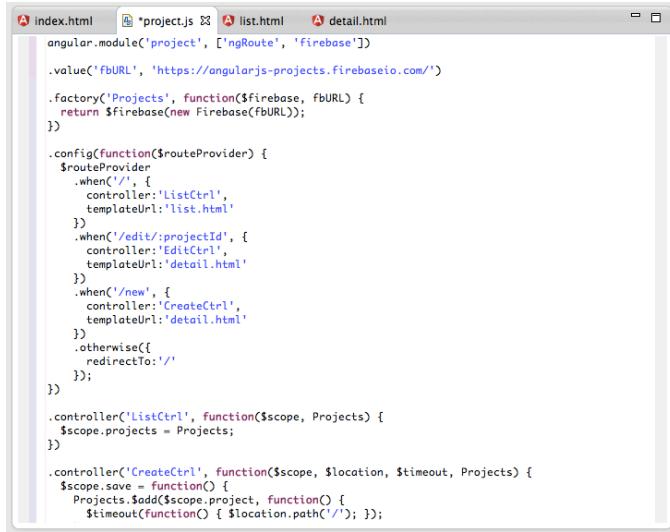


Here is what they look like (with content from <http://angularjs.org>)

**index.html :**

```
<!DOCTYPE html>
<html ng-app="project">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-resource.min.j
    </script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-route.min.js">
    </script>
    <script src="https://cdn.firebaseio.com/v0.firebaseio.js"></script>
    <script src="https://cdn.firebaseio.com/lib/angularfire/0.5.0/angularfire.min.js"></script>
    <script src="project.js"></script>
  </head>
  <body>
    <h2>JavaScript Projects</h2>
    <div ng-view></div>
  </body>
</html>
```

**project.js :**



```

angular.module('project', ['ngRoute', 'firebase'])

.value('fbURL', 'https://angularjs-projects.firebaseio.com/')

.factory('Projects', function($firebase, fbURL) {
  return $firebase(new Firebase(fbURL));
})

.config(function($routeProvider) {
  $routeProvider
    .when('/', {
      controller:'ListCtrl',
      templateUrl:'list.html'
    })
    .when('/edit/:projectId', {
      controller:'EditCtrl',
      templateUrl:'detail.html'
    })
    .when('/new', {
      controller:'CreateCtrl',
      templateUrl:'detail.html'
    })
    .otherwise({
      redirectTo:'/'
    });
})

.controller('ListCtrl', function($scope, Projects) {
  $scope.projects = Projects;
})

.controller('CreateCtrl', function($scope, $location, $timeout, Projects) {
  $scope.save = function() {
    Projects.$add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  }
})

```

list.html :



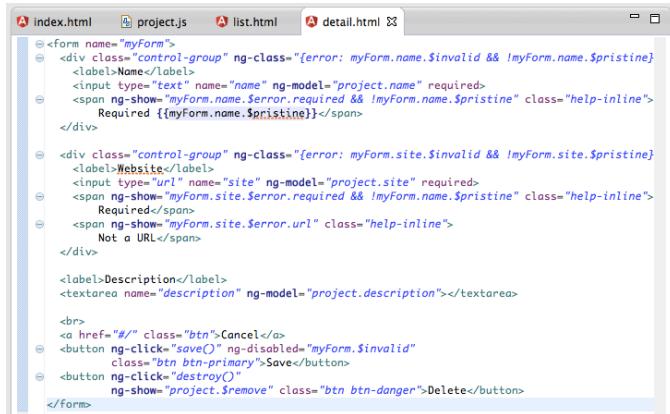
```

<input type="text" ng-model="search" class="search-query" placeholder="Search">


| Project                          | Description             | <a href="#/new">&lt;i class="icon-plus-sign"&gt;&lt;/i&gt;</a>                |
|----------------------------------|-------------------------|-------------------------------------------------------------------------------|
| <a href="#">{{project.site}}</a> | {{project.description}} | <a href="#/edit/{{project.\$id}}">&lt;i class="icon-pencil"&gt;&lt;/i&gt;</a> |


```

detail.html :



```

<form name="myForm">
  <div class="control-group" ng-class="{error: myForm.name.$invalid && !myForm.name.$pristine}>
    <label>Name</label>
    <input type="text" name="name" ng-model="project.name" required>
    <span ng-show="myForm.name.$error.required && !myForm.name.$pristine" class="help-inline">
      Required {{myForm.name.$pristine}}
    </span>
  </div>

  <div class="control-group" ng-class="{error: myForm.site.$invalid && !myForm.site.$pristine}>
    <label>Website</label>
    <input type="url" name="site" ng-model="project.site" required>
    <span ng-show="myForm.site.$error.required && !myForm.name.$pristine" class="help-inline">
      Required
    </span>
    <span ng-show="myForm.site.$error.url" class="help-inline">
      Not a URL
    </span>
  </div>

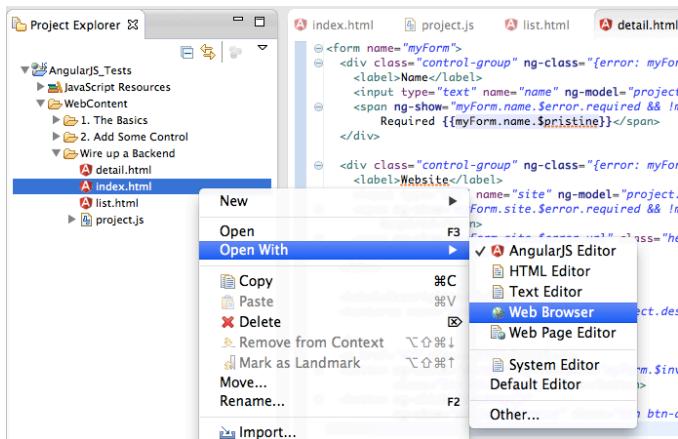
  <label>Description</label>
  <textarea name="description" ng-model="project.description"></textarea>

  <br>
  <a href="#" class="btn"><Cancel/></a>
  <button ng-click="save()" ng-disabled="myForm.$invalid" class="btn btn-primary">Save</button>
  <button ng-click="destroy()" ng-show="project.$remove" class="btn btn-danger">Delete</button>
</form>

```

## 2) Running in Browser and noticing CSS problems

To see this example in action, lets open it in Eclipse's WebBrowser:



... which looks like this:

Project	Description
<a href="#">AngularJS hh</a>	HTML enhanced for web apps!
<a href="#">Backbone</a>	Models for your apps.
<a href="#">Batman</a>	Quick and beautiful.
<a href="#">Cappuccino</a>	Objective-J.
<a href="#">Ember</a>	Ambitious web apps.
<a href="#">GWT</a>	JS in Java.
<a href="#">jQuery</a>	Write less, do more.
<a href="#">Knockout</a>	MVVM pattern.
<a href="#">Sammy</a>	Small with class.
<a href="#">Spine</a>	Awesome MVC Apps.
<a href="#">SproutCore</a>	Innovative web-apps.

... with a search bar at the top that can be used to filter the loaded data:

Project	Description
<a href="#">AngularJS hh</a>	HTML enhanced for web apps!
<a href="#">Batman</a>	Quick and beautiful.

Only problem is that it doesn't look at all like it does on the <http://angularjs.org> page (see below)

The screenshot shows the AngularJS project editor interface. On the left, there is a code editor window containing the following HTML code:

```

1. <!doctype html>
2. <html ng-app="project">
3.   <head>
4.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/
angular.min.js"></script>
5.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/
angular-route.min.js">
6.       </script>
7.       <script src="https://cdn.firebaseio.com/v0/firebase.js"></script>
8.       <script src="https://cdn.firebaseio.com/libs/angularfire/0.5.0/
angularfire.min.js"></script>
9.       <script src="project.js"></script>
10.   </head>
11.   <body>
12.     <h2>JavaScript Projects</h2>
13.     <div ng-view></div>
14.   </body>
15. </html>

```

On the right, there is a sidebar titled "JavaScript Projects" with a search bar and a list of projects:

- Project** AngularJS hh **Description** HTML enhanced for web apps!
- Backbone** Backbone **Description** Models for your apps.
- Batman** Batman **Description** Quick and beautiful.
- Cappuccino** Cappuccino **Description** Objective-J.
- Ember** Ember **Description** Amazing web apps.
- GWT** GWT **Description** JS in Java.
- JQuery** JQuery **Description** Write less, do more.
- Knockout** Knockout **Description** MVVM pattern.
- Sammy** Sammy **Description** Small with class.
- Spine** Spine **Description** Awesome MVC Apps.
- SproutCore** SproutCore **Description** Innovative web-apps.
- test** test **Description** test<h1>xss</h1> test<h1>xss</h1>

Note how not only the css styles are different, the *add* and *edit* links (i.e. icons) are missing:

The screenshot shows a simplified list view of JavaScript projects. The title is "JavaScript Projects". Below it is a search bar. A table lists the projects:

Project	Description	Add
AngularJS hh	HTML enhanced for web apps!	
Backbone	Models for your apps.	
Batman	Quick and beautiful.	
Cappuccino	Objective-J.	

Here is what the *edit* page should look like:

The screenshot shows the "Edit Project" form for the "Cappuccino" project. The title is "JavaScript Projects". The form fields are:

- Name: Cappuccino
- Website: <http://cappuccino.org/>
- Description: Objective-J.

At the bottom are three buttons: "Cancel", "Save" (blue), and "Delete" (red).

Here is what the \_new \_page should look like:

## JavaScript Projects

Name

Website

Description

### 3) Running example on JSFiddle

On the <http://angularjs.org> page there is a button called *Edit Me* (top right) which when clicked will do a cross-site POST submission to <http://jsfiddle.net/api/post/library/pure/>

Hint: hover over me .

## JavaScript Projects

Search

Project	Description
AngularJS	HTML enhanced for web apps!
Backbone	Models for your apps.
Batman	Quick and beautiful.
Cappucino	Objective-J.
Ember	Ambitious web apps.
GWT	JS in Java.
jQuery	Write less, do more.
Knockout	MVVM pattern.
Sammy	Small with class.
Spine	Awesome MVC Apps.
SproutCore	Innovative web-apps.

... which looks like this (notice the extra [Bootstrap](#) css included that is not present on the code sample provided in the <http://angularjs.org> page )

For reference here is what the *Edit Me* form looks like in the <http://angularjs.org> page (with the values passed as hidden parameters to *jsfiddle*)

```
<!-- Cache FILE: list.html -->
<script type="text/ng-template" id="list.html">
<input type="hidden" name="app" value="Angular Example: ">
<input type="hidden" name="css" value="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css" style="display:none;">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-resource.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-route.min.js"></script>
<script src="https://www.gstatic.com/firebasejs/3.1.1/firebase.js"></script>
<script src="https://cdn.firebaseio.com/lib/angularfire/0.5.0/angularfire.min.js"></script>
<!-- CACHE FILE: detail.html -->
<script type="text/ng-template" id="detail.html">
<form ng-submit="ctrl.create()" style="margin-bottom: 10px;">
  <div class="control-group" ng-class="error: ctrl.errorName || ctrl.errorEmail || ctrl.errorPhone || ctrl.errorAddress">
    <input type="text" ng-model="ctrl.name" placeholder="Name" style="width: 100%; height: 30px;">
  </div>
  <div class="control-group" ng-class="error: ctrl.errorEmail || ctrl.errorPhone || ctrl.errorAddress">
    <input type="text" ng-model="ctrl.email" placeholder="Email" style="width: 100%; height: 30px;">
  </div>
  <div class="control-group" ng-class="error: ctrl.errorPhone || ctrl.errorAddress">
    <input type="text" ng-model="ctrl.phone" placeholder="Phone" style="width: 100%; height: 30px;">
  </div>
  <div class="control-group" ng-class="error: ctrl.errorAddress">
    <input type="text" ng-model="ctrl.address" placeholder="Address" style="width: 100%; height: 30px;">
  </div>
  <div style="text-align: right; margin-top: 10px;">
    <button type="submit" class="btn btn-primary" style="width: 100px; height: 30px;">Create
  </div>
</form>
<table border="1" style="width: 100%; border-collapse: collapse; font-size: 10pt; border: none; text-align: center; margin-top: 10px;">
  <thead>
    <tr style="background-color: #ccc; border: none; font-weight: bold;">
      <th style="padding: 5px;">#Project NameDescriptionActions{{project.$index+1}}{{project.title}}{{project.description}}
```

And here is the jsfiddle API description for the method used (<http://doc.jsfiddle.net/api/post.html>)

Display a Fiddle from POST

- Input
  - URL Structure
  - POST Variables
- Example

There is a way to display a fiddle using the POST query.

## Input

### URL Structure ¶

One need to choose the framework using the URL:

URL: `http://jsfiddle.net/api/post/{framework}/{version}/`

Use `library/pure` for no framework:

URL: `http://jsfiddle.net/api/post/library/pure/`

There is an option to add dependencies as a comma separated list:

URL: `http://jsfiddle.net/api/post/{framework}/{version}/dependencies/{dependency_list}/`

**framework**  
the desired framework name. Which framework should be loaded with the fiddle.

**version**  
substring of the framework version - the last passing will be used. If 1.3 will be given, jsFiddle will use the 1.3.1 and 1.3

**dependency\_list**  
comma separated list of dependency substrings. It would mark any dependency containing the substring.

#### 4) Fixing CSS issue

Since I wanted to match the CSS of my local test page to the style used in the example embedded in the <http://angularjs.org> page, the best place to look is on the source code of that page :)

Using Chrome Browser Inspector, on the \_Styles \_tab, I was able to see that they were using two css libraries: **Bootstrap** and **Font Awesome**:

The screenshot shows the Chrome Developer Tools' Elements panel with the 'Styles' tab selected. An element with the class 'icon-pencil' is highlighted. The styles listed include:

```

:ng-valid" placeholder="Search">
derBy:'name' -->
derBy:'name' class="ng-scope">...</tr>
| orderBy:'name' -->
orderBy:'name'" class="ng-scope">

```

element.style {

```

[bootstrap.min.css:248]
child, [class*= icon-]:last-child {
}
```

a [class^="icon-"], a [class\*=" icon-"] {
display: inline-block;
text-decoration: inherit;
}

[font-awesome.css:55]
[font-awesome.css:33]

body tr.ng-scope td a.icon-pencil

Back in Eclipse, I added the bootstrap css reference

The screenshot shows the Eclipse IDE's code editor with the file 'index.html' open. The code includes a link to 'bootstrap.min.css'.

```

<!DOCTYPE html>
<html ng-app="project">
<head>
<link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css"/>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>

```

And on refresh, the page looked better (but the **edit** and **add** icons/links where still missing)

The screenshot shows the Eclipse IDE's code editor with the file 'index.html' open. The page displays a list of projects with Bootstrap CSS styling applied.

Project	Description
ah	akkkkk
AngularJS	HTML enhanced for web apps!
Backbone	Models for your apps.
Batman	Quick and beautiful.
Cappuccino	Objective-J.
Ember	Ambitious web apps.
GWT	JS in Java.
jQuery	Write less, do more.
Knockout	MVVM pattern.
Sammy	Small with class.
Spine	Awesome MVC Apps.
SproutCore	Innovative web-apps.

So I went back to the angularjs.org source code to see what they were using:

The screenshot shows the browser developer tools' Source tab with the URL 'view-source:angularjs.org'. The code includes:

```

<!DOCTYPE html>
<html class="no-js lt-ie9 lt-ie8 lt-ie7"> <html lang="en"> <!-->
<!-- If IE 7--> <html class="no-js lt-ie9 lt-ie8 lang=en"> <!-->
<!-- If IE 8--> <html class="no-js lt-ie9 lang=en"> <!-->
<!-- If gt IE 8--><!--> <html class="no-js" lang=en> <!-->
<!-->
<meta charset="utf-8"
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"
<meta name="Description"
<meta name="AngularJS" content="AngularJS is what HTML would have been, had it been designed for building web-apps.
  Declarative templates, dependency injection, MVW, MVVN, MVC, dependency injection and great
  utility library all implemented with pure client-side JavaScript!">
<title>AngularJS - Superheroic JavaScript MW Framework</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
<link rel="stylesheet" href="bootstrap.min.css">
<link rel="stylesheet" href="font-awesome.css">
<link href="google-code-prettify/prettify.css" type="text/css" rel="stylesheet">
<base href="/">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-animate.js"></script>

```

First I tried adding the latest **Font Awesome** from their CDN:

The screenshot shows the Eclipse IDE's code editor with the file 'index.html' open. A new link to 'font-awesome.css' has been added to the head section.

```

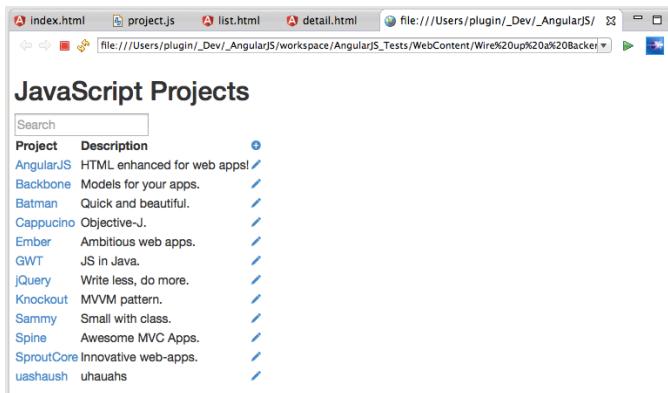
<!DOCTYPE html>
<html ng-app="project">
<head>
<link href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css" rel="stylesheet"
<link href="http://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" rel="stylesheet"
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>

```

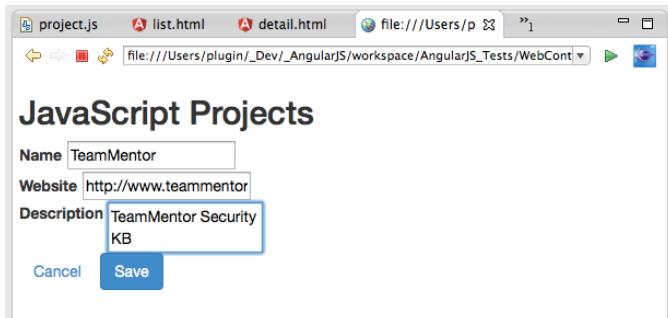
But since that didn't work, I decided to add both `font-awesome.css` \*\*and `docs.css`\*\* references:

```
<!doctype html>
<html ng-app="project">
<head>
  <link href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css" rel="stylesheet"/>
  <link href="http://angularjs.org/css/font-awesome.css" rel="stylesheet"/>
  <link href="http://angularjs.org/css/docs.css" rel="stylesheet">
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
```

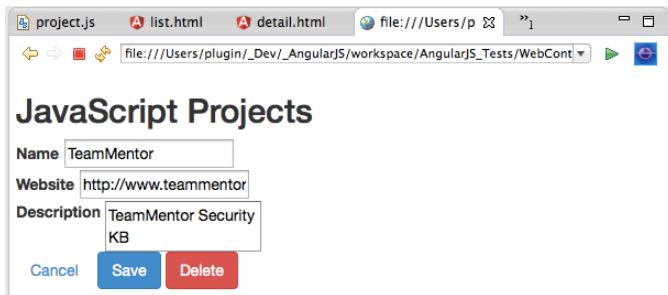
... with much better results (note the icons on the right)



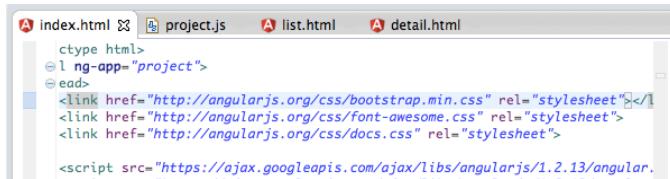
But there was still something wrong, since the *new* page looked like this:



... and the *edit* page looked like this:



Since it was all good on the angular.org site, my next try was to use the `bootstrap.min.css` file from it (vs the CDN)

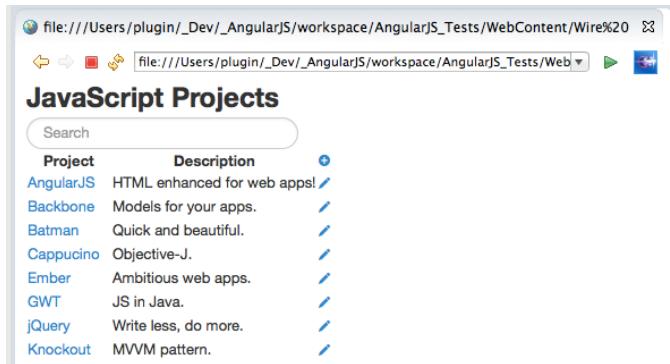


```

  ctype html>
  @l ng-app="project"
  @ead>
  <link href="http://angularjs.org/css/bootstrap.min.css" rel="stylesheet">
  <link href="http://angularjs.org/css/font-awesome.css" rel="stylesheet">
  <link href="http://angularjs.org/css/docs.css" rel="stylesheet">
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.

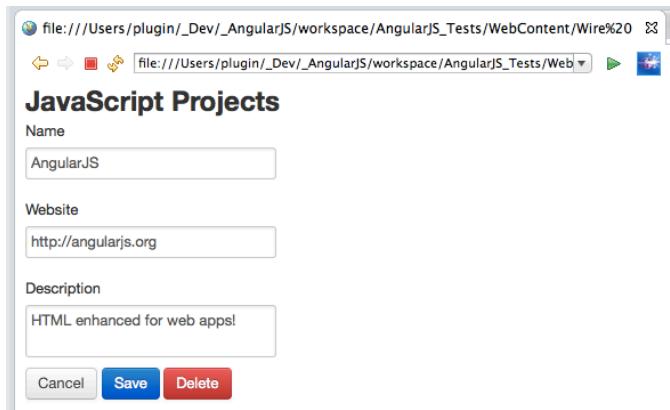
```

With the local page:



Project	Description
AngularJS	HTML enhanced for web apps!
Backbone	Models for your apps.
Batman	Quick and beautiful.
Cappuccino	Objective-J.
Ember	Ambitious web apps.
GWT	JS in Java.
jQuery	Write less, do more.
Knockout	MVVM pattern.

... and the *edit* page now looking just like the real thing (note the Search TextBox (above) and the fields (below))



Name	<input type="text" value="AngularJS"/>
Website	<input type="text" value="http://angularjs.org"/>
Description	<input type="text" value="HTML enhanced for web apps!"/>
	<input type="button" value="Cancel"/> <input type="button" value="Save"/> <input type="button" value="Delete"/>

That said, the layout where still a little bit different (namely the background of the local file which was white).

So I had a look at the style of the example in angular.org and noticed that there was a \_div \_element with the bootstrap classes **well** \*\*and\*\* **ng-scope**\*\*:

The screenshot shows the browser's developer tools with the 'Elements' tab selected. A specific div element is highlighted with a green background, which corresponds to the 'well ng-scope' class defined in the CSS. The element's content is visible, including an h2 header and some nested divs. The right-hand panel displays the 'Styles' tab, showing the CSS rules applied to the selected element, such as 'min-height: 20px;' and 'background-color: #f5f5f5;'. Below the styles, there is a list of other JavaScript projects.

```

<div well="" ng-scope="">
  <h2>JavaScript Projects</h2>
  ...
</div>

```

Back in the local file, I edited the index.html to also have a similar div:

This screenshot shows the local development environment. The 'index.html' file is open in the code editor. A div element with the class 'well ng-scope' is selected and highlighted with a blue background. The element's content includes an h2 header and a nested div with 'ng-view'. The right-hand panel shows the 'Elements' tab, and the bottom status bar indicates the file path: 'file:///Users/plugin/\_Dev/\_AngularJS/workspace/AngularJS\_Tests/WebContent/Wire'.

```

<div well="" ng-scope="">
  <h2>JavaScript Projects</h2>
  ...
</div>

```

And finally the local form looked just like the original:

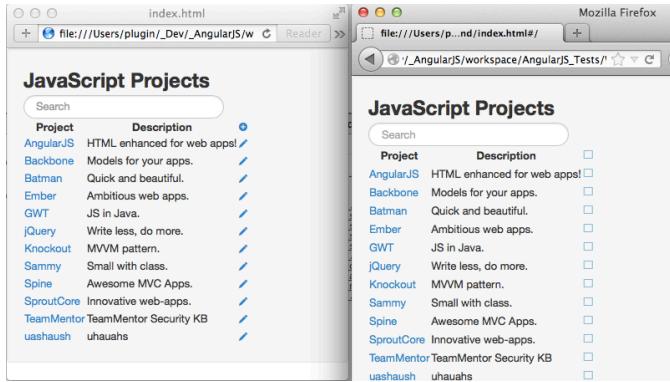
This screenshot shows the final result of the local development. The browser window displays the 'JavaScript Projects' page, identical to the one shown in the first screenshot. The 'well ng-scope' div is clearly visible, containing the project list. The right-hand panel shows the 'Elements' tab, and the bottom status bar indicates the file path: 'file:///Users/plugin/\_Dev/\_AngularJS/workspace/AngularJS\_Tests/WebContent/Wire'.

Project	Description
AngularJS	HTML enhanced for web apps!
Backbone	Models for your apps.
Batman	Quick and beautiful.
Ember	Ambitious web apps.
GWT	JS in Java.
jQuery	Write less, do more.
Knockout	MVVM pattern.
Sammy	Small with class.
Spine	Awesome MVC Apps.
SproutCore	Innovative web-apps.

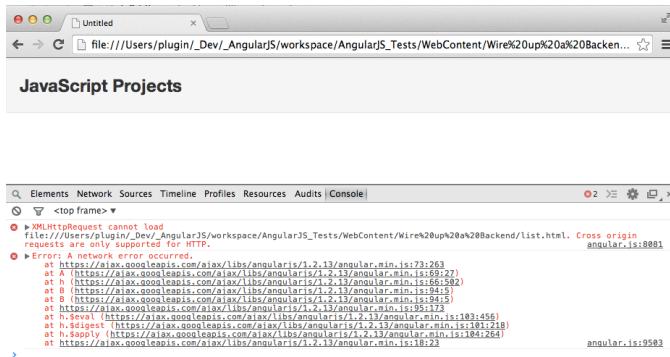
\*\*5) Running local sample on multiple browsers \*\*

At this stage I was curious on what this form would look like on multiple browsers, and found out something interesting.

Here is the page on Safari (left) and Firefox (right), which are loaded ok:

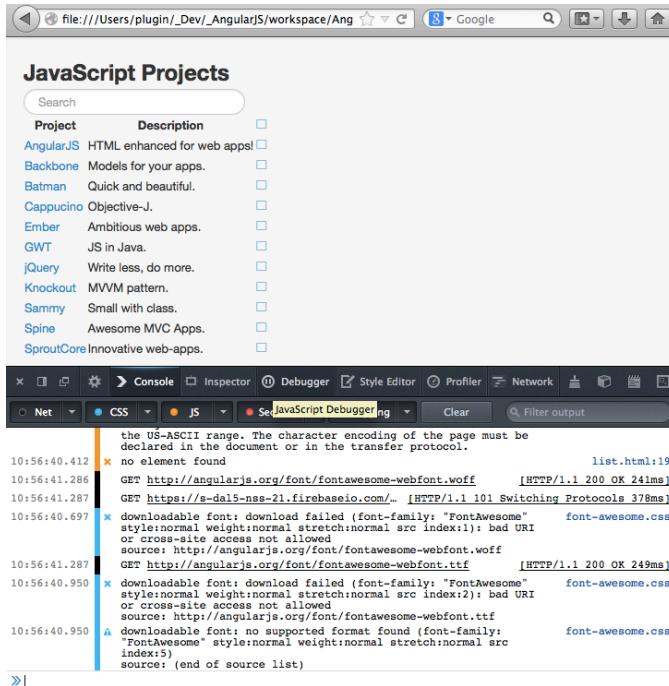


But on Chrome, it doesn't work at all, and the culprit is security!

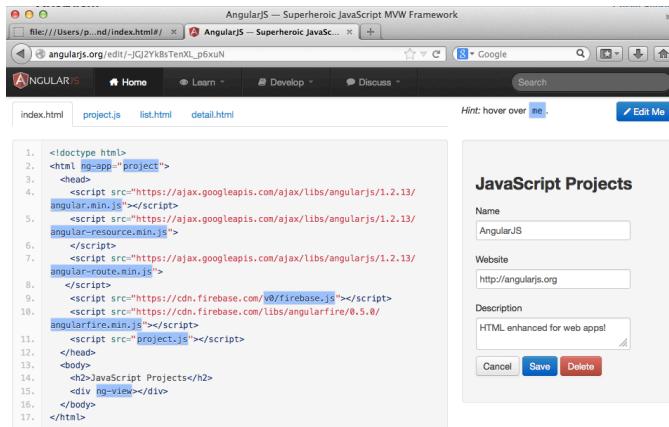


Basically chrome does not allow Cross origin requests from html files loaded from the local disk (which does make sense from a security point of view).

As a final little comment, in case you noticed that on the Firefox screenshot the Angular content was correctly displayed but the icons didn't show up, that much be a similar problem with loading **Bootstrap** or **Font Awesome** from the local disk



Here is the same example running on Firefox browser, so it works when loaded from http:



## Firebase

This example uses the [Firebase](#) which is a very powerful API to store and sync data in realtime.

After writing this post, I spend some time researching how it works and can be used.

Here are some posts I've written about Firebase:

- Using Firebase to sync data with a webpage (via Javascript, REST and Firebase Admin panel)
- XSS considerations when developing with Firebase
- Trying out Firebase (Beta) hosting solution and good example of Firebase Security rules
- First PoC of sending TeamMentor's server-side request URLs to Firebase (and seeing it in realtime in an AngularJS page)

- A really SIMPLE and clean AngularJS+Firebase example
  - (if you are reading this after Feb 2014) checkout the [Firebase section of this blog](#) for newer posts
- 

[Table of Contents](#) | [Code](#)

## 1.5 Using AngularJS in Eclipse, Part 4) Create Components

This is the last of four posts on how to run (inside Eclipse) the examples provided in [AngularJS's home page](#):

- [Using AngularJS in Eclipse, Part 1\) The Basics](#)
- [Using AngularJS in Eclipse, Part 2\) Add Some Control](#)
- [Using AngularJS in Eclipse, Part 3\) Wire up a Backend](#)
- Using AngularJS in Eclipse, Part 4) Create Components

The example covered on this post is the *Create Components*:

**Directives**  
Directives is a unique and powerful feature available only in Angular. Directives let you invent new HTML syntax, specific to your application.

**Reusable Components**  
We use directives to create reusable components. A component allows you to hide complex DOM structure, CSS, and behavior. This lets you focus either on what the application does or how the application looks separately.

**Localization**  
An important part of serious apps is localization. Angular's locale aware filters and stemming directives give you building blocks to make your application available in all locales.

**Code Snippets:**

```

<!DOCTYPE html>
<html ng-app="app">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
    <script src="components.js"></script>
    <script src="app.js"></script>
    <link rel="stylesheet" href="bootstrap.css">
  </head>
  <body>
    <tab>
      <pane title="Localization">
        Date: {{ '2012-04-01' | date:'fullDate' }} <br>
        Currency: {{ 123456 | currency }} <br>
        Number: {{ 98765.4321 | number }} <br>
      </pane>
      <pane title="Pluralization">
        <div ng-controller="BeerCounter">
          <div ng-repeat="beerCount in beers">
            <ng-pluralize count="beerCount" when="beerForms"></ng-pluralize>
          </div>
        </div>
      </pane>
    </tab>
  </body>
</html>

```

**Localization Examples:**

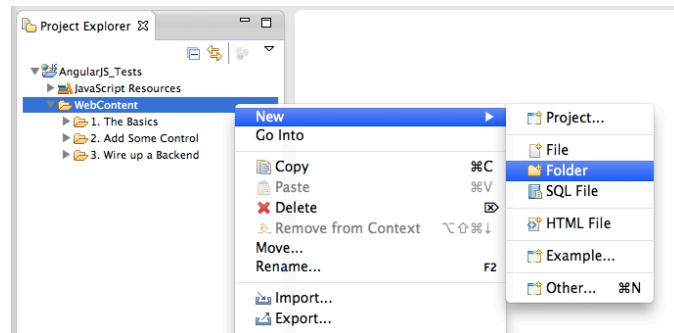
Locale: US
Date: Sunday, April 1, 2012 Currency: \$123,456.00 Number: 98,765,432

**Pluralization Examples:**

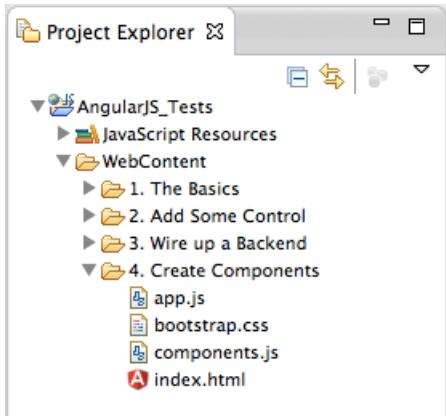
Locale: SK
Date: nedele, 1. aprília 2012 Currency: 123 456,00 € Number: 98 765,432

### 1) Creating the test files

First step is to create a folder called 4. *Create Components*:



... with the 4 files provided by this example:



### index.html

```
<!doctype html>
<html ng-app="app">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js">
    <script src="components.js"></script>
    <script src="app.js"></script>
    <link rel="stylesheet" href="bootstrap.css">
  </head>
  <body>
    <tabs>
      <pane title="Localization">
        Date: {{ '2012-04-01' | date:'fullDate' }} <br>
        Currency: {{ 123456 | currency }} <br>
        Number: {{ 98765.4321 | number }} <br>
      </pane>
      <pane title="Pluralization">
        <div ng-controller="BeerCounter">
          <div ng-repeat="beerCount in beers">
            <ng-pluralize count="beerCount" when="beerForms"></ng-pluralize>
          </div>
        </div>
      </pane>
    </tabs>
  </body>
</html>
```

### components.js

```

angular.module('components', [])
.directive('tabs', function() {
  return {
    restrict: 'E',
    transclude: true,
    scope: {}
  };
  controller: function($scope, $element) {
    var panes = $scope.panes = [];
    $scope.select = function(pane) {
      angular.forEach(panes, function(pane) {
        pane.selected = false;
      });
      pane.selected = true;
    }
    this.addPane = function(pane) {
      if (panes.length === 0) $scope.select(pane);
      panes.push(pane);
    }
  },
  template:
    '<div class="tabbable">' +
    '<ul class="nav nav-tabs">' +
    '<li ng-repeat="pane in panes" ng-class="{active:pane.selected}">' +
    '<a href="#" ng-click="select(pane)">{{pane.title}}</a>' +
    '</li>' +
    '</ul>' +
    '<div class="tab-content" ng-transclude></div>' +
    '</div>',
    replace: true
  };
})
.directive('pane', function() {
  return {
    require: '^tabs',
    restrict: 'E',
    transclude: true,
    scope: { title: '@' },
    link: function(scope, element, attrs, tabsCtrl) {
      tabsCtrl.addPane(scope);
    },
    template:
      '<div class="tab-pane" ng-class="{active: selected}" ng-transclude>' +
      '</div>',
      replace: true
    };
});

```

### app.js

```

angular.module('app', ['components'])
.controller('BeerCounter', function($scope, $locale) {
  $scope.beers = [0, 1, 2, 3, 4, 5, 6];
  if ($locale.id === 'en-us') {
    $scope.beerForms = {
      0: 'no beers',
      one: '{0} beer',
      other: '{0} beers'
    };
  } else {
    $scope.beerForms = {
      0: 'žiadne pivo',
      one: '{0} pivo',
      few: '{0} pivá',
      other: '{0} piv'
    };
  }
});

```

and bootstrap.css

```

@import '//netdna.bootstrapcdn.com/bootstrap/2.0.2/css/bootstrap.min.css';

```

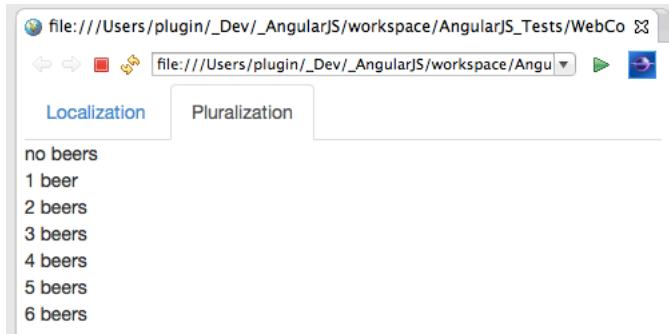
## 2) Running in browser (and failing to change locale)

Here is what it looks like when loaded locally (in Eclipse):



This example shows how we can create components (i.e directives ) that are able to handle multiple languages or currencies.

The Localization tab (shown above) is currently set for english (USA) and the Pluralization tab (shown below) is currently set to English.



This demo also has support for SK, but there didn't seem to be an easy way to change the browser's locale (so that we can see the SK values in action)

But on the [angular.org](http://angular.org) page they were able to show both US and SK side by side:

*Hint: hover over me .*

**Edit Me**

**Locale: US**

Localization      Pluralization

Date: Sunday, April 1, 2012  
Currency: \$123,456.00  
Number: 98,765.432

**Locale: SK**

Localization      Pluralization

Date: nedel'a, 1. apríla 2012  
Currency: 123 456,00 €  
Number: 98 765,432

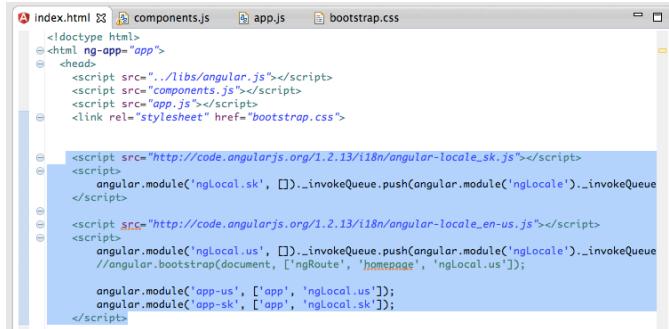
so there had to be a way to do it :)

### 3) programatically change the locale

Looking at the code from the [angular.org](http://angular.org) the code changes made below seem to be the ones required.

Basically we need to:

- load up the *angular-locale\_sk.js* file followed by immediately creating a module called *\_ngLocal.sk*
- load up the *angular-locale\_en-us.js* file followed by immediately creating a module called *\_ngLocale.us*
- create a module called *app-us* that depends/consumes *ngLocal.us*
- create a module called *app-sk* that depends/consumes *ngLocal.sk*



```

<!DOCTYPE html>
<html ng-app="app">
<head>
  <script src="../libs/angular.js"></script>
  <script src="components.js"></script>
  <script src="app.js"></script>
  <link rel="stylesheet" href="bootstrap.css">

  <script src="http://code.angularjs.org/1.2.13/i18n/angular-locale_sk.js"></script>
  <script>
    angular.module('ngLocal.sk', [])
      .invokeQueue.push(angular.module('ngLocale')._invokeQueue)
  </script>
  <script src="http://code.angularjs.org/1.2.13/i18n/angular-locale_en-us.js"></script>
  <script>
    angular.module('ngLocal.us', [])
      .invokeQueue.push(angular.module('ngLocale')._invokeQueue)
      //angular.bootstrap(document, ['ngRoute', 'homepage', 'nglocal.us']);

    angular.module('app-us', ['app', 'ngLocal.us']);
    angular.module('app-sk', ['app', 'ngLocal.sk']);
  </script>

```

Then we can change the *ng-app* to be *app-sk*:



```

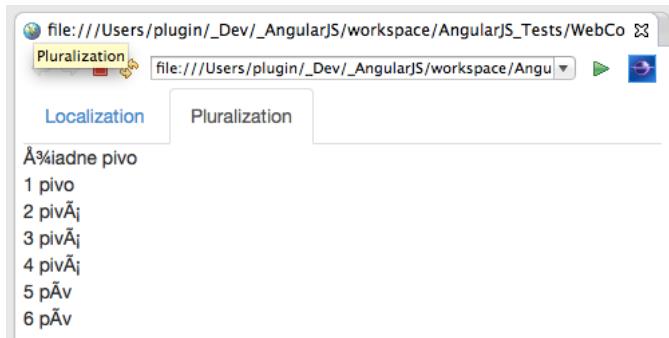
<!DOCTYPE html>
<html ng-app="app-sk">
<head>
  <script src="../libs/angular.js"></script>
  <script src="components.js"></script>
  <script src="app.js"></script>
  <link rel="stylesheet" href="bootstrap.css">

```

... and now the locale is SK (i.e. Slovak):



... with the text showing in Slovak (well I hope it does, since I don't speak Slovakian :))



If we wanted to have the site back in english (US), we can change the *ng-app* to be *app-us*

```

<!doctype html>
<html ng-app="app-us">
  <head>
    <script src="../libs/angular.js"></script>
    <script src="components.js"></script>
    <script src="app.js"></script>
    <link rel="stylesheet" href="bootstrap.css">

    <script src="http://code.angularjs.org/1.2.13/i18n/angular-localization.js"></script>
    <script>
      angular.module('ngLocal.sk', [])._invokeQueue();
    </script>
    <script src="http://code.angularjs.org/1.2.13/i18n/angular-pluralization.js"></script>
    <script>
      angular.module('ngLocal.us', [])._invokeQueue();
      angular.bootstrap(document, ['ngRoute', 'ngResource']);
      angular.module('app-us', ['app', 'ngLocal.us'])
      angular.module('app-sk', ['app', 'ngLocal.sk']);
    </script>
  </head>
  <body>
    <div ng-view></div>
  </body>
</html>

```

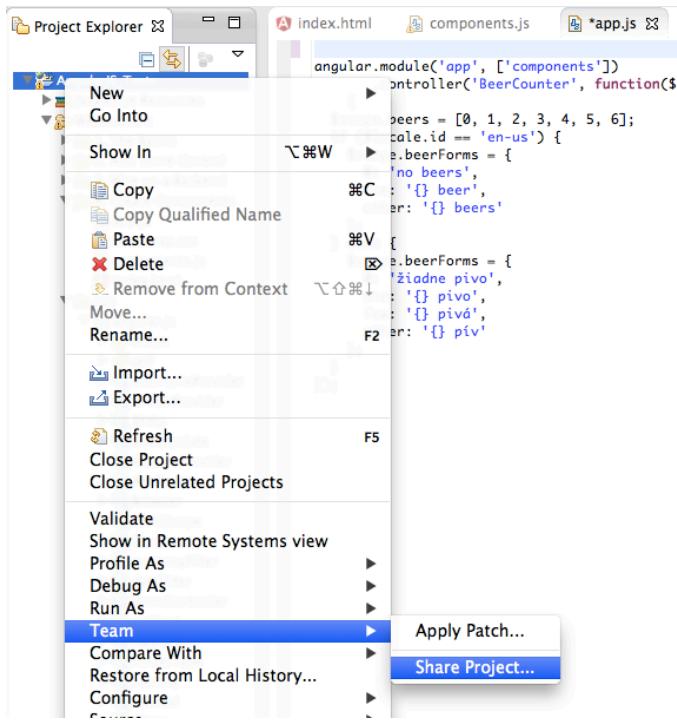
... and we're back in USA currency and English language:

Count	Plural Form
0	no beers
1	1 beer
2	2 beers
3	3 beers
4	4 beers
5	5 beers
6	6 beers

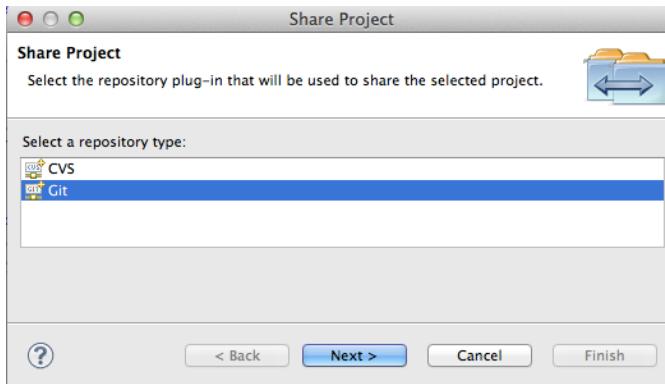
#### 4) viewing Git Diffs in Eclipse

At this stage I wanted to see git diffs **\*\*inside eclipse (for example ‘current changes vs last committed code’)**, but the option I was looking for (*‘Compare With -> HEAD Revision’*) was not there.

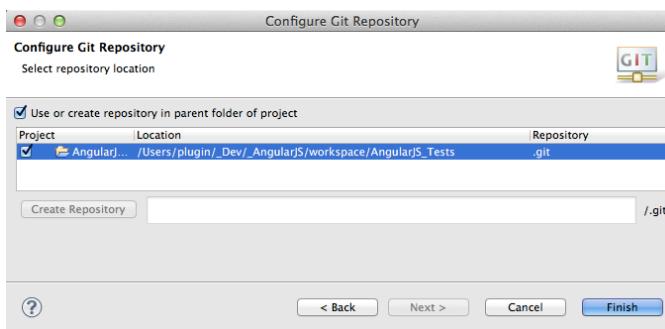
The problem was that I needed to \_‘Share current project’, \_which can be done via the **Team** menu:



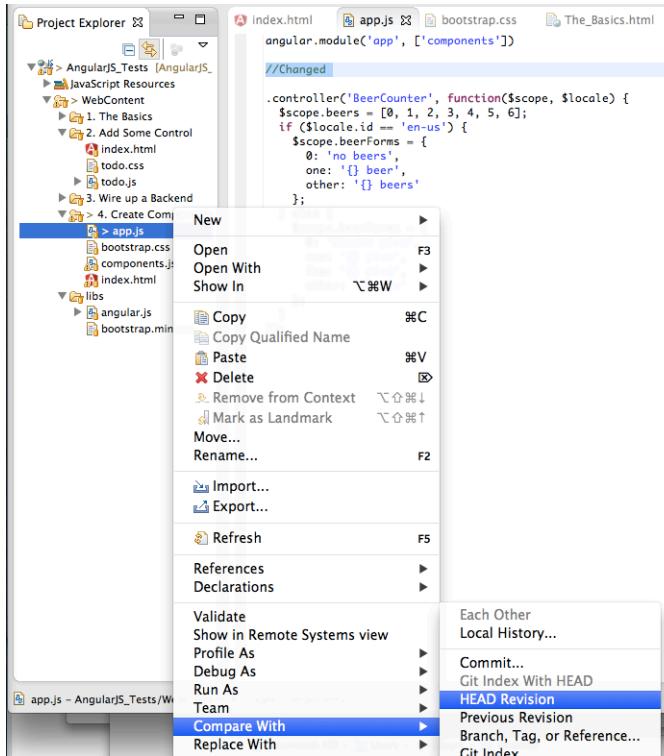
After choosing the Share Project ... \*\*menu option, \*\*I was asked to chose the Git repository type (which is a bit weird since this was already a Git repository)



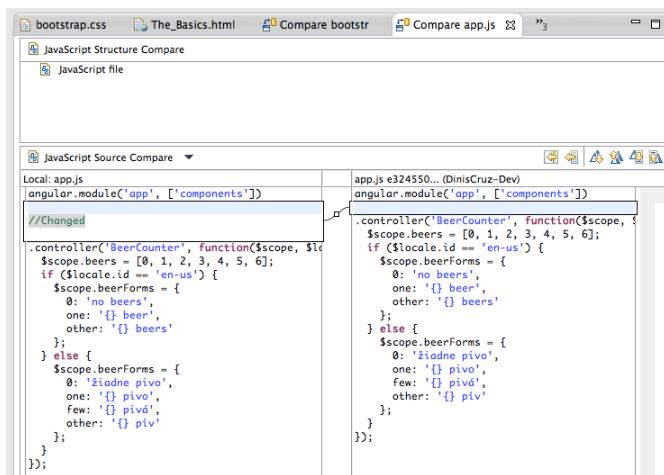
... and chose to use the repository in the parent folder of the selected project:



Once that completed, I was able to see the **Compare With -> HEAD Revision** option:



... which provides this really nice *Git diff* UI:



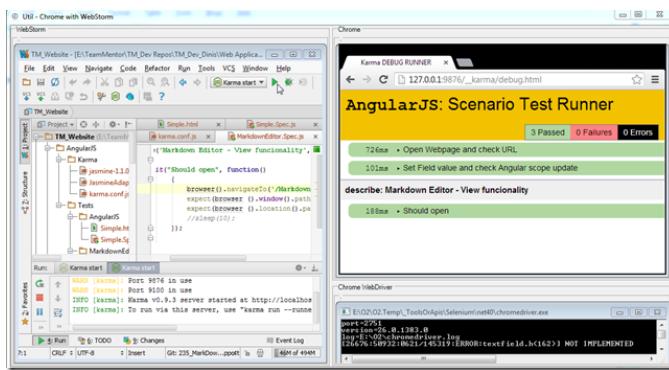
## 1.6 AngularJS code editor using UI-Bootstrap and CodeMirror (done without using jQuery)

I'm adding a number of [AngularJS](#) views to [TeamMentor](#), and here is a simple HTML based source code editor inspired on the [How to Integrate Codemirror With Angular UI](#) post and [ui-codemirror](#) repository.

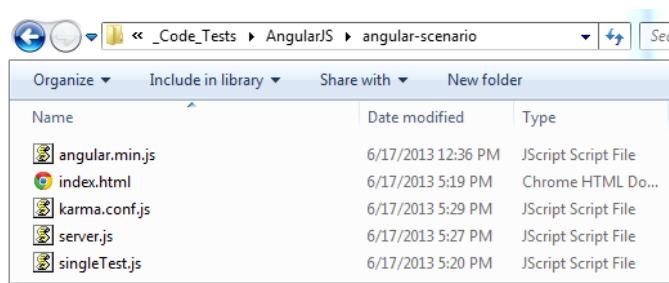
In the end, these are the APIs I used:

- <http://angularjs.org/>
- <http://angular-ui.github.io/bootstrap/>
- <http://twitter.github.io/bootstrap>
- <http://codemirror.net/> (just the core bit)

And this is what it looks like:



The source code editor is showing the contents of the current page (dynamically fetched using Angular `$http.get`) and the bottom yellow div is showing (in real-time) the contents of the source code editor:



**What is nice about this example is that I didn't use jQuery at all!**

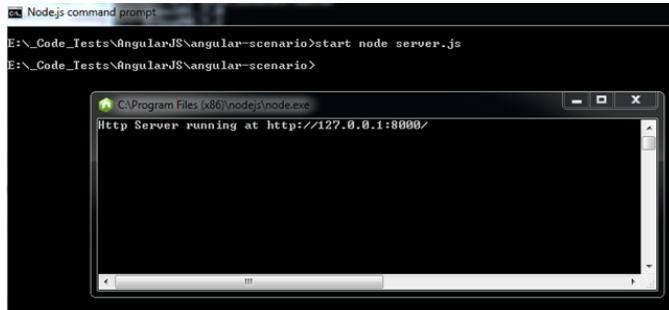
The great posts <http://stackoverflow.com/a/15012542> and [AngularJS for jQuery Developers](#) explain why learning to code in Angular without JQuery is so important.

Basically, it's better not have jQuery available, since them, the only way to solve a problem, is the 'AngularJS way' :)

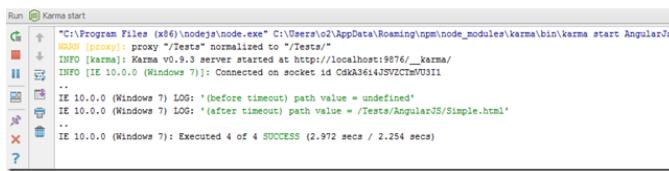
**How it works:**

Here is a brief explanation of the code behind this PoC (included in full at the end of this page):

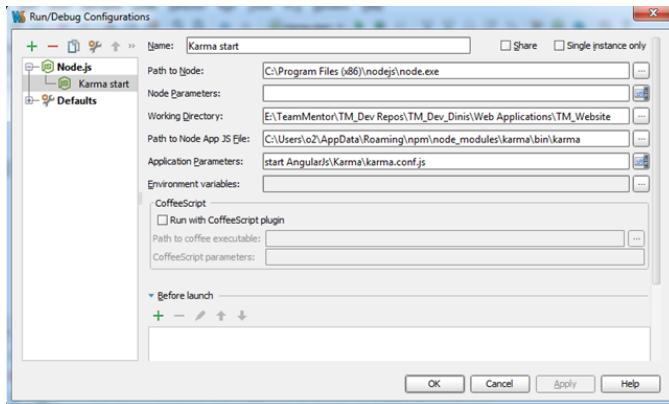
First we load the Javascript and CSS:



... them set up *AngularJS* module and *codeMirror* value



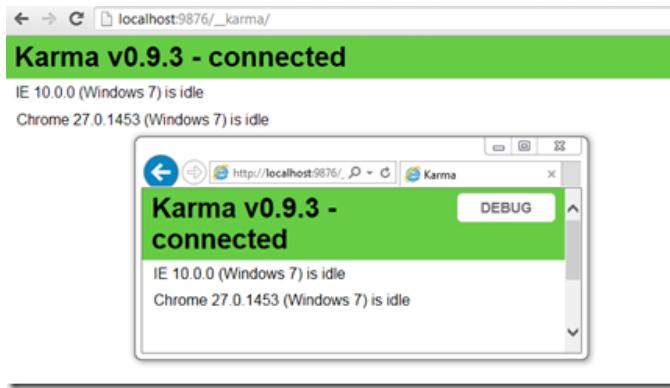
... use a *controller* to get the code to show (using *\$http.get*) and assign it to the *the\_ \$scope.code\_* variable



... configure angularJS in the HTML by setting the *textarea* element to be a *codemirror* (linked to the *\*\*\$scope.code \*\*model*)



... finally show the current value of *\$scope.code* in side an bootstrap *alert* element



### Complete Source code:

```
1  <!DOCTYPE html>
2
3  <html>
4      <head>
5          <title>CodeMirror with AngularJS</title>
6          <link href="/Content/bootstrap.min.css" rel="stylesheet">
7          <script src="/Scripts/angular.min.js" type="text/javascript"></script>
8          <script src="/Scripts/angular-ui.js" type="text/javascript"></script>
9          <script src="/Scripts/ui-bootstrap-tpls-0.3.0.min.js" type="text/javascript"><scr\
10     ipt>
11
12     <link href="/Content/codemirror-3.01/codemirror.css" rel="stylesheet"/>
13     <link href="/Content/codemirror-3.01/theme/rubyblue.css" rel="stylesheet"/>
14     <script src="/Scripts/codemirror-3.01/codemirror.js" type="text/javascript"><scri\
15     pt>
16     <script src="/Scripts/codemirror-3.01/mode/javascript.js" type="text/javascript"><\
17 /script>
18
19
20     <script type="text/javascript">
21         var myApp = angular.module('myApp', ['ui', 'ui.bootstrap']);
22
23         myApp.value('ui.config',
24             {
25                 codemirror:
26                     {
27                         mode: 'javascript',
28                         lineNumbers: true,
29                         matchBrackets: true,
30                         theme: 'rubyblue'
31                     }
32             });
33     </script>
```

```
33
34     function codeCtrl($scope,$http)
35     {
36         $scope.docLocation = document.location.href;
37         $http.get($scope.docLocation)
38             .success(function (data)
39             {
40                 $scope.code = data;
41             });
42         // $scope.code = "var a = 'somecode'; \n//which also shows above</h1>";
43     }
44 </script>
45 </head>
46 <body ng-app="myApp">
47     <div class="well well-large">
48         <div class="container">
49             <h2>CodeMirror working with AngularJS and Bootstrap</h2></div>
50         </div>
51         <div ng-controller="codeCtrl">
52             <div class="container">
53
54                 <h4>Code Editor:</h4>
55                 <p>With the the contents of this page (i.e.: {{docLocation}} )</p>
56
57                 <textarea ui-codemirror ng-model="code"></textarea>
58
59                 <br/><hr/><br/>
60
61                 <h4>Bootstrap alert style</h4>
62                 <p>
63                     Showing in real time the contents of the code shown above (make a chan\
64 ge to try it)
65                 </p>
66                 <alert type="success">{{code}}</alert>
67             </div>
68         </div>
69     </body>
70 </html>
```

## 2 KarmaJS

This section has the following chapters:

- [A small AngularJS Jasmine test executed by KarmaJS](#)
  - [Creating an Eclipse UI to run AngularJS e2e tests using Karma](#)
  - [Running KarmaJS's AngularJS example test/e2e/angular-scenario \(on Chrome\)](#)
- 

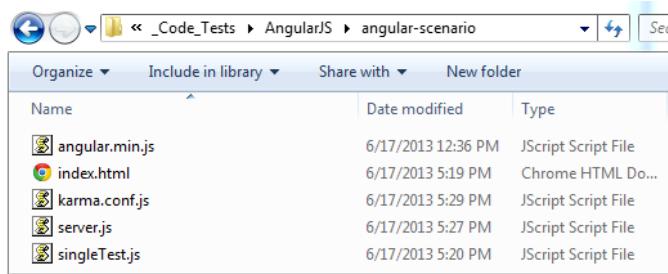
[Table of Contents](#) | [Code](#)

## 2.1 A small AngularJS Jasmine test executed by KarmaJS

When I try to understand how a particular technology works I always like to create a simple test case with a small number of moving parts.

This post shows such example for an AngularJS page, a Jasmine test, a NodeJS web server and a KarmaJS execution cycle.

The files used/customised were based on the KarmaJS [test/e2e/angular-scenario](#) example:



Let's look at each file's contents and what they do

1) **angular.min.js** is the latest version of AngularJS

2) **index.html** is a simple AngularJS example:

```

1  <!DOCTYPE html>
2  <html xmlns:ng="http://angularjs.org" id="ng-app" ng-app>
3  <head>
4      <meta charset="utf-8">
5      <title>Sample Angular App</title>
6      <script src="angular.min.js"></script>
7  </head>
8  <body>
9      <div>
10         <label>Name:</label>
11         <input type="text" ng-model="yourName" placeholder="Enter a name here">
12         <hr>
13         <h1>Hello {{yourName}}!</h1>
14     </div>
15 </body>
16 </html>
```

3) **karma.conf.js** is a reduced to the normal KarmaJS config file

```
1 module.exports = function(karma)
2 {
3     karma.configure(
4         {
5             // generic
6             frameworks : ['ng-scenario'],
7             urlRoot : '/__karma/',
8             autoWatch : true,
9             plugins : ['karma-ng-scenario'],
10
11            //project specific
12            proxies : { '/': 'http://localhost:8000/' },
13            files : ['singleTest.js'],
14
15            //run specific
16            singleRun : true,
17        });
18}
```

4) **singleTest.js** checks if AngularJS is working as expected

```
1 /** A Sample Angular E2E test */
2
3 describe('SimpleTest', function()
4 {
5
6     it('A small Angular Test', function()
7     {
8         browser().navigateTo('/index.html');
9         input('yourName').enter('A Pirate!');
10        expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!!');
11    });
12});
```

5) **server.js** is a working Web NodeJS server (also reduced for easier reading):

```
1 var sys = require('sys'),
2     http = require('http'),
3     fs = require('fs'),
4     url = require('url'),
5     events = require('events');
6
7 var DEFAULT_PORT = 8000;
8
9 function main(argv)
10 {
11     new HttpServer({
12         'GET': createServlet(StaticServlet),
13         'HEAD': createServlet(StaticServlet)
14     }).start(Number(argv[2]) || DEFAULT_PORT);
15 }
16
17 function escapeHtml(value)
18 {
19     return value.toString().
20     replace('<', '&lt;').
21     replace('>', '&gt;').
22     replace('"', '&quot;');
23 }
24
25 function createServlet(Class)
26 {
27     var servlet = new Class();
28     return servlet.handleRequest.bind(servlet);
29 }
30
31 /**
32 * An Http server implementation that uses a map of methods to decide
33 * action routing.
34 *
35 * @param {Object} Map of method => Handler function
36 */
37 function HttpServer(handlers)
38 {
39     this.handlers = handlers;
40     this.server = http.createServer(this.handleRequest_.bind(this));
41 }
42
43 HttpServer.prototype.start = function(port) {
44     this.port = port;
45     this.server.listen(port);
```

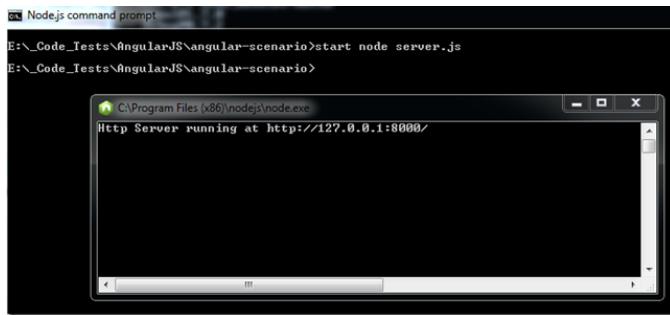
```
46     sys.puts('Http Server running at http://127.0.0.1:' + port + '/');
47 };
48
49 HttpServer.prototype.parseUrl_ = function(urlString) {
50     var parsed = url.parse(urlString);
51     parsed.pathname = url.resolve('/', parsed.pathname);
52     return url.parse(url.format(parsed), true);
53 };
54
55 HttpServer.prototype.handleRequest_ = function(req, res)
56 {
57     var logEntry = req.method + ' ' + req.url;
58     if (req.headers['user-agent'])
59     {
60         logEntry += ' ' + req.headers['user-agent'];
61     }
62     sys.puts(logEntry);
63     req.url = this.parseUrl_(req.url);
64     var handler = this.handlers[req.method];
65     if (!handler)
66     {
67         res.writeHead(501);
68         res.end();
69     } else
70     {
71         handler.call(this, req, res);
72     }
73 };
74
75 /**
76 * Handles static content.
77 */
78 function StaticServlet() {}
79
80 StaticServlet.MimeMap =
81 {
82     'txt': 'text/plain',
83     'html': 'text/html',
84     'css': 'text/css',
85     'xml': 'application/xml',
86     'json': 'application/json',
87     'js': 'application/javascript',
88     'jpg': 'image/jpeg',
89     'jpeg': 'image/jpeg',
90     'gif': 'image/gif',
```

```
91     'png': 'image/png',
92     'manifest': 'text/cache-manifest'
93 };
94
95 StaticServlet.prototype.handleRequest = function(req, res)
96 {
97     var self = this;
98     var path = ('./' + req.url.pathname).replace('//','/').replace(/%(..)/g, function(\
99 match, hex)
100     {
101         return String.fromCharCode(parseInt(hex, 16));
102     });
103     var parts = path.split('/');
104
105     fs.stat(path, function(err, stat)
106     {
107         if (err)
108             return self.sendMissing_(req, res, path);
109         return self.sendFile_(req, res, path);
110     });
111 };
112
113 StaticServlet.prototype.sendFile_ = function(req, res, path)
114 {
115     var self = this;
116     var file = fs.createReadStream(path);
117     res.writeHead(200,
118     {
119         // CSP headers, uncomment to enable CSP
120         // "X-WebKit-CSP": "default-src 'self';",
121         // "X-Content-Security-Policy": "default-src 'self'", 
122         'Content-Type': StaticServlet.
123         MimeMap[path.split('.').pop()] || 'text/plain'
124     });
125     if (req.method === 'HEAD')
126     {
127         res.end();
128     } else
129     {
130         file.on('data', res.write.bind(res));
131         file.on('close', function()
132         {
133             res.end();
134         });
135         file.on('error', function(error)
```

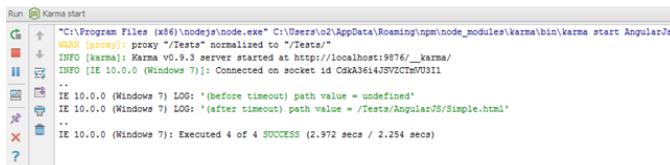
```
136     {
137         self.sendError_(req, res, error);
138     });
139 }
140 );
141
142 StaticServlet.prototype.sendError_ = function(req, res, error)
143 {
144     res.writeHead(500, {
145         'Content-Type': 'text/html'
146     });
147     res.write('<!doctype html>\n');
148     res.write('<title>Internal Server Error</title>\n');
149     res.write('<h1>Internal Server Error</h1>');
150     res.write('<pre>' + escapeHtml(sys.inspect(error)) + '</pre>');
151     sys.puts('500 Internal Server Error');
152     sys.puts(sys.inspect(error));
153 };
154
155 StaticServlet.prototype.sendMissing_ = function(req, res, path)
156 {
157     path = path.substring(1);
158     res.writeHead(404, {
159         'Content-Type': 'text/html'
160     });
161     res.write('<!doctype html>\n');
162     res.write('<title>404 Not Found</title>\n');
163     res.write('<h1>Not Found</h1>');
164     res.write('<p>The requested URL ' + escapeHtml(path) +
165             ' was not found on this server.</p>');
166     res.end();
167     sys.puts('404 Not Found: ' + path);
168 };
169
170 // Must be last,
171 main(process.argv);
```

### To see this in action

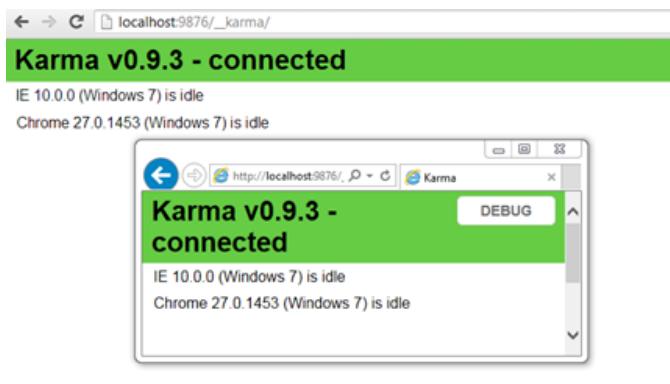
- 1) start the web server using *start node server.js*



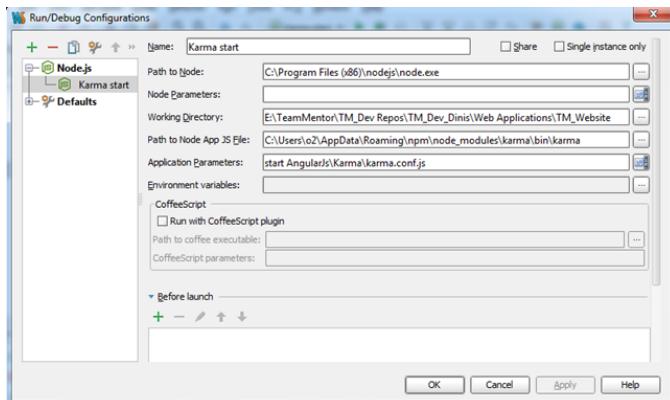
## 2) start karma with *karma start karma.conf.js*



## 3) on a local browser open [http://localhost:9876/\\_karma\\_](http://localhost:9876/_karma/) which will execute the test



Since we have **singleRun** \*\*set to true (in ***karma.conf.js***), the karma process ends after each execution, which means that the captured browsers will go into a ‘wait state’ (i.e. waiting for another karma server to come back to life)



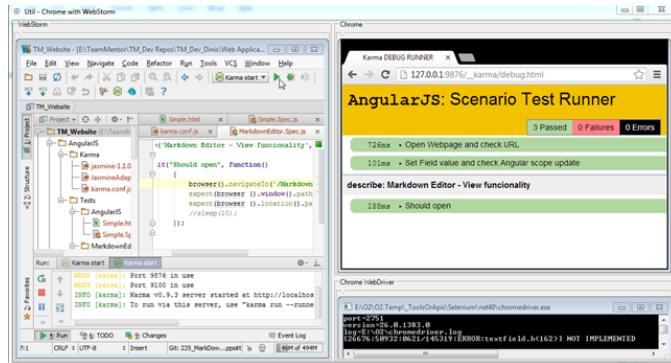
## 4) notice that back in karma, the tests executed ok:

```
E:\Code\Tests\AngularJS\angular-scenario>karma start karma.conf.js
[INFO] [Karma]: Karma v0.9.3 server started at http://localhost:9876/_karma/
[INFO] [Chrome 27.0.1453 (Windows 7)]: Connected on socket id 5WJYzHyc60uMrZ_-2v
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (0.368 secs / 0.184 secs)
```

So here it is, a pretty small example of a really powerful combination of technologies (and UnitTests workflows)

**\*\*NOTE:** while creating this post, I wrote an O2 Platform C# script to help \*\*

this script created a nice integrated test UI



Which allowed me to (in the same window) make changes and see its impact

**C# script of UI shown above**

```

1 var topPanel = "PoC - Karma and Angular run".popupWindow(1200,700);
2 //var topPanel = panel.clear().add_Panel();
3 //var textBox = topPanel.clear().add_RichTextBox();
4 var codeDir = @"E:\Code\Tests\AngularJS\angular-scenario";
5 var karma = @"C:\Users\o2\AppData\Roaming\npm\node_modules\karma\bin\karma";
6
7 var testPage = "http://localhost:8000/index.html";
8 var testRunner = "http://localhost:9876/_karma/";
9
10 var karmaConfig = codeDir.pathCombine("karma.conf.js");
11 var serverConfig = codeDir.pathCombine("server.js");
12 var unitTestFile = codeDir.pathCombine("e2eSpec.js");
13
14 Process karmaProcess = null;
15
16 Action startWebServer =
17     ()=>{
18         Processes.startProcessAndRedirectIO("node", serverConfig, codeDir, (line)=>line.\n19     info());
20     };
21
22 Action runKarma =
23     ()=>{
24         Action<string> consoleOut =
```

```
25         (consoleLine)=> consoleLine.info(); //textBox.append_Line(consoleLine);
26
27     var command = "start \"\{0}\\" ".format(karmaConfig);
28     karmaProcess = "node".startProcess("\\" + karma + "\\" " + command, consoleOut) \
29 ;
30     };
31
32 if (testPage.GET().notValid())
33 {
34     "Starting WebServer".info();
35     startWebServer();
36     1000.wait();
37 }
38
39 var toolStrip = topPanel.insert_Above_ToolStrip();
40 var codeEditor_Test = topPanel.add_SourceCodeEditor();
41 var ie_UnitTestRunner = codeEditor_Test.insert_Right().add_WebBrowser_with_NavigationBar() \
42 ;
43 var ie_Site = ie_UnitTestRunner.insert_Below().add_WebBrowser_with_NavigationBar();
44 var codeEditor_Config = codeEditor_Test.insert_Below().add_SourceCodeEditor();
45
46 codeEditor_Test.open(unitTestFile);
47 codeEditor_Config.open(karmaConfig);
48 ie_Site.open(testPage);
49 ie_UnitTestRunner.open(testRunner);
50 toolStrip.add_Button("Run", "btExecuteSelectedMethod_Image".formImage(), ()=>runKarma());
51
52
53 runKarma();
54
55 //using System.Diagnostics
```

---

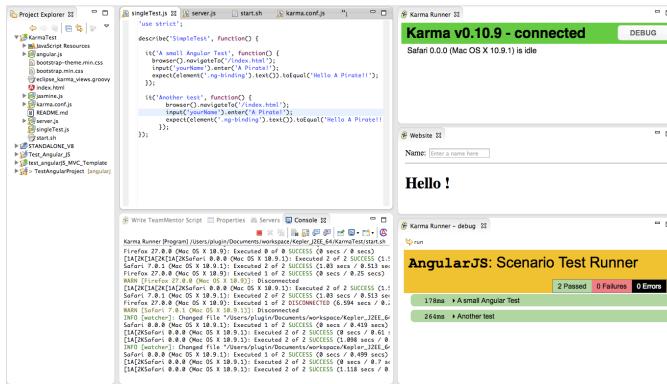
[Table of Contents](#) | [Code](#)

## 2.2 Creating an Eclipse UI to run AngularJS e2e tests using Karma

This post shows how I created a nice set of views in Eclipse to quickly see the execution result of AngularJS e2e (end-to-end) tests, without leaving the Eclipse UI.

The image below shows this UI in action, where:

- The source code of the test is shown in the Eclipse Java editor
- Just below is the **console out** of the Karma runner (which is detecting files changes)
- On the top-right is the hooked browser (i.e. the one that will run the tests)
- On the middle-right is the simple AngularJS **Hello World** page
- On the bottom-right is the debug view of the hooked browser (which is what you get if you click on the **Debug** Button included on the top-right page)



Here are the tools and Eclipse Plugins used to create this UI:

- Eclipse Kepler for Java EE Developers with:
  - Eclipse Groovy REPL Scripting Environment 1.6.0 (update site)
  - AngularJS Eclipse Plugin (update site)
  - Nodeclipse (update site)
  - Ansi Console (update site)
- NodeJs with
  - Karma

The code used on this sample is based on my past attempts of using Karma (see [A small AngularJS Jasmine test executed by KarmaJS](#), [Running KarmaJS's AngularJS example test/e2e/angular-scenario \(on Chrome\)](#) and [KarmaJS related posts](#))

Although the AngularJS and NodeJs Eclipse plugins provide nice helpers and views, they didn't provide (yet) the kind of UI that I was looking for. Namely they didn't support the use of KarmaJS to run AngularJS tests.

But since I now have the ability to quickly manipulate and create new Eclipse views without restarting Eclipse (using the [Groovy REPL script environment](#)) it was not very hard to create the UI that I wanted (see [Eclipse Plugin that allows the execution of REPL Groovy Scripts \(in the current Eclipse Instance\) and Fluent API for Eclipse+SWT](#)).

Basically the brief was to:

- Create new (Eclipse) view with a browser showing `http://localhost:9879/_karma/` (the KarmaJS browser runner/hook)
- Create new view with *a browser showing `http://localhost:9879/index.html`* (the AngularJS page)
- Create new view with a browser showing `http://localhost:9879/_karma/debug.html` (debug view of Karma runner), with an toolbar button to refresh it.

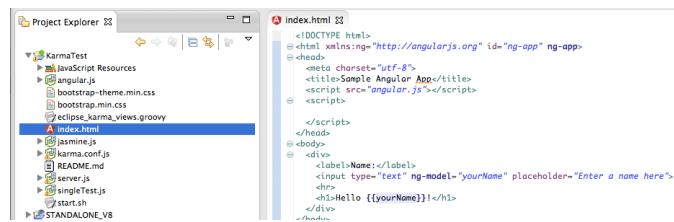
Here is [the gist](#) with the code that creates this 3 Eclipse views:

Hopefully this [Groovy](#) script is easier to read (the idea of the Fluent API that I added to the Groovy REPL was allow the easy reading and understanding of scripts like this).

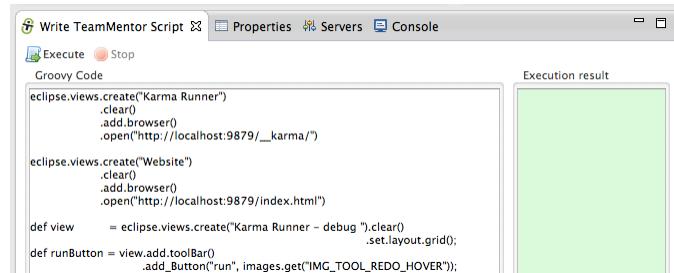
## 1) Setting up the environment

Now lets look at how this environment was setup:

We start with a simple AngularJS project test page that will just show the model-binding capabilities of AngularJS (yes I know that all those JS files should be in separate folders :))



Then I opened up [Groovy REPL script environment](#) and wrote [the script](#) (shown above) that creates the multiple views:



Clicking on `Execute` will create 3 views:

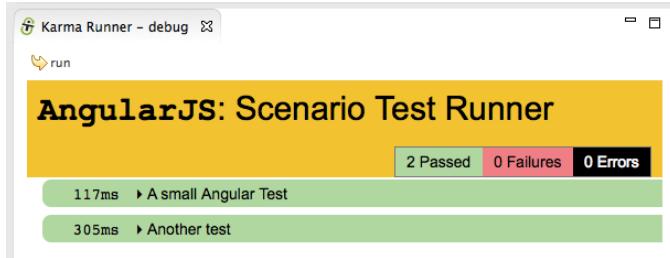
a) the *Karma Runner* view:



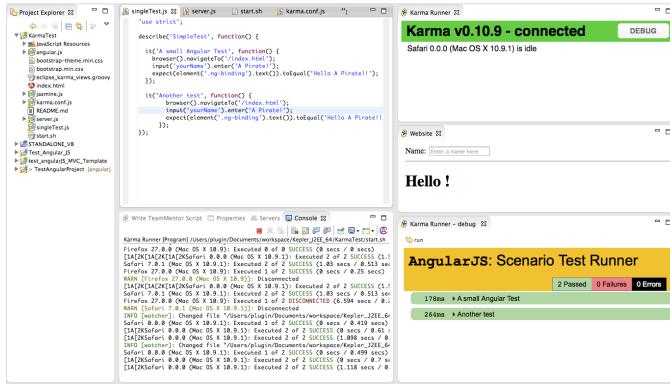
b) the *Website* view:



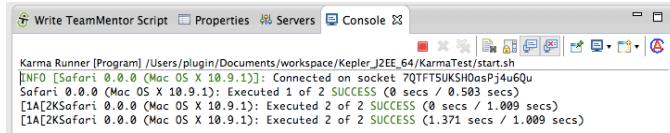
c) the **Karma Runner - debug** view:



... which (as seen in the first screenshot of this post) looks like this:



Part of this UI (but not created using the Groovy script) is an Eclipse *Console \_with the\_ 'Karma Runner process' **console** \*\*out\*\**:



The AngularJS tests were executed when the **Karma Runner** view was opened, because the Karma process (configured via Karma.config.js) is set to wait for new browser's hooks/runners and (on connection or code changes) execute the configured tests.

## 2) Making changes to existing tests

To see Karma in action, let's make a test fail :)

The current AngularJS page is basically just echoing/binding the contents of the **Name** TextBox into the **H1** tag:



Here is the test that is currently being executed, which is an e2e test that runs on the browser (just like Selenium or Watin).

```
'use strict';

describe('SimpleTest', function() {
  it('A small Angular Test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate!');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });

  it('Another test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate!');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });
});
```

Hopefully this type of Fluent API is easy to read:

- browser navigates to */index.html*
- value is inserted into the *yourName* input field (the TextBox)
- the value of the binded element (*.ng-binding*) is checked to see if it matches

To make this test fail, let's change just the \*\**yourName* \*\*value:

```
'use strict';

describe('SimpleTest', function() {
  it('A small Angular Test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate!');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });

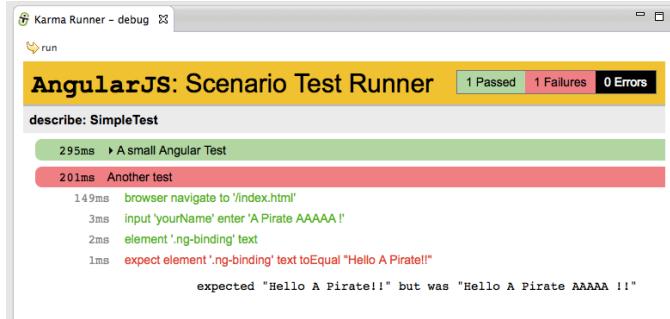
  it('Another test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate AAAA !!');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });
});
```

Immediately after saving the test with the changes shown above, Karma refreshes the hooked browsers in order to trigger the execution of the tests.

And as the image below will show, there is now one failed test:

Karma Runner [Program] /Users/plugin/Documents/workspace/Kepler\_J2EE\_64/KarmaTest/start.sh  
INFO [Safari 0.0.0 (Mac OS X 10.9.1)]: Connected on socket 7QTFTSUH0osPj4u6Qu  
Safari 0.0.0 (Mac OS X 10.9.1): Executed 1 of 2 SUCCESS (0 secs / 0.503 secs)  
[1A2K]Safari 0.0.0 (Mac OS X 10.9.1): Executed 2 of 2 SUCCESS (0 secs / 1.009 secs)  
[1A2K]Safari 0.0.0 (Mac OS X 10.9.1): Executed 2 of 2 SUCCESS (1.371 secs / 1.009 secs)  
INFO [watcher]: Changed file "/Users/plugin/Documents/workspace/Kepler\_J2EE\_64/KarmaTest/singleTest.  
Safari 0.0.0 (Mac OS X 10.9.1): Executed 1 of 2 SUCCESS (0 secs / 1.253 secs)  
[1A2K]Safari 0.0.0 (Mac OS X 10.9.1): SimpleTest Another test FAILED  
expect element '.ng-binding' text toEqual "Hello A Pirate!!"  
expected "Hello A Pirate!!" but was "Hello A Pirate AAAAAA !!"  
Safari 0.0.0 (Mac OS X 10.9.1): Executed 2 of 2 (1 FAILED) (0 secs / 1.725 secs)  
[1A2K]Safari 0.0.0 (Mac OS X 10.9.1): Executed 2 of 2 (1 FAILED) (2.143 secs / 1.725 secs)

In cases like this, the **\_Karma Runner - debug** is very useful since it will show more details on what happened, and where the test failed:



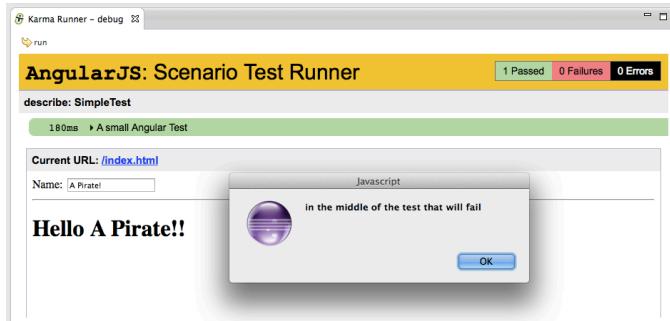
Just to confirm that the page is actually being loaded and the tests are happening in a real browser instance, if we add an Javascript alert to the current test:

```
'use strict';

describe('SimpleTest', function() {
  it('A small Angular Test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate!');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });

  it('Another test', function() {
    browser().navigateTo('/index.html');
    input('yourName').enter('A Pirate AAAAAA !!');
    alert('in the middle of the test that will fail');
    expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
  });
});
```

... we will get an alert box in the **Karma Runner - debug** (note that on the screenshot below we're seeing the image of the final state of execution of the previous test (in this case the '*A small Angular Test*')

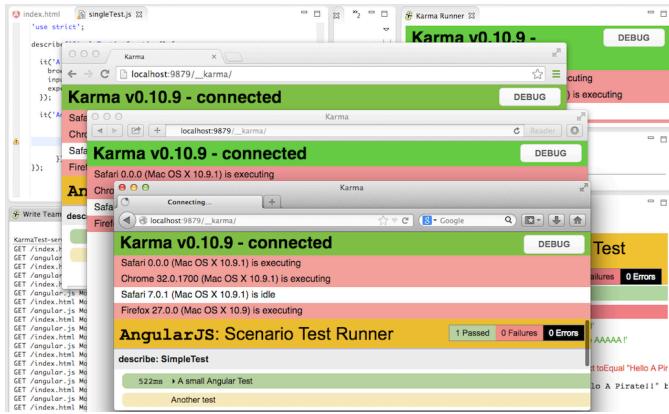


### 3) Hooking multiple browsers and only failing on Firefox

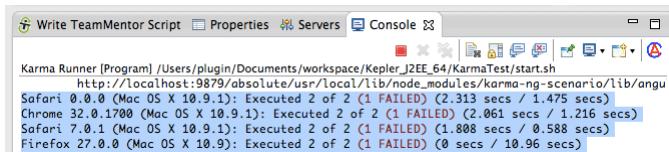
In order to run the tests in multiple browsers all we need to do is to open the `http://localhost:9879/_karma/` page (in the target browser), and a test execution will be triggered.

Note how on the image below:

- Eclipse is in the background (containing the views previously created and showing the **console out** of the Karma test runner process)
- There are 3 stand lone browser windows (Chrome, Safari and Firefox)
- Firefox (top most window) shows the tests being executed (in green the test executed, in yellow the test being executed)

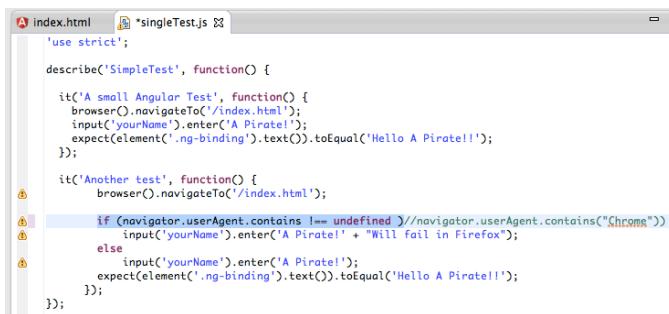


After execution, a quick look at the Karma runner shows that the modified test failed (as expected) on all browsers:



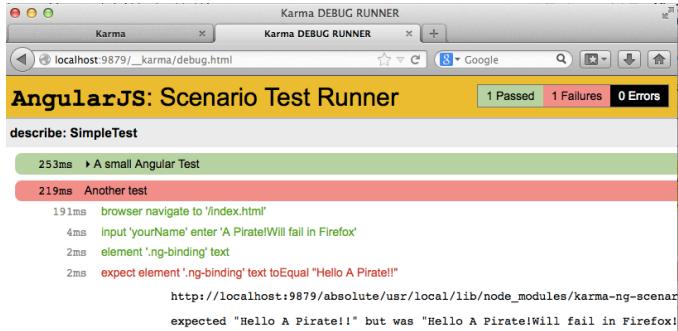
Just to make sure all is working as expected, let's create a test that will fail only in one browser.

For example Firefox is the only one that has defined the **navigator.userAgent.contains** Javascript function (so we can use this to detect Firefox, and create a test that will only fail if executed inside it):



After saving the changes, the tests will be executed on all 4 browsers, with the one test that failed being the one executed in Firefox:

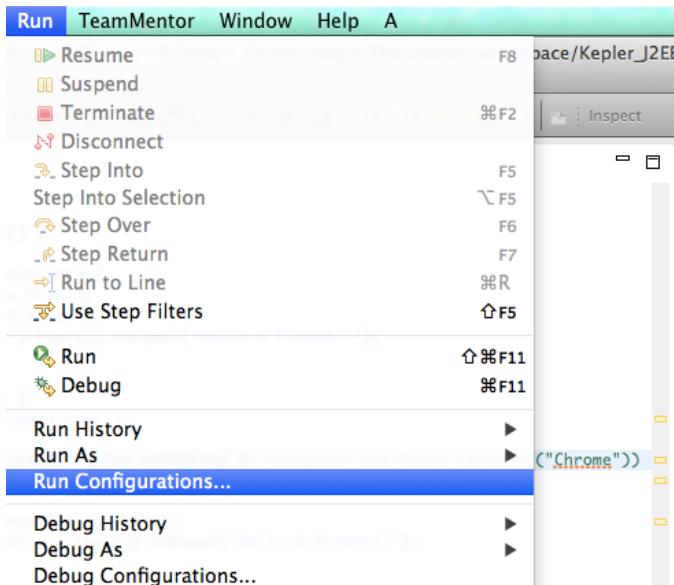
Refreshing the Firefox debug runner, shows the failed test and assert:



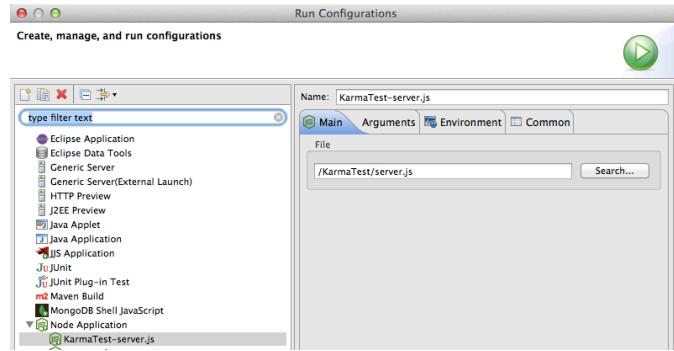
## 4) Setup

In addition to creating the views programatically, I also setup an Eclipse\_ Run configurations \*\* for \*\* - NodeJS and an \_External Tools Configuration \_ for KarmaJS.

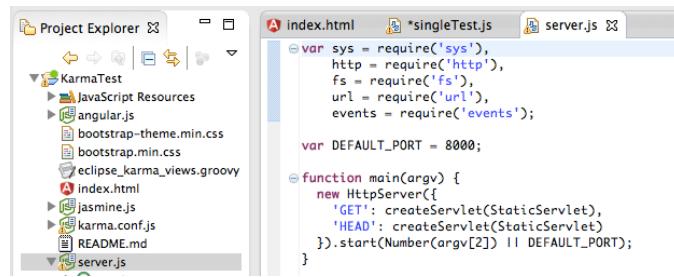
The NodeJS configuration was done here:



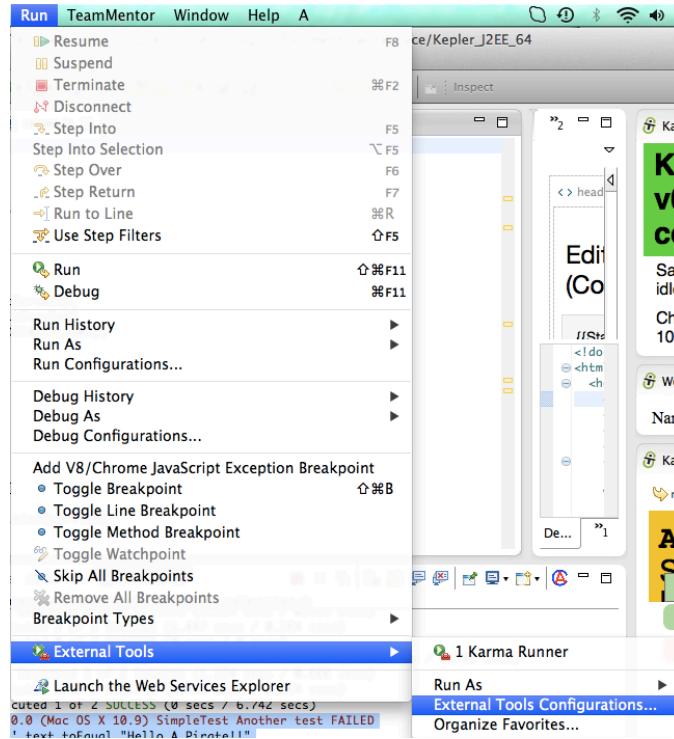
... where I created a Node Application runner that was pointed to *server.js*.



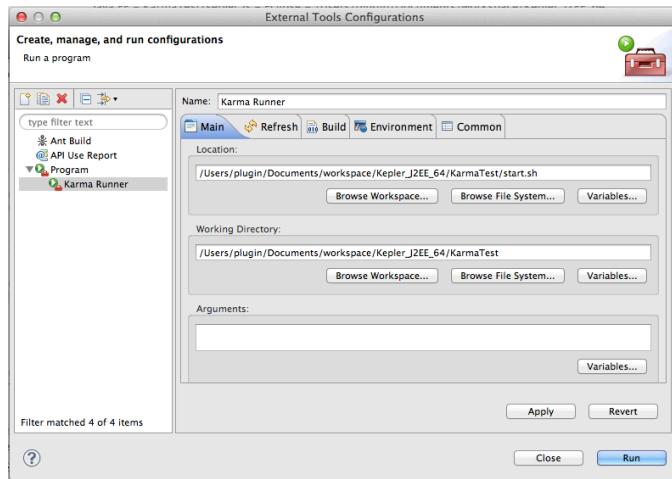
... which contains a simple NodeJS web server (we could also have used NodeJS Express, but that would have added a lot of other 'stuff' to this project)



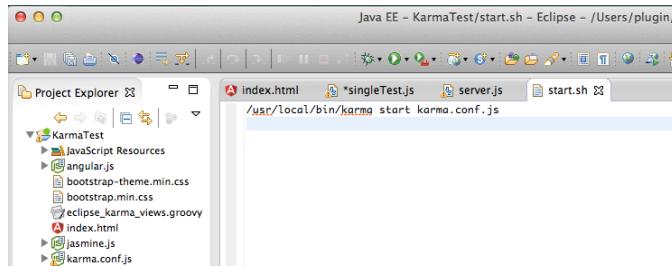
The KarmaJS config was done as an *External Tools Configuration*



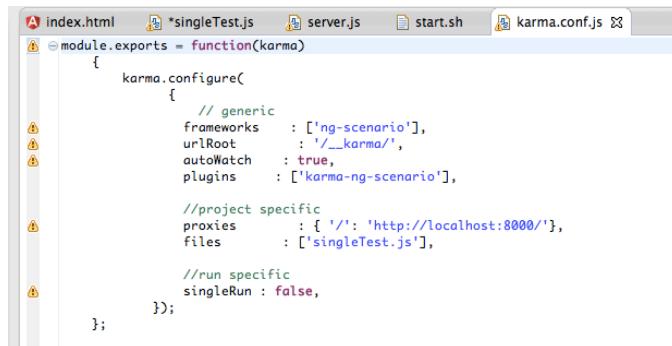
... where I simply run the ***start.sh*** bash file



... which starts **KarmaJS**:



... using this simple **karma.conf.js** configuration



Note: I started the Karma process like this because there was an issue with **KarmaJS** finding the \*\*NodeJS executable from Eclipse (and at the time I didn't had time to debug why that was happening)

#### External References:

- [AngularJS docs on E2E Testing](#)
- [AngularJS Testing with Karma and Jasmine](#)
- [Test runners for AngularJS - how to run the tests from eclipse IDE and CI server without too much complication?](#)
- [How to run KarmaJS from Eclipse](#)
- [Setting-up AngularJS, Angular Seed, Node.js and Karma](#)

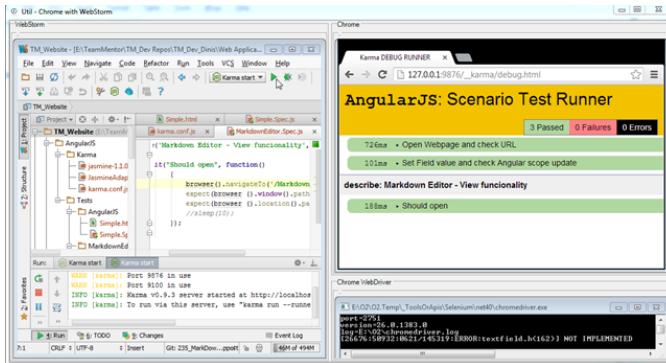
- Can't run e2e tests with karma (for angular)
- 

[Table of Contents](#) | [Code](#)

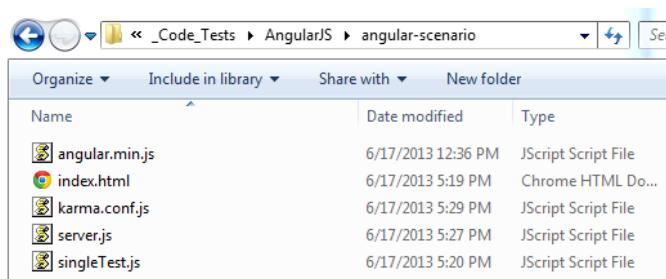
## 2.3 Running KarmaJS's AngularJS example test/e2e/angular-scenario (on Chrome)

To learn and get an idea of how **Karma** (the ‘*Spectacular Test Runner for JavaScript*’) works, and how it can be used to create browser automation tests, here are the steps I took to get the [test/e2e/angular-scenario](#) example to work.

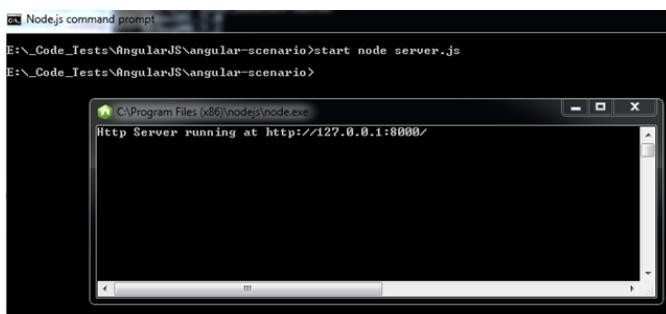
It all started with a clone of: <git@github.com:karma-runner/karma.git>



With this KarmaJS test example (see below), being the one that we are going to use:



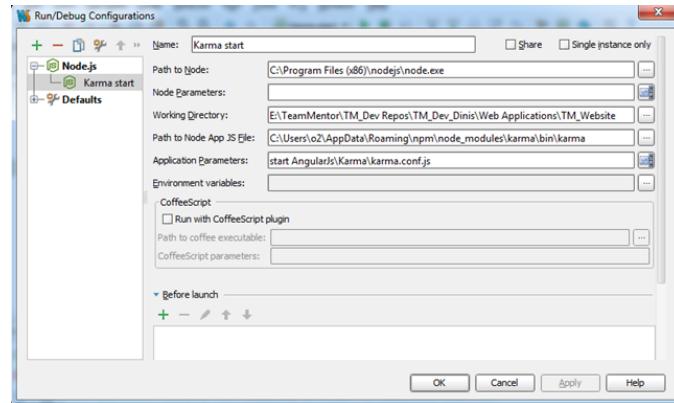
I then opened an nodejs command prompt and navigated to the folder shown above:



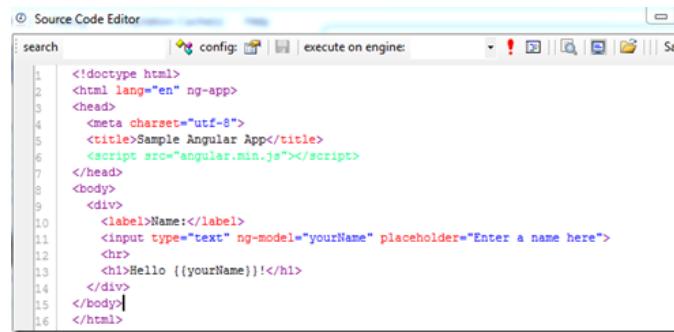
... used `_node server.js` to start a local webserver



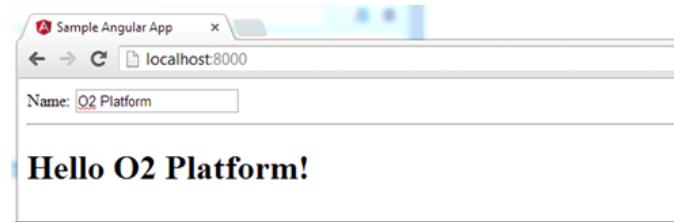
... on port 8000:



The test case we are using (on KarmaJS's *test/e2e/angular-scenario*) is a simple AngularJS example, which just consumes the angular-min.js file model attribute:



... and uses angular to populate the `{{yourName}}` value dynamically (wired to the input field via the `ng-model="yourName"`)



Next we are going to run this [Jasmine](#) ('Behavior-Driven Development framework for testing JavaScript code') test using KarmaJS

```

1  /** A Sample Angular E2E test */
2
3  describe('My Sample App', function() {
4
5    it('should let Angular do its work', function() {
6      browser().navigateTo('/index.html');
7      input('yourName').enter('A Pirate!');
8      expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!');
9    });
10
11  xit('should skip this e2e test', function() {
12     sleep(15);
13     browser().navigateTo('/index.html');
14   });
15 });
16

```

... which *should* work with just: karma start karma.conf.js

```

E:\_Code\Tests\AngularJS\karma\test\e2e\angular-scenario>karma start karma.conf.js
[2013-06-17 15:14:02.789] [ERROR] config  Invalid config file!
[ReferenceError: module is not defined
ReferenceError: module is not defined
at evalMachine<anonymous>:1:1
at Object.parseConfig <C:\Users\v2\appData\Roaming\npm\node_modules\karma\lib\config.js:158:8
at Object.exports.start <C:\Users\v2\appData\Roaming\npm\node_modules\karma\lib\server.js:19:20
at Object.<anonymous> <C:\Users\v2\appData\Roaming\npm\node_modules\karma\bin\karma:28:39
at Module._compile <module.js:456:26
at Object.Module._extensions..js <module.js:474:10
at Object.Object.defineProperty无助于解决这个问题。
at Function.Module._load <module.js:312:12
at Function.Module.runMain <module.js:497:10

```

... but it didn't

There is a module dependency missing, which in this case can be resolved by running this command from the root of the karma repository:

```

E:\_Code\Tests\AngularJS\karma\test>cd ..
E:\_Code\Tests\AngularJS\karma\test>cd ..
E:\_Code\Tests\AngularJS\karma>karma start test/e2e/angular-scenario>karma.conf.js
[00:00:00.000] [Karma]: Port 9876 in use
[00:00:00.000] [Karma]: Port 9876 in use
[INFO] [Karma]: Karma v0.9.3 server started at http://localhost:9876/_karma/
[INFO] [Karma]: Starting browser Chrome
[INFO] [Karma]: Connected on socket id hc37aP-60tflq4EXKxUZ
[Chrome 27.0.1453 (Windows 7)]: My Sample App should let Angular do its work FAILED


[Chrome 27.0.1453 (Windows 7)]: Executed 1 of 1 <1 FAILED> <0.936 secs / 0.141 secs>

```

**UPDATE:** the issue above was caused by the fact that I had an the official released version of karma installed globally which is the one that was running when I tried it on the 'test/e2e/angular-scenario' folder'

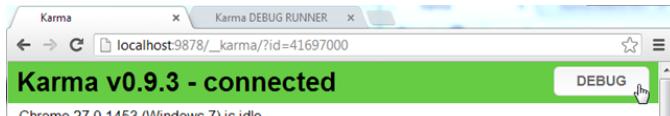
And now (based on an option from the karma.conf.js) a Chrome window opened up:

```

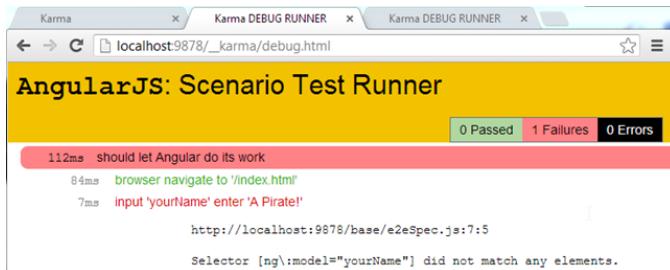
E:\_Code\Tests\AngularJS\karma>cd ..
E:\_Code\Tests\AngularJS\karma>karma start angular-scenario>karma.conf.js
[2013-06-17 15:14:02.349] [ERROR] config  Invalid config file!
[ReferenceError: module is not defined
ReferenceError: module is not defined
at evalMachine<anonymous>:1:1
at readConfigFile <C:\Users\v2\appData\Roaming\npm\node_modules\karma\lib\config.js:158:8
at Object.parseConfig <C:\Users\v2\appData\Roaming\npm\node_modules\karma\lib\config.js:177:19
at Object.exports.start <C:\Users\v2\appData\Roaming\npm\node_modules\karma\lib\server.js:19:20
at Object.<anonymous> <C:\Users\v2\appData\Roaming\npm\node_modules\karma\bin\karma:28:39
at Module._compile <module.js:456:26
at Object.Module._extensions..js <module.js:474:10
at Object.Object.defineProperty无助于解决这个问题。
at Function.Module._load <module.js:312:12
at Function.Module.runMain <module.js:497:10

```

Clicking on the *Debug* button:

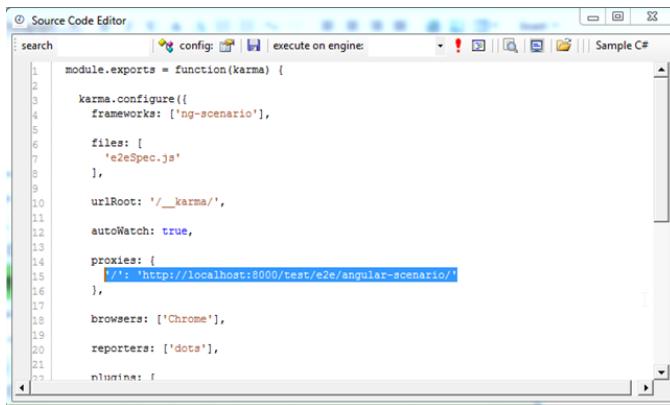


... we can more details on the test that failed:

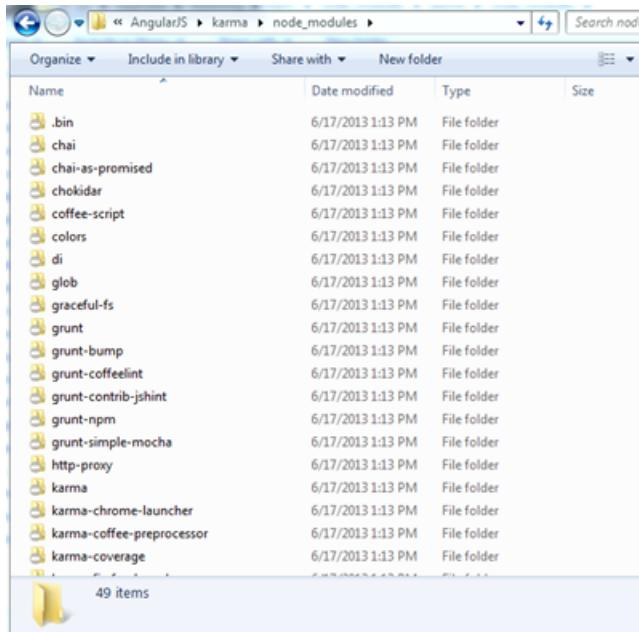


In this case the problem is that the ‘proxy mapping’ that karma does is not correct

If we look at the karma.config.js file



.... and the unit test *brower().navigateTo* command



... we can see that karma will try to open */index.html* from *http://localhost:8000/test/e2e/angular-scenario/index.html*

That is not going to work since the page we want it is on <http://localhost:8000/index.html> (which happened because we started \_node server.js \_on the\*\* .../test/e2e/angular-scenario \*\*folder)

```
mock-httpr-response@0.1.1 node_modules\mock-httpr-response
+-- which@0.5 node_modules\which
+-- sinon-chai@2.3.1 node_modules\sinon-chai
+-- chai-as-promised@3.2.5 node_modules\chai-as-promised
+-- grunt-simple-mocha@0.4.0 node_modules\grunt-simple-mocha
+-- mocha@0.11 node_modules\mocha
+-- chai@1.5.0 node_modules\chai
+-- timer-shim@0.2.3-1 node_modules\timer-shim
+-- linkedlist@1.0.1
+-- qq@0.3.5 node_modules\qq
+-- q@0.8.4
+-- mocha@1.8.2 node_modules\mocha
+-- growl@1.2.0
+-- debug@0.7.2
+-- commander@0.6.1
+-- dnock@0.1.2
+-- md5@0.3.3
+-- ms@0.3.0
+-- Jade@0.26.3 <nodejs@0.3.0>
grunt@0.4.1 node_modules\grunt
+-- hooker@0.2.3
+-- asynclib@0.22
+-- event@0.4.12
+-- nopt@0.1.0 <abbrev@0.1.0.4>
+-- rimraf@0.3.0 <graceful-fs@1.1.14>
+-- concat-stream@0.3.3
+-- underscore@string@2.2.0rc
+-- iconv-lite@0.2.10
+-- lodash@0.9.2
+-- nodeunit@0.1.2 <lodash@0.1.0.1>
+-- js-yaml@2.0.5 <esprima@1.0.3, argparse@0.1.15>
sinon@1.6.0 node_modules\sinon
+-- buster-format@0.5.5 <buster-core@0.6.4>
+-- grunt-contrib-jshint@0.3.0 node_modules\grunt-contrib-jshint
+-- jshint@1.0 <peakle@0.0.1, clobber@0.4-2, underscore@1.4.4, shelljs@0.1.4, esprima@1.0-dev>
```

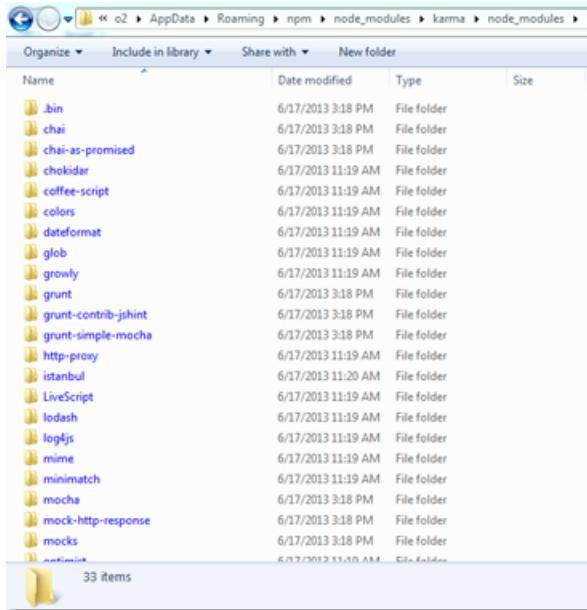
One way to solved this, is to change the ‘proxies’ mapping to ‘/’: ‘<http://localhost:8000/>’

```
C:\Users\o2\appData\Roaming\npm\node_modules\karma>npm install
npm WARN package.json dateformat@0.2.1-2.3 No repository field.
npm WARN package.json istanbul@0.3.1-22 No repository field.
npm WARN package.json parsebeautify@0.1.0 No repository field.
npm GET https://registry.npmjs.org/mock-httpr-response
npm GET https://registry.npmjs.org/timer-shim
npm GET https://registry.npmjs.org/qq
npm GET https://registry.npmjs.org/which
npm GET https://registry.npmjs.org/sinon
npm GET https://registry.npmjs.org/sinon-chai
npm GET https://registry.npmjs.org/gaks
npm GET https://registry.npmjs.org/chai-as-promised
npm GET https://registry.npmjs.org/mocha
npm GET https://registry.npmjs.org/grunt-contrib-jshint
npm GET https://registry.npmjs.org/chai
npm GET https://registry.npmjs.org/jshint
npm GET https://registry.npmjs.org/qunit
npm 304 https://registry.npmjs.org/which
```

... and after stopping and starting the karma server:

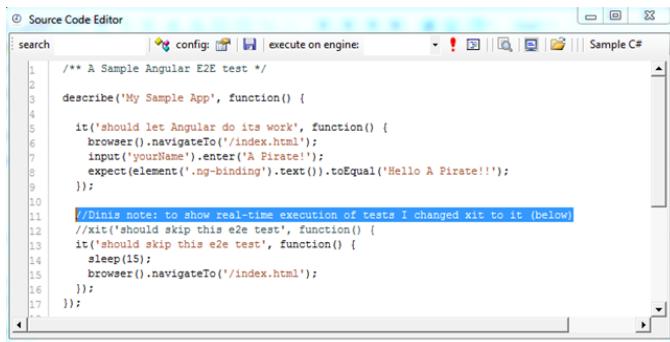
```
E:\_Code\_Tests\AngularJS\karma>karma start test\test\angular-scenario\karma.conf.js
[WARN] Karma: Port 9876 in use
[WARN] Karma: Port 9100 in use
[WARN] Karma: Port 9877 in use
[INFO] Karma: To run with this server, use "karma run --runner-port 9101"
[INFO] Karma: Node v0.9.8 (env) started at http://localhost:9878/_karma/
[INFO] Launcher: Starting browser Chrome
[INFO] Chrome 27.0.1453 (Windows 7): Connected on socket id PsBXb23HdnfMMUczNNVd
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS <0.447 secs / 0.208 secs>
```

all tests pass :)



## Changing and executing tests in real time

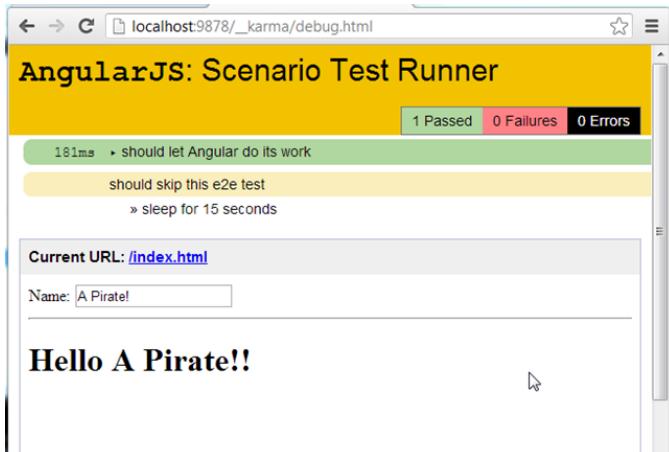
What is really cool with this set-up is that (as long as the karma process is running), because the **autoWatch** value (in `karma.conf.js`) was set to true, if I make a change to the test file:



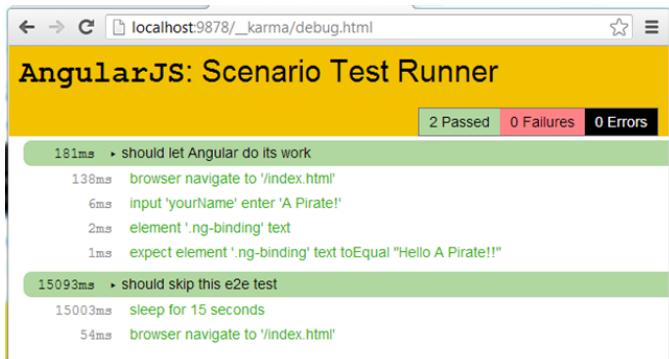
... the karma runner will detect the changes and rerun the tests (note that there are 2 tests executed now)

```
 karma@0.9.3 C:\Users\o2\AppData\Roaming\npm\node_modules\karma
 └── pause@0.0.1
   ├── d@0.0.1
   └── colors@0.6.0-1
     ├── graceful-fs@1.2.2
     └── inherits@0.1.4
   ├── chokidar@0.6.2
   └── mime@1.2.9
     ├── qs@0.7.0
     └── coffee-script@1.6.3
   ├── minimatch@0.2.12 (<signum@0.0.0, lru-cache@0.3.0)
   └── lodash@0.1.1
     ├── graceful-fs@1.2.2
     └── inherits@0.1.4
   ├── opt@0.1.2 (<inherits@0.1.0)
   └── useragent@0.6.6 (<lru-cache@2.2.4)
   ├── log4js@0.6.6 (<dequeue@0.3.0, server@0.1.4, async@0.1.15, readable-stream@0.1.0)
   ├── http-proxy@0.10.2 (<pjson@0.2.3, util@0.1.2)
   └── socket.io@0.9.16 (<base64@0.1.0, policyfile@0.0.4, redis@0.7.3, socket.io-client@0.9.16)
```

This 2nd test shows an interesting behaviors since it will make the test wait for 15 seconds (with the browser window opened):



Note how the time execution time for the 2nd test was ~15 secs



And if we modify the 2nd test to:

```

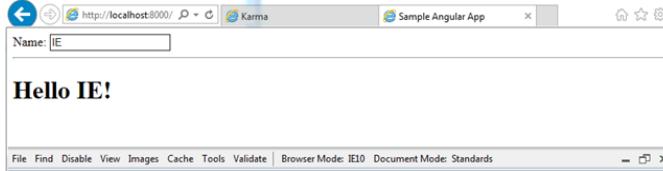
13
14
15
16
17
18
19
20
21
  });
});
});
```

... the execution test UI will look like this (note that the execution was triggered when I saved the test file :)

This screenshot shows a dual-pane view. On the left is a "Source Code Editor" showing the test code with a modified sleep duration. On the right is the "Karma DEBUG RUNNER" window showing the test results. The test "should skip this e2e test" now takes 10 seconds instead of 15. The test output shows the steps: navigating to index.html (~7ms), entering "Waiting 10 seconds" (~5ms), sleeping (~5ms), and finally displaying "Hello Waiting 10 seconds!".

\*\*NOTE 1: \*\*to solve the ENOENT error shown the first screenshot of localhost:8000, we just needed to a

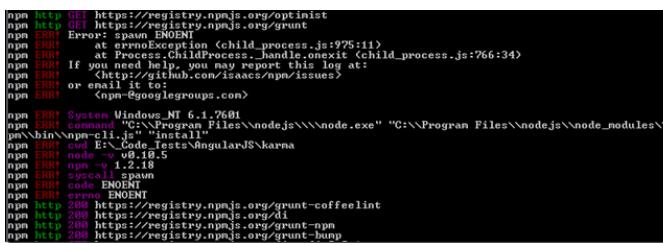
favicon.ico file to the \*\*lib/nodeserver \*\*folder



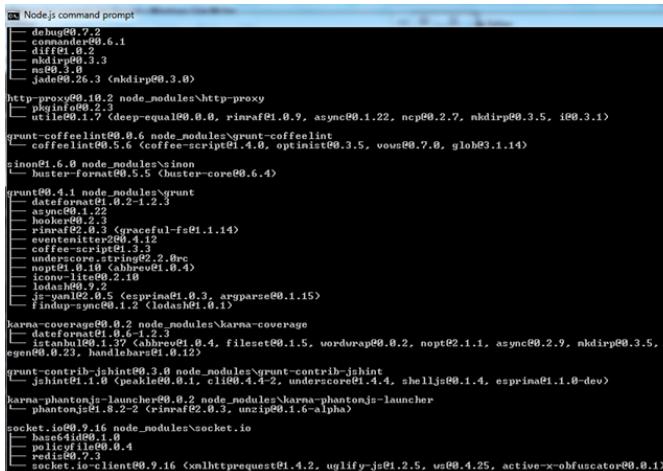
... and now the error doesn't happen anymore



Note 2: \*\*when trying to run Karma for the first time, I had a prob with grunt where it was failing with an \*\*Error: spawn ENOENT:



... this was resolved by installed the 32bit version of nodeJS and running **npm install** on the karma folder (after cloning it)




---

Table of Contents | [Code](#)

# 3 Firebase

This section has the following chapters:

- First PoC of sending TeamMentor's server-side request URLs to Firebase
  - Trying out Firebase (Beta) hosting solution and good example of Firebase Security rules
- 

[Table of Contents](#) | [Code](#)

### 3.1 First PoC of sending TeamMentor's server-side request URLs to Firebase (and seeing it in realtime in an AngularJS page)

After getting my head around how Firebase works (see [Using Firebase to sync data with a webpage \(via Javascript, REST and Firebase Admin panel\)](#) and [Trying our Firebase \(Beta\) hosting solution and good example of Firebase Security rules](#)), I really wanted to see how it could work on a key feature that I've been wanting to add to TeamMentor for ages: Realtime \*\*\*\*viewing of \*\*\*\*traffic and logs

And it worked :)

This is really exciting!!! (can you tell :) ), specially since I can see so many great uses of this type of technique and technology in TeamMentor (for example it will allow for much better understanding on how the content is used, and for better collaboration between its readers (and authors))

I'm going to write a number of blog posts that explain how this works in detail, but the core of it is the C# code shown below ([gist here](#)) which is running on a test TeamMentor instance, and basically hooks into the ASP.net HTTP pipeline in order to send the current URL to Firebase (using [Firebase's REST API](#)):

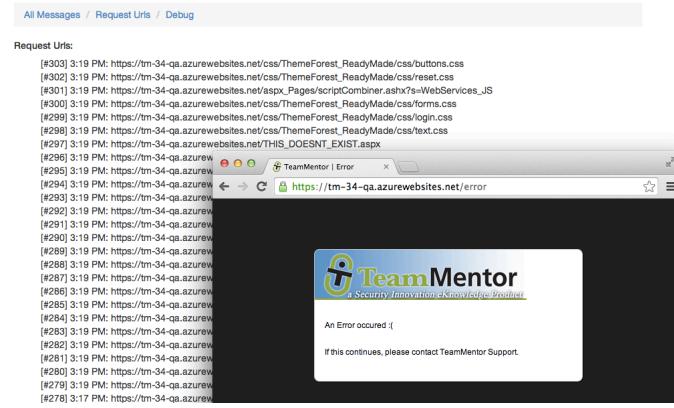
Here is the AngularJS+Firebase Html app that I created in Eclipse which shows the data received from the TeamMentor website (i.e. all requests made by a client visiting its home page):

ID	Time	URL
[#278]	3:17 PM	https://tm-34-qa.azurewebsites.net/HTML_Pages/Gui/TM_3_with_Panels.htm?time=1393514278810
[#277]	3:17 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/RBAC_CurrentPrincipal_Roles
[#276]	3:17 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/Current_User
[#275]	3:17 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/JsTreeWithFolders
[#274]	3:17 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/Current_User
[#273]	3:17 PM	https://tm-34-qa.azurewev
[#272]	3:17 PM	https://tm-34-qa.azurewev
[#271]	3:17 PM	https://tm-34-qa.azurewev
[#270]	3:17 PM	https://tm-34-qa.azurewev
[#269]	3:17 PM	https://tm-34-qa.azurewev
[#268]	3:17 PM	https://tm-34-qa.azurewev
[#267]	3:17 PM	https://tm-34-qa.azurewev
[#266]	3:17 PM	https://tm-34-qa.azurewev
[#265]	3:17 PM	https://tm-34-qa.azurewev
[#264]	3:17 PM	https://tm-34-qa.azurewev
[#263]	3:17 PM	https://tm-34-qa.azurewev
[#262]	3:17 PM	https://tm-34-qa.azurewev

... here are the logs for a page that doesn't exist:

ID	Time	URL
[#296]	3:19 PM	https://tm-34-qa.azurewebsites.net/THIS_DOESNT_EXIST
[#295]	3:19 PM	https://tm-34-qa.azurewebsites.net/THIS_DOESNT_EXIST
[#294]	3:19 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/GetGuidanceItemHtml
[#293]	3:19 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/JsTreeWithFolders
[#292]	3:19 PM	https://tm-34-qa.azurewebsites.net/Aspx_Pages/TM_WebServices.asmx/Current_User
[#291]	3:19 PM	https://tm-34-qa.azurewev
[#290]	3:19 PM	https://tm-34-qa.azurewev
[#289]	3:19 PM	https://tm-34-qa.azurewev
[#288]	3:19 PM	https://tm-34-qa.azurewev
[#287]	3:19 PM	https://tm-34-qa.azurewev
[#286]	3:19 PM	https://tm-34-qa.azurewev

... here are the requests for the TeamMentor error page:



### Quick look at the C# code executed on the TeamMentor server \*\*([gist here](#)):\*\*

Here is how the HTTP pipeline is hooked (using the TMEvents helper object from TM)

```

30 if (TMEvents.OnApplication_BeginRequest.size() > 1)
31 TMEvents.OnApplication_BeginRequest.remove();
32
33 logDebugMsg("mapping TMEvents.OnApplication_BeginRequest to send url to Firebase");
34
35 TMEvents.OnApplication_BeginRequest.Add(
36     ()=>
37         logRequestURL(HttpContext.Current.Request.Url.str());
38 );
39 return TMEvents.OnApplication_BeginRequest.size();
40
41 //using System.Threading;
42 //using TeamMentor.CoreLib;
43 //O2Ref:TeamMentor.CoreLib.dll

```

... the *logDebugMsg* and *logRequestURL* lambda functions are used to set the Firebase object to store the received messages:

```

20 Action<string> logDebugMsg =
21     (message)>>{
22         sendData("debugMsg", message);
23     };
24
25 Action<string> logRequestURL =
26     (url)>>{
27         sendData("requestUrl", url);
28     };

```

... the *sendData* lambda function is used to configure the Firebase target app and authorisation token:

```

13 Action<string,string> sendData =
14     (area, data) => {
15         var app           = "tm-admin-test";
16         var authToken   = "1luXuQ0tpzhrnrg2LrV1DNU17tOBu0peTqR6hNhFzm";
17         sendData_Raw(app, authToken, area, data);
18     };

```

... the **\*\*sendData\_Raw\*\*** lambda function is the one that sends the REST/POST request to the mapped Firebase app:

```

1 Action<string,string,string> sendData_Raw =
2     (app,token,area,data)>>{
3         ThreadPool.QueueUserWorkItem((o)=>
4             {
5                 var url      = "https://(0).firebaseio.com/(1).json?auth=(2)".format(app, area, token);
6                 var now      = Datetime.Now.ToString("yyyy-MM-ddTHH:mm:ss.fffZ");
7                 var postData = ("\"(0)\":{(1)}").format(now, data.replace("\\", "\\"));
8                 url.POST(postData);
9             });
10    };
11

```

Let me know if you have any questions regarding this example.

### Firebase recovers connection

In terms of being able to recover from going offline, Firebase seems to do a good job at reconnecting back to the server once the host box is back online (see image below which happened between me leaving my house and connecting into my current WIFI location)



The screenshot shows the Network tab of the Chrome DevTools developer tools. It lists several network requests and responses:

- XHR finished loading: "http://localhost:8080/FirebaseAngularJS/Poc\_FirebaseTeamMentor/Data/directives/topMenu.html". angular.js:8801
- XHR finished loading: "http://localhost:8080/FirebaseAngularJS/Poc\_FirebaseTeamMentor/Data/views/requestUrl.html". angular.js:8801
- event listener for 'load' event on document: use the 'document.addEventListener()' method.
- XHR finished loading: "http://localhost:8080/FirebaseAngularJS/Poc\_FirebaseTeamMentor/Data/views/messages.html". angular.js:8801
- ⚠ WebSocket connection to 'ws://s-dal5-nss-25.firebaseio.com/.ws?v=5\$5n\$tm-admin-test' failed: WebSocket is closed before the connection is established. firebase.js:153
- ⚠ WebSocket connection to 'ws://t-m-admin-test.firebaseio.com/.ws?v=5' failed: WebSocket is closed before the connection is established. firebase.js:153

[Table of Contents](#) | [Code](#)

## 3.2 Trying out Firebase (Beta) hosting solution and good example of Firebase Security rules

Since Firebase now offers a [Beta hosting service](#) (and I was looking for a quick way to publish one of the firebase PoCs I'm working at the moment), I decided to take it for a spin.

I have to say that I'm really impressed with the end result, and as you will see below, there entire process (from install to published website) was really smooth.

Note 1: in the example below I already had created an Firebase app to hold the data (see [Using Firebase to sync data with a webpage \(via Javascript, REST and Firebase Admin panel\)](#) for details on how to create one)

Note 2: at the time I wrote this post, the website created is/was (depending on when you are reading this) hosted at <https://tm-admin-test.firebaseio.com/>

Starting with the instructions from [Firebase hosting page](#):

---

\*\*

\*\*

**Quickstart**

- Web
- Node.js
- iOS / OS X
- Java / Android
- REST
- Firebase Hosting**
- Resources**
- Firebase Tutorial
- Downloads
- Examples
- Integrations

**Firebase Hosting (Beta)**

Firebase Hosting is a place for all your Firebase-powered app's static files such as HTML, CSS, JavaScript and image files on your own subdomain of firebaseapp.com: <yourapp>.firebaseapp.com

Note that Firebase Hosting is in beta, we'd appreciate your feedback! Please email [Chris Raynor](mailto:Chris Raynor) with comments and suggestions.

In order to access Firebase Hosting you need to use the Firebase Command Line Tools. First you need to have [Node.js](#) and [npm](#) installed, then run:

```
$ npm install -g firebase-tools
```

This will install the globally available `firebase` command. Use `firebase --help` for a full list of commands and `firebase <command> --help` for more detailed information.

### 1) Install the firebase tools

```
*Czencs-MacBook-Air:FireBase AngularJS plugin$ sudo npm install -g firebase-tools
npm http 200 https://registry.npmjs.org/firebase-tools
npm http 200 https://registry.npmjs.org/firebase-tools
npm http 200 https://registry.npmjs.org/bitalloons
npm http 200 https://registry.npmjs.org/prompt
npm http 200 https://registry.npmjs.org/tar
npm http 200 https://registry.npmjs.org/open
npm http 200 https://registry.npmjs.org/optimist
npm http 200 https://registry.npmjs.org/tar
npm http 200 https://registry.npmjs.org/tar/-/tar-0.1.19.tgz
npm http 200 https://registry.npmjs.org/tar/-/tar-0.1.19.tgz
```

```
.....
npm http 200 https://registry.npmjs.org/request/-/request-2.9.203.tgz
npm http 200 https://registry.npmjs.org/eyes/-/eyes-0.1.8.tgz
npm http 200 https://registry.npmjs.org/stack-trace/-/stack-trace-0.0.9.tgz
npm http 200 https://registry.npmjs.org/async
npm http 200 https://registry.npmjs.org/request/-/request-2.9.203.tgz
/usr/bin/firebase -> /usr/lib/node_modules/firebase-tools/bin/firebase
firebase-tools@0.1.6 /usr/lib/node_modules/firebase-tools
  └── open@0.0.4
    └── optimist@0.6.1 (wordwrap@0.0.2, minimist@0.0.8)
      ├── tar@0.1.19 (inherits@2.0.1, block-stream@0.0.7, fstream@0.1.25)
      └── bitalloons@0.1.2 (base64-js@0.0.6, glob@3.2.9)
        └── prompt@0.2.12 (revalidator@0.1.6, pkginfo@0.3.0, util@0.2.1, read@1.0.5, winston@0.6.2)
```

\*\*

\*\*

### 2) Run the firebase BootStrap

```
zens-MacBook-Air:FireBase_AngularJS plugin$ firebase bootstrap
Login required
Email: dinis.cruz@owasp.org
Password:
```

... after logging in (above), we are asked to chose the 'firebase' app to use (below)

```
----- YOUR FIREBases -----
incandescent-fire-1320
teammentor
tm-admin-test
xss
-----
Firebase: tm-admin-test
```

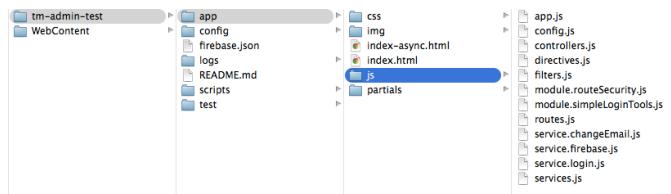
... then pick the template to use (I chose the *angular* one)

```
----- AVAILABLE TEMPLATES -----
angular
chat
-----
Template: angular
```

... and that's it:

```
Bootstrapping into directory 'tm-admin-test'
Downloading and unpacking template
Writing firebase.json settings file
BOOTSTRAPPING SUCCESSFUL - Instructions and setup guide here: https://github.com/firebase/angularFire-seed
zens-MacBook-Air:FireBase_AngularJS plugin$
```

Here are the files created locally (all under the *tm-admin-test* folder)



### 3) Deploy the application

```
zens-MacBook-Air:tm-admin-test plugin$ firebase deploy
Updating security rules...
Deploying...
DEPLOYED SUCCESSFULLY - View your app at: https://tm-admin-test.firebaseioapp.com
zens-MacBook-Air:tm-admin-test plugin$
```

... which is also super quick.

After completion the default browser is opened with the created website, which looked like this:

### Home

Not logged in

Open [Forge](#) or a [new window](#) to see this input update in real time:

AngularFire-seed v0.6

### 4) Testing the test app

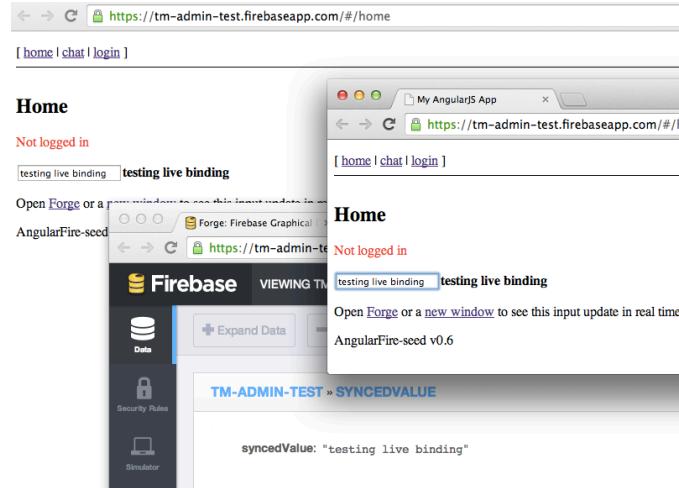
\*\*

The created/published test app has a simple **Firebase 3-way data binding** example, where changes in an **angular-binded model**, are propagated to the server, and distributed to any connected clients (the image below shows two browsers and the firebase admin panel, all synced in real time (regardless where the data is

changed))

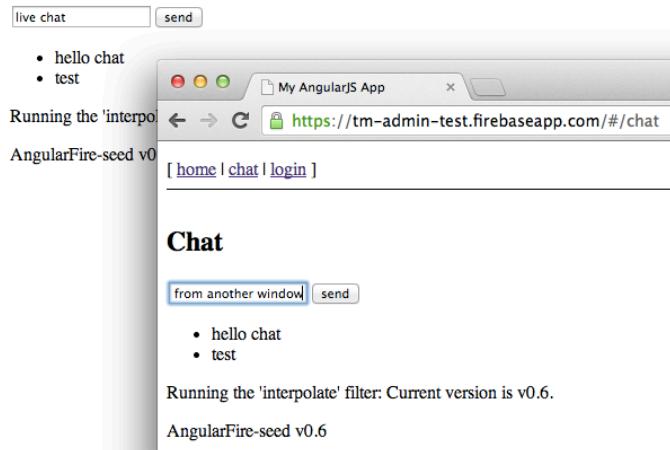
\*\*

\*\*



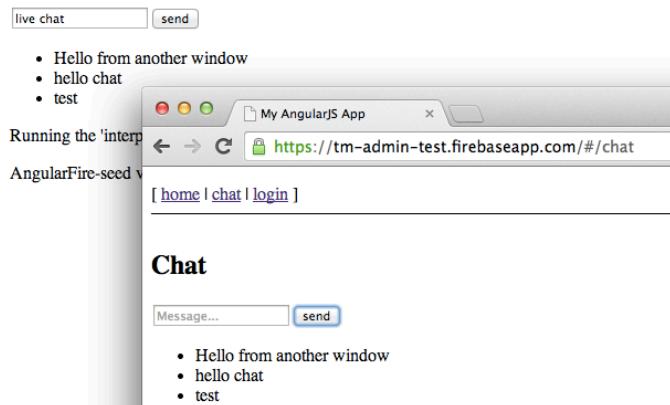
There is a basic \*\*Chat \*\*page:

## Chat



... which allows new messages to be added and viewed in real time:

## Chat



Also included is a Login page (with account creation capabilities):



## Login Page

email

password

AngularFire-seed v0.6

## 5) Review the security rules

At this stage it's good to take a look at the security rules added to this Firebase app.

In the image below we can see that:

- Anonymous users **can read** all data, but **not write** by default
- The **syncedValue** object **can be read and written** by all
  - there is a validation on this field that ensures that it is not empty and not bigger than 100 chars
- The **messages** object (i.e. 'firebase section/subdomain'):
  - can be read by all
  - each message must include a property called `_text` with a max length of 1000 chars
  - (I think) the `.validate = false` means that no other properties can be added
- the **users** object:
  - each **\$user** child object:
    - \* can be read if the current user matches its name (i.e. it is logged in as that user)
    - \* each user can write into two fields: **name** and **email** (both with a max length of 2000)

```

FIREBASE RULES
1  {
2    "rules": {
3      ".read": false,
4      ".write": false,
5      "syncedValue": {
6        ".read": true,
7        ".write": true,
8        ".validate": "newData.isString() && newData.val().length <= 100"
9      },
10     "messages": {
11       ".read": true,
12       "sendMessage": {
13         ".write": true,
14         ".validate": "newData.hasChildren(['text'])",
15         "text": {
16           ".validate": "newData.isString() && newData.val().length <= 1000"
17         },
18         "sother": {
19           ".validate": false
20         }
21       }
22     },
23     "users": {
24       "$user": {
25         ".read": "auth.uid === $user",
26         ".write": "auth.uid === $user && (!newData.exists() || newData.hasChildren())",
27         "name": {
28           ".validate": "newData.isString() && newData.val().length <= 2000"
29         },
30         "email": {
31           ".validate": "newData.isString() && newData.val().length <= 2000"
32         },
33         "sother": {
34           ".validate": false
35         }
36       }
37     }
38   }
39 }
```

## 6) Create an account

To test the provided authorisation solution, let's start by trying to login with an account that doesn't exist:

**email**

**password**

Error: The specified user does not exist.

AngularFire-seed v0.6

... then click on the **Register** button (which just adds a **confirm pass** field)

### Login Page

**email**

**password**

**confirm pass**

Here is what the 'post register/login' page looks like (note that the top-menu-bar **login** link was also changed to **account**)

[ [home](#) | [chat](#) | [account](#) ]

## Account

The screenshot shows the 'Account' page with three sub-forms:

- Profile:** Fields for Name (A) and Email (a@a.com).
- Change Password:** Fields for Old Password, New Password, Confirm Password, and a 'update password' button.
- Change Email:** Fields for New Email, Password, and a 'update email' button.

Top right: Log Out, AngularFire-seed v0.6

Clicking on *home* takes us to the first page, which also shows a different message (now that we are logged in)

[ [home](#) | [chat](#) | [account](#) ]

## Home

Hello! You are logged in.

[testing live binding](#) | [testing live binding](#)

Open [Forge](#) or a [new window](#) to see this input update in real time:

AngularFire-seed v0.6

Interestingly the *Chat* page doesn't seem to take into account that we are logged in now (would be nice to show the current user in the message posted)

## Chat

Message...

- after login
- Hello from another window
- hello chat
- test

Running the 'interpolate' filter: Current version is v0.6.

AngularFire-seed v0.6

## 6) Take a look at data (as stored in the assigned Firebase app)

\*\*

\*\*Using the control panel for the current app:

a) here is the *syncedValue* object

TM-ADMIN-TEST > SYNCEDVALUE

syncedValue: "aaaaaaaa"	Legend
	Changed Added Deleted Moved

b) here is the *messages* object (which is a firebase kind-of-array, based on a name/value pair. The name is the *node* text (for example \_JGoCMZRRGo1WQHmYxmc) and the value is the child \_text node value (for example \_test\_):

TM-ADMIN-TEST > MESSAGES

```

messages
  -JGoCMZRQ8o1WGQHmYxm
    text: "test"
  -JGoDgvG0H2zaBcajSac
  -JGoDnCPx_FZDre_Gkj + x
  -JGoERJwIX_5e3x-4EE
    text: "after login"

```

Legend  
■ Changed  
■ Added  
■ Deleted  
■ Moved

c) There was a new **users** node/object in the root of the app's data:

TM-ADMIN-TEST

```

tm-admin-test
  -JGgRoMbs20fNW_hRxhk
  -JGhdYj5J00cpp_kh1LR
  -JGheXxa041fGjNEJSG9
  -JGhiWt9wQZqJ-tKv28N
  -JGiELu4urkW7E7NIh84
  -JGiEMjHDqjAPBFqiEA4
  -JGjGdLaSu_Z--ifVvjM
  debugMsg
  messages
  requestUrl
  syncedValue: "aaaaaaaa"
  users
  simplelogin:2

```

Legend  
■ Changed  
■ Added  
■ Deleted  
■ Moved

... which contains the user provided (and ‘editable by that user’) data (*email* and *name*)

TM-ADMIN-TEST > USERS

```

users
  simplelogin:2
    email: "a@a.com"
    name: "A"

```

Legend  
■ Changed  
■ Added  
■ Deleted  
■ Moved

And where is the user’s credentials?

The user created was added to the current list of ‘registered users’ , and it must be stored somewhere inside firebase servers since we don’t have access to it (hopefully it is stored using a nice strong hashing algorithm)

Registered Users

User ID	Email
1	dinis.cruz@owasp.org
2	a@a.com

Analytics  
Simple Login +  
Hosting  
Secrets

## Wrapping up

\*\*

\*\*Kudos to the Firebase team for providing such easy-to-use hosting solution (next step for me is to try to use it in the TeamMentor PoC I’m currently working on)

\*\*

\*\*For reference the AngularJS+Firebase website that was created by the Firebase cli tool, is the one available at <https://github.com/firebase/angularFire-seed> (which contains a nice \*\*README.md \*\*file with more details)

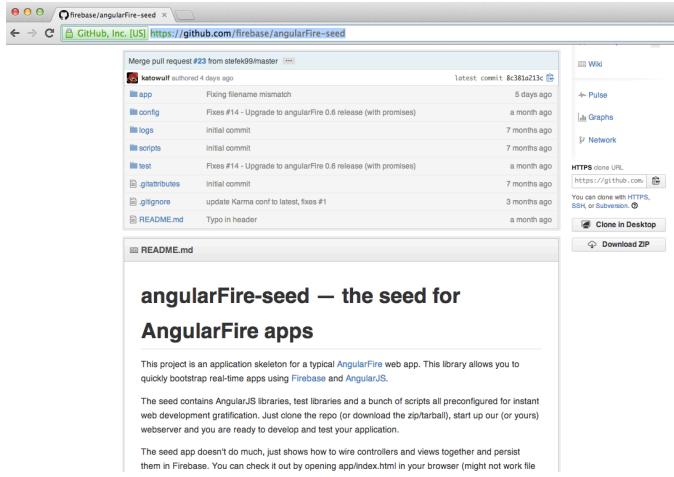


Table of Contents | [Code](#)

## 4 Misc Tricks

This section has the following chapters:

- [Programmatically changing an AngularJS scope variable and adding Firebug Lite to an AngularJs app](#)
  - [Hubspot current.js code includes JQuery on it](#)
  - [Submitting TM users to HubSpot via TBOT interface \(using Angular JS\)](#)
- 

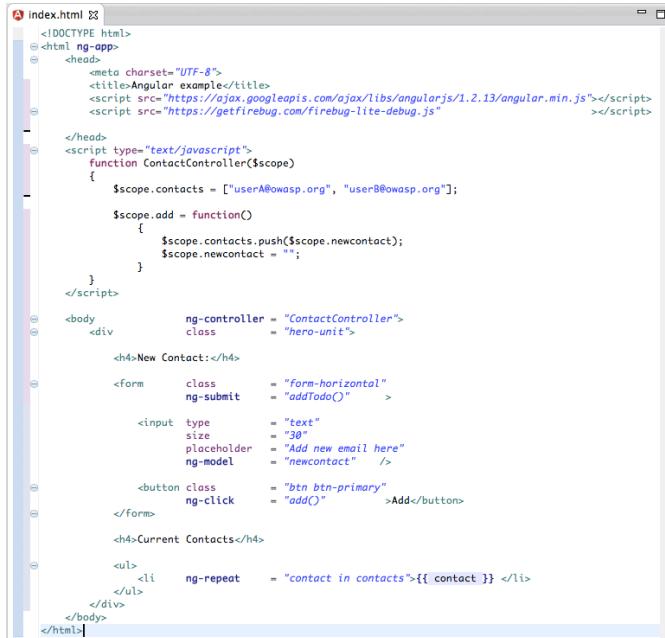
[Table of Contents](#) | [Code](#)

## 4.1 Programatically changing an AngularJS scope variable and adding Firebug Lite to an AngularJs app

In this post I'm going to show two really nice tricks that help when developing AngularJS applications:

- adding [Firebug Lite](#) to the current browser
- changing the scope value outside a normal AngularJS controller, service or module

Let's say that we are inside Eclipse and have this simple AngularJS app ([gist here](#))

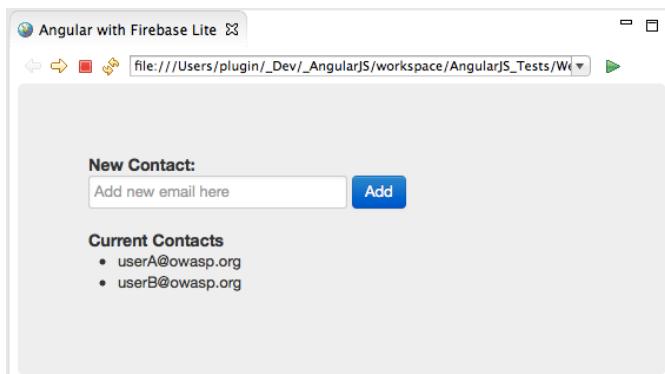


```

<!DOCTYPE html>
<html ng-app>
  <head>
    <meta charset="UTF-8">
    <title>Angular example</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
    <script src="https://getfirebug.com/firebug-lite-debug.js"></script>
  </head>
  <script type="text/javascript">
    function ContactController($scope)
    {
      $scope.contacts = ["userA@owasp.org", "userB@owasp.org"];
      $scope.add = function()
      {
        $scope.contacts.push($scope.newcontact);
        $scope.newcontact = "";
      }
    }
  </script>
  <body>
    <div ng-controller="ContactController" class="hero-unit">
      <h4>New Contact:</h4>
      <form class="form-horizontal" ng-submit="addTodo()">
        <input type="text" size="30" placeholder="Add new email here" ng-model="newcontact" />
        <button class="btn btn-primary" ng-click="add()>Add</button>
      </form>
      <h4>Current Contacts:</h4>
      <ul>
        <li ng-repeat="contact in contacts">{{ contact }}</li>
      </ul>
    </div>
  </body>
</html>

```

... which looks like this when executed:



To add Firebug Lite to this page, all we need to do is to add a script reference to <https://getfirebug.com/firebug-lite-debug.js>

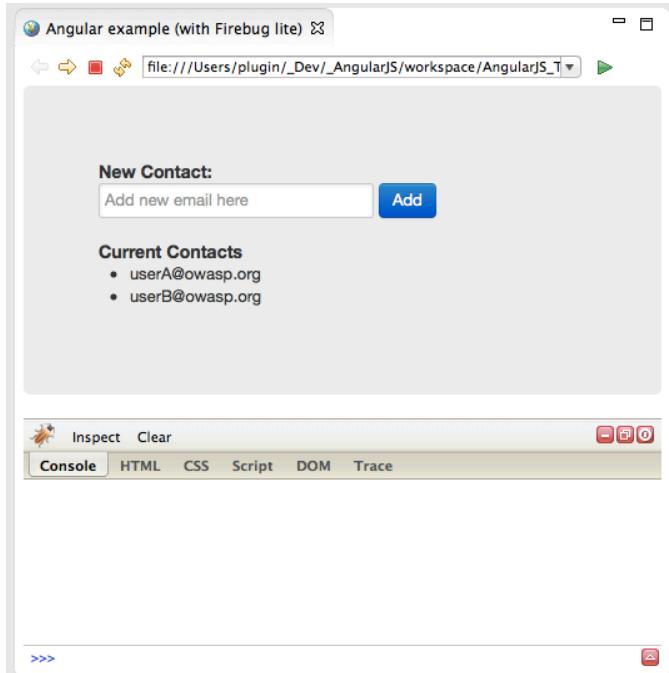


```

<!DOCTYPE html>
<html ng-app>
<head>
  <meta charset="UTF-8">
  <title>Angular example (with Firebug lite)</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
  <script src="https://getfirebug.com/firebug-lite-debug.js"></script>
  <link href="http://angularjs.org/css/bootstrap.min.css" rel="stylesheet" type="text/css">
</head>
<body>
  <script type="text/javascript">
    function ContactController($scope) {
      $scope.contacts = ["userA@owasp.org", "userB@owasp.org"];
      $scope.add = function() {
        $scope.contacts.push($scope.newcontact);
        $scope.newcontact = "";
      }
    }
  </script>

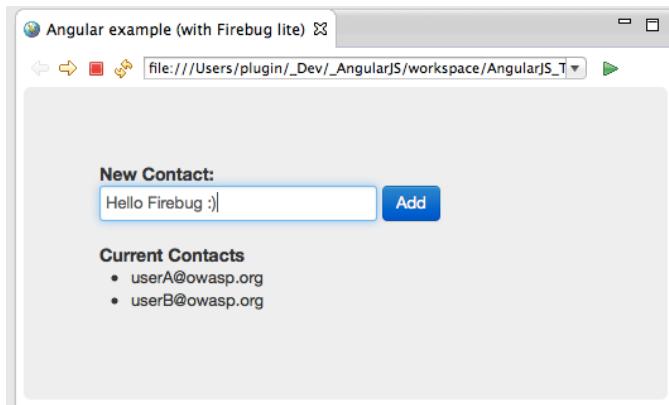
```

... and after refresh we will have a nice Firebug Lite console (and other nice goodies) at the bottom of our page :)



Next lets see how to access and change the AngularJS `$scope` of the `ContactController`.

The objective is to access programmatically the `New Contact` value (set below to `Hello Firebug :)`)



In Firebug Lite, we can access the scope object by executing: `var scope = angular.element(document.body).scope()`

```

Inspect Clear
Console HTML CSS Script DOM Trace
>>> var scope = angular.element(document.body).scope()
undefined
>>> scope
Object { $id="003", this=Object, $$listeners=Object, ... }

```

... and the New Contact \*\*value value using: \*\*`scope.newcontact`

```

Inspect Clear
Console HTML CSS Script DOM Trace
>>> var scope = angular.element(document.body).scope()
undefined
>>> scope
Object { $id="003", this=Object, $$listeners=Object, ... }
>>> scope.newcontact
"Hello Firebug :)"

```

If we change the **New Contact** value here (in Firebug Lite) using `scope.newcontact = "Hello AngularJS"`

```

Inspect Clear
Console HTML CSS Script DOM Trace
>>> var scope = angular.element(document.body).scope()
undefined
>>> scope
Object { $id="003", this=Object, $$listeners=Object, ... }
>>> scope.newcontact
"Hello Firebug :)"
>>> scope.newcontact = "Hello AngularJS"

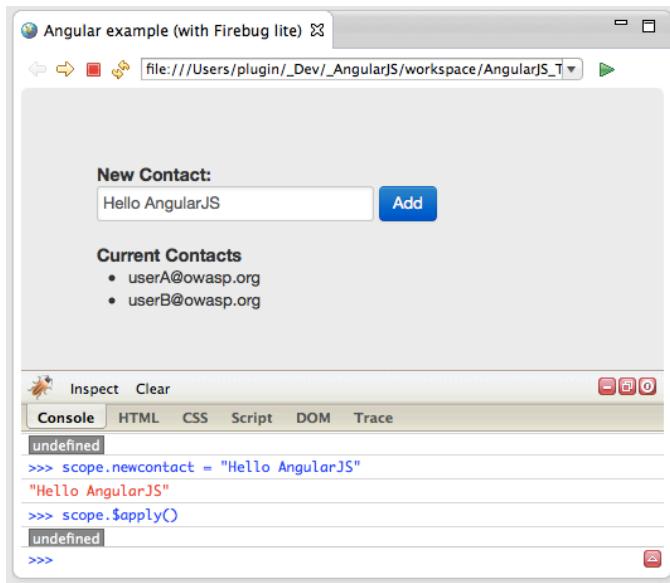
```

... we will notice that the value will not automagically (ala AngularJS way) change in the binded (via **ng-model**) input field

The screenshot shows a browser window with an AngularJS application. The page has two sections: 'New Contact:' with an input field containing 'Hello Firebug :)' and a button labeled 'Add'. Below it is a 'Current Contacts' section with a list of items: 'userA@owasp.org' and 'userB@owasp.org'.

The reason that didn't happen is because the change was done outside the AngularJS \$digest cycle.

The solution is to call the `scope.$apply()` function (and the input field will be updated):



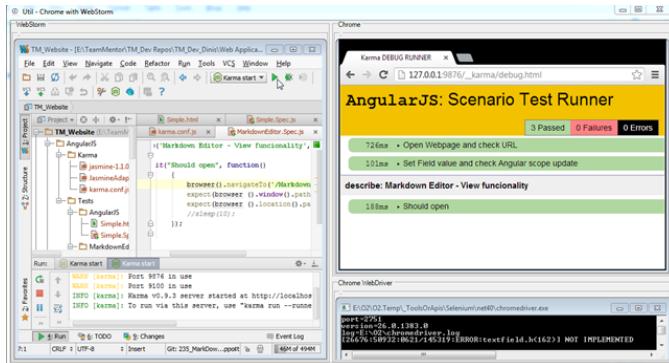
---

[Table of Contents](#) | [Code](#)

## 4.2 Hubspot current.js code includes JQuery on it

Although I'm using Angular.js on the HubSpot TBot page (see [Submitting TM users to HubSpot via TBOT interface \(using Angular JS\)](#)) I'm still more comfortable/knowledgeable in jQuery, so I decided to use it to populate the HubSpot fields.

So my first action was to load jQuery at the top of the TBot page:



which allowed me to do this:

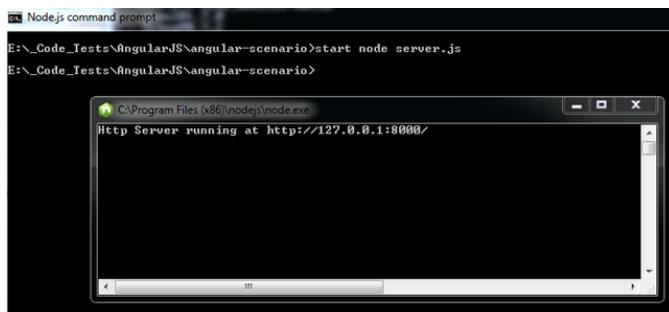
```

function TM_HubSpot($scope, $http)
{
    var loadData = function()
    {
        userName = window.location.search.substring(1);
        if (userName != "")
        {
            $scope.userName = userName;
            $http.get('/rest/user/' + userName).success(function(data)
            {
                _data = data;
                $scope.userData = data;
                $(".hs-input[name='firstname']").val(data.FirstName);
                $(".hs-input[name='lastname']").val(data.LastName);
                $(".hs-input[name='email']").val(data.Email);
                $(".hs-input[name='company']").val(data.Company);
                $(".hs-input[name='jobtitle']").val(data.Title);
                $(".hs-input[name='country']").val(data.Country);
                $(".hs-input[name='state']").val(data.State);
            });
        }
        else
        {
            $scope.userName = "No username provided";
        }
    }
}

```

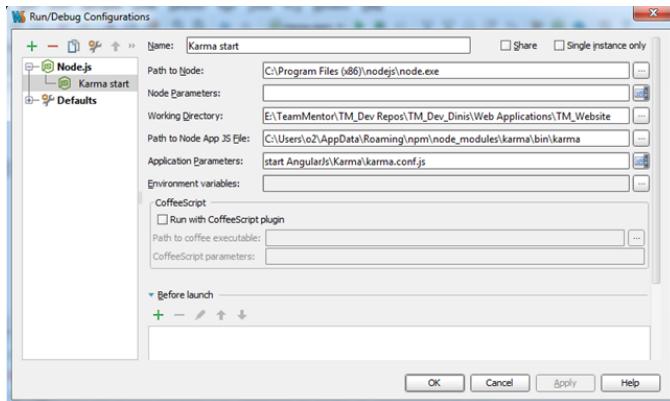
(note the mix of JQuery and AngularJS code)

But then as I looked through the <http://js.hubspot.com/forms/current.js> file (which you will also notice that for TM, I saved it on the `/Customizations` folder)



In the *Util - Javascript Format (and Beautify).h2* O2 Platform tool:

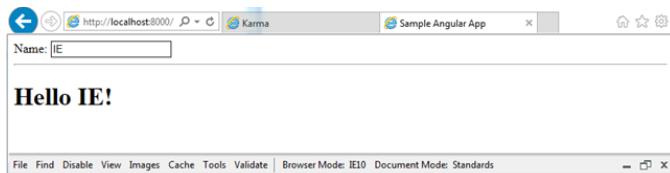
I noticed that they had embedded `jQuery` \*\*in the \*\*`_current.js` code, namely at the `window.hsjQuery` variable



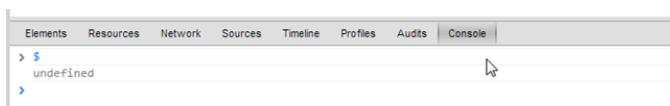
this means that we can use jQuery to access the page's DOM

```
E:\src\Code_Tests\Angular\angular-scenario> karma start karma.conf.js
INFO [HTTP]: Karma v0.9.3 server started at http://localhost:8888/karma/
INFO [Chrome 27.0.1453 (Windows 7)]: Connected on socket id 5MWWjMs6y0nMrJ-2w
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (0.366 secs / 0.184 secs)
```

Now, if we remove the jQuery script import:



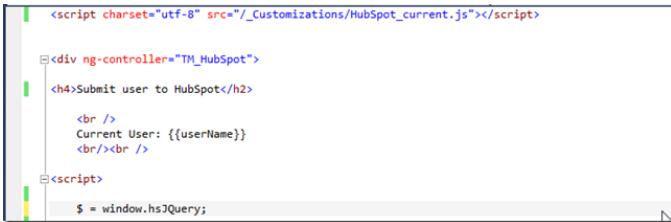
the \$ variable will not be available:



But if we assign `$` to `window.jQuery`, we are back in action:



Finally, we add that assignment to the TBot page



```
<script charset="utf-8" src="/_Customizations/_HubSpot_current.js"></script>

<div ng-controller="TM_HubSpot">
  <h4>Submit user to HubSpot</h4>
  <br />
  Current User: {{user_name}}
  <br /><br />
<script>
  $ = window.hjQuery;
```

And the previous code (that used the `$`) is working again (now using the JQuery version from `current.js`)

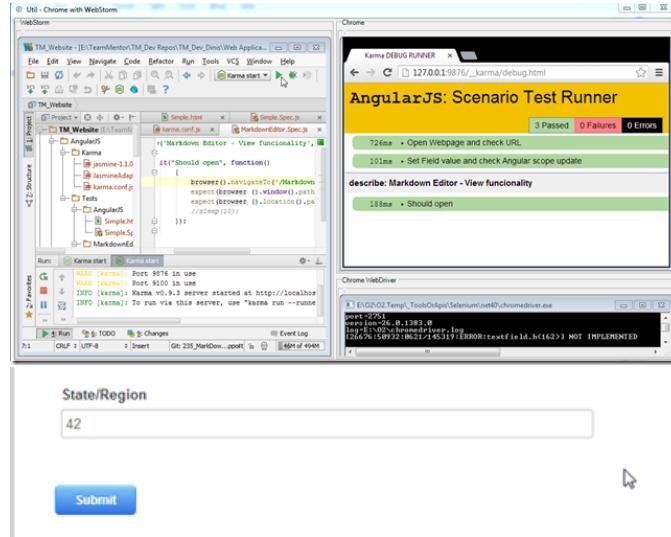
---

[Table of Contents](#) | [Code](#)

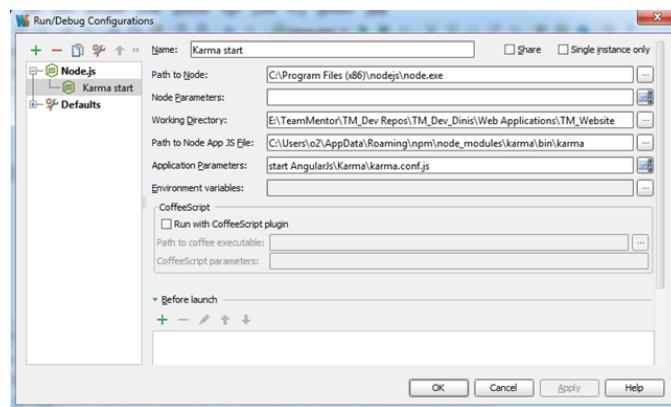
## 4.3 Submitting TM users to HubSpot via TBOT interface (using Angular JS)

Following the need to submit TM new users to HubSpot, I just created an TBot razor page (TM admin script) that uses Angular JS to get data about a particular user and populates a form that can then be submitted to HubSpot.

Here is what the Form looks like for the admin user (value provided on the url):

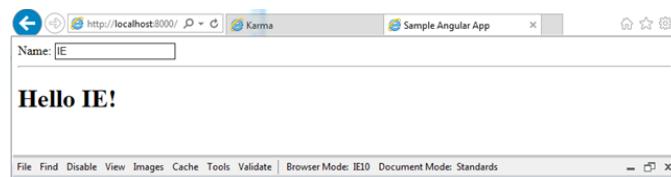


Pressing the Submit button will send the data to HubSpot which is captured like this:



This works using the new HubSpot form API and Structure (which is quite nice).

The form is created using this HubSpot provided script:



```
E:\...\Code\Tests\AngularJS\angular-scenario>karma start karma.conf.js
[INFO] [Karma]: Karma v0.9.3 server started at http://localhost:9876/_karma/
[INFO] [Chrome 27.0.1453 (Windows 7)]: Connected on socket id 5WJYzHyc60uMrZ_-2v
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (0.368 secs / 0.184 secs)
```

Which is then auto-populated using Angular.js

```
var loadData = function()
{
    var userName = window.location.search.substring(1);
    if (userName != "")
    {
        $scope.userName = userName;
        $http.get('/rest/user/' + userName).success(function(data)
        {
            _data = data;
            $scope.userData = data
            $(".hs-input[name='firstname']").val(data.FirstName);
            $(".hs-input[name='lastname']").val(data.LastName);
            $(".hs-input[name='email']").val(data.Email);
            $(".hs-input[name='company']").val(data.Company);
            $(".hs-input[name='jobtitle']").val(data.Title);
            $(".hs-input[name='country']").val(data.Country);
            $(".hs-input[name='state']").val(data.State);
        });
    }
    else
        $scope.userName = "No username provided";
}
```

---

[Table of Contents](#) | [Code](#)

# 5 IDEs

This section has the following chapters:

- [Eclipse Groovy REPL script to sync a Browser with file changes](#)
  - [Eclipse Groovy script to remove the ‘busy’ image from the WebBrowser Editor](#)
  - [Using Chrome inside a native VisualStudio pane \(using Window Handle Hijacking\)](#)
  - [Using WebStorm with Chrome and ChromeDriver \(to view KarmaJS execution results\)](#)
  - [When the best way to automate Chrome is to use ... Chrome](#)
  - [Adding KarmaJS support to WebStorm to automatically run tests on file changes](#)
- 

[Table of Contents](#) | [Code](#)

## 5.1 Eclipse Groovy REPL script to sync a Browser with file changes (with recursive folder search via Java's WatchService)

Since I am using Eclipse to develop using AngularJS (see [Creating an Eclipse UI to run AngularJS e2e tests using Karma](#)), I needed a way to refresh the browser window every-time I made changes to any AngularJS related file (note that due to the nature of the AngularJS projects, I need the change to trigger on any change made inside the root folder and all its subfolders).

Since there didn't seem to be an easy way to do this ('*auto browser refresh on file changes*') in Eclipse, I used the [Eclipse Groovy REPL Scripting Environment](#) to develop a script/macro that:

- Based on a title of an opened eclipse editor file:
- ... find the full path of that file, and:
- ... create a [Java WatchService](#) that monitors the file's folder and subfolders, and:
- ... when a StandardWatchEventKinds.ENTRY\_MODIFY is received :
  - Create/Open a new Eclipse view with a browser (called ***Synced Browser***), and:
  - ...refresh the index page

For reference here is the groovy code for this script ([gist here](#)):

Originally I had tried to use Eclipse file change events (like on this [SO thread](#)), but that didn't work as well as the WatchService.

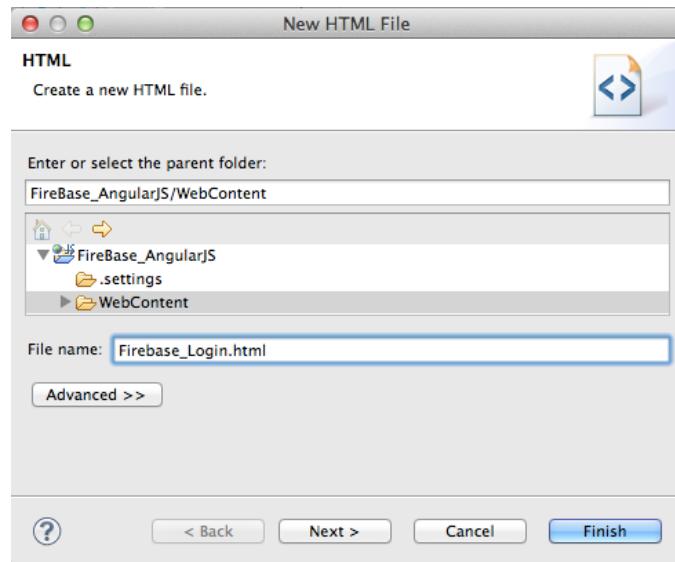
A next step is to create a mini UI to allow the configuration of the target files (maybe as a view added to the next version of the [Groovy REPL Eclipse](#) plugin)

### Seeing it in action

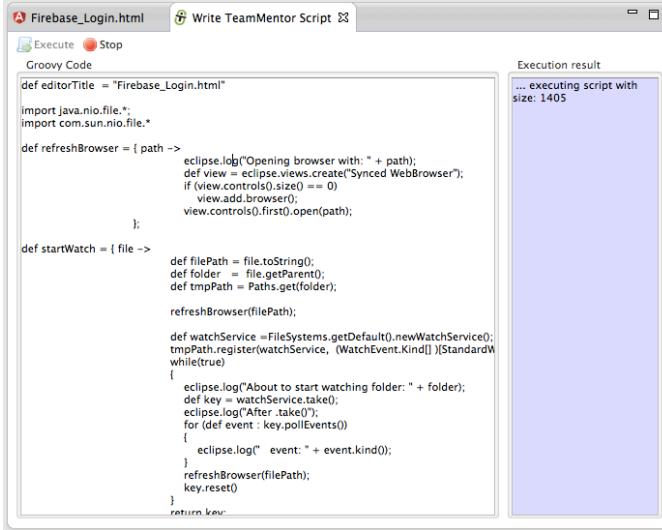
\*\*

\*\*Here is how to test this script:

1) create a Web project with an Html file on it:



2) run the Groovy code in the REPL window (note that the image below is using a different root file and the version of script is an [older one](#) (which didn't contain the recursive folder monitoring)):



The screenshot shows the Eclipse IDE interface. On the left, the 'Groovy Code' editor contains a Groovy script. On the right, the 'Execution result' view displays the output of the script's execution.

```

A Firebase_Login.html Write TeamMentor Script
Execute Stop
Groovy Code
def editorTitle = "Firebase_Login.html"
import java.nio.file.*;
import com.sun.nio.file.*;

def refreshBrowser = { path ->
    eclipse.log("Opening browser with: " + path);
    def view = eclipse.views.create("Synced WebBrowser");
    if (view.controls.size() == 0)
        view.add.browser();
    view.controls[0].open(path);
}

def startWatch = { file ->
    def filePath = file.toString();
    def folder = file.getParent();
    def tmpPath = Paths.get(folder);

    refreshBrowser(filePath);

    def watchService = FileSystems.getDefault().newWatchService();
    tmpPath.register(watchService, (WatchEvent.Kind[]) StandardWatchEventKinds.all());
    while(true)
    {
        eclipse.log("About to start watching folder: " + folder);
        def key = watchService.take();
        eclipse.log("After .take()");
        for (def event : key.pollEvents())
        {
            eclipse.log(" event: " + event.kind);
            refreshBrowser(filePath);
            key.reset();
        }
    }
    return key;
}

```

Execution result  
... executing script with size: 1405

... on execution you will see a new view (called *Synced WebBrowser*) show up in your current Eclipse instance.

3) make some changes on the Html file



The screenshot shows the Eclipse IDE interface. The 'HTML' editor tab is active, displaying the content of the 'Firebase\_Login.html' file.

```

<!doctype html>
@<html ng-app>
@<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-route.min.js"></script>
<link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
</head>
<body>
<h2>Firebase with Login (will be shown on Save)</h2>
<div ng-view></div>
</body>
</html>

```

4) and note that the *Synced WebBrowser* view will be refreshed automatically (it takes about 500ms to 1s for the change to be picked up (see this [SO answer](#) for why I had to use the *SensitivityWatchEventModifier.HIGH* setting on the WatchService))

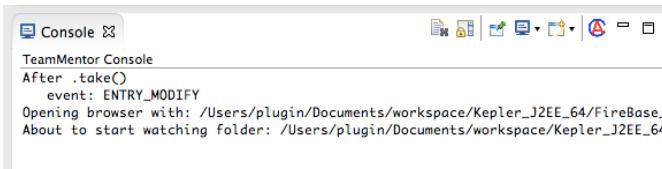


The screenshot shows the Eclipse IDE interface. The 'Synced WebBrowser' view is open, displaying the updated HTML content: '

## Firebase with Login (will be shown on Save)

'.

5) if you open the *TeamMentor Console*, you will also see a number of log messages that help to see what is going on:



The screenshot shows the Eclipse IDE interface. The 'Console' view is open, displaying log messages from the 'TeamMentor Console'.

```

Console
TeamMentor Console
After .take()
event: ENTRY_MODIFY
Opening browser with: /Users/plugin/Documents/workspace/Kepler_J2EE_64/FireBase
About to start watching folder: /Users/plugin/Documents/workspace/Kepler_J2EE_64

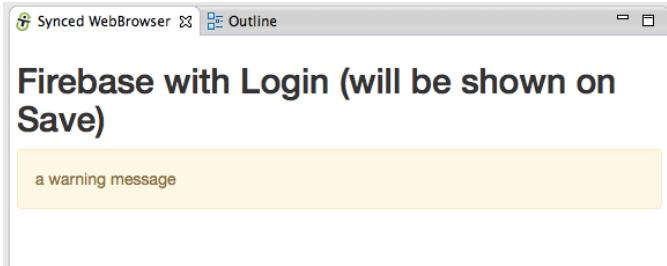
```

6) here is another example where I added a new [Bootstrap](#) css div:

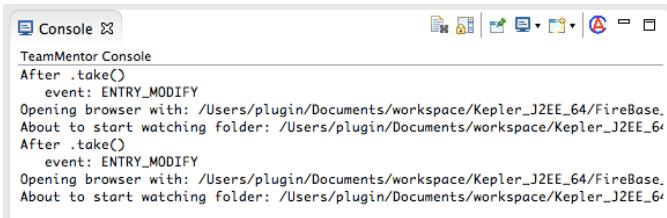


```
<!doctype html>
<html ng-app>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js">
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular-route.js">
  <link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
</head>
<body>
  <h2>Firebase with Login (will be shown on Save) </h2>
  <div class="alert alert-warning">a warning message</div>
  <div ng-view></div>
</body>
</html>
```

7) which was immediately (~500ms) shown on save



8) note that the log message shows the events being triggered and the resetting of the *WatcherService*:



```
Console
TeamMentor Console
After .take()
  event: ENTRY_MODIFY
Opening browser with: /Users/plugin/Documents/workspace/Kepler_J2EE_64/FireBase.
About to start watching folder: /Users/plugin/Documents/workspace/Kepler_J2EE_64/
After .take()
  event: ENTRY_MODIFY
Opening browser with: /Users/plugin/Documents/workspace/Kepler_J2EE_64/FireBase.
About to start watching folder: /Users/plugin/Documents/workspace/Kepler_J2EE_64/
```

---

[Table of Contents](#) | [Code](#)

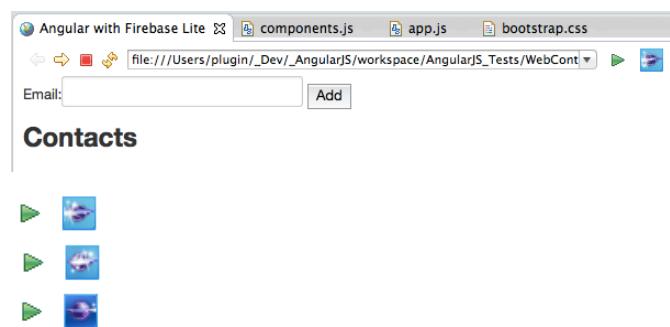
## 5.2 Eclipse Groovy script to remove the 'busy' image from the WebBrowser Editor

Now that I'm doing [AngularJS](#) and [Firebase](#) development inside Eclipse, there was a little 'thing' that was starting to drive me crazy: ***The animation icon on the top right of the Eclipse WebBrowser!***

\*\*

Apart from the [mosaic](#) 2000s look (which is kinda ok), there is a big problem with pages that keep running for a while: ***the animation doesn't stop!***

\_ This means that if you are developing inside Eclipse, there is this 'thing' (i.e. the top right icon) that keeps moving and demand my brain's attention:



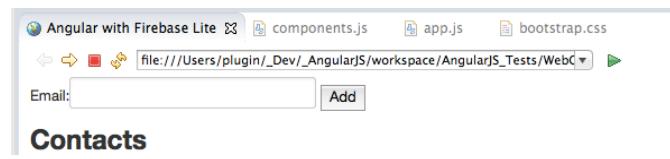
Since there didn't seem to be an preference or option to disable this behaviour, it was time to open up the [Eclipse Groovy REPL Scripting Environment](#) and fix it with a script :)

### The solution

After a quick prototyping, I come up with this script to remove all icons from all opened WebBrowser editors ([gist here](#)):

```
1. final script.groovy
1 def visible = false;
2
3 for(editor in eclipse.editors.list())
4     if (editor.id == "org.eclipse.ui.browser.editor")
5         editor.getEditor(true).webBrowser.busy.visible = visible;
```

After execution the icon is not there anymore :)



### How I found the solution

Here are the steps I took to find the `_busy` object to set the `visible` value to false

- 1) in the [Groovy REPL](#) I started by getting a list the `ids` of all current opened Eclipse views, but there didn't seem to be any web browser in there:

The screenshot shows the 'Write TeamMentor Script' dialog. In the 'Groovy Code' panel, the code 'return eclipse.views.ids()' is entered. In the 'Execution result' panel, a list of Eclipse view IDs is displayed:

```
org.eclipse.ui.navigator.ProjectExplorer
org.eclipse.ui.views.ContentOutline
org.eclipse.ui.views.PropertySheet
org.eclipse.wst.server.ui.ServersView
org.eclipse.wst.common.snippets.internal.ui.SnippetsView
org.eclipse.ui.views.ProblemView
org.eclipse.egit.ui.RepositoriesView
org.eclipse.team.ui.GenericHistoryView
g2.scripts.views.SimpleEditor
org.eclipse.pde.runtime.LogView
org.eclipse.egit.ui.StagingView
tm.eclipse.ui.views.Eclipse_Panel
```

2) then I looked at the list the *ids* of all current opened editors, and found one that sounded promising:  
**org.eclipse.ui.browser.editor**

The screenshot shows the 'Write TeamMentor Script' dialog. In the 'Groovy Code' panel, the code 'return eclipse.editors.ids()' is entered. In the 'Execution result' panel, a list of Eclipse editor IDs is displayed:

```
org.eclipse.angularjs.editor
org.eclipse.angularjs.editor
org.eclipse.ui.browser.editor
org.eclipse.wst.jsdt.ui.CompilationUnitEditor
org.eclipse.wst.jsdt.ui.CompilationUnitEditor
org.eclipse.wst.css.core.csssource.source
```

3) to confirm that that was the one I wanted, I looked at the *titles* of the current editors, and confirmed that the browser window I want to capture was there (the title was “Angular with Firebase Lite”);

The screenshot shows the 'Write TeamMentor Script' dialog. In the 'Groovy Code' panel, the code 'return eclipse.editors.titles()' is entered. In the 'Execution result' panel, a list of editor titles is displayed:

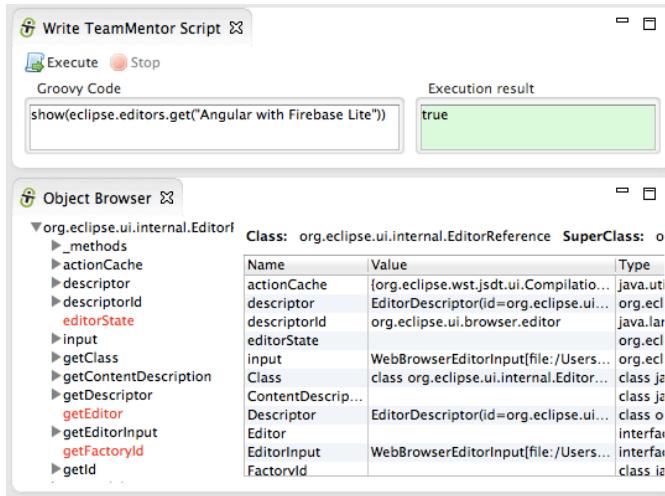
```
index.html
index.html
Angular with Firebase Lite
components.js
app.js
bootstrap.css
```

4) now that I knew the title, it was easy to get an *EditorReference* to it:

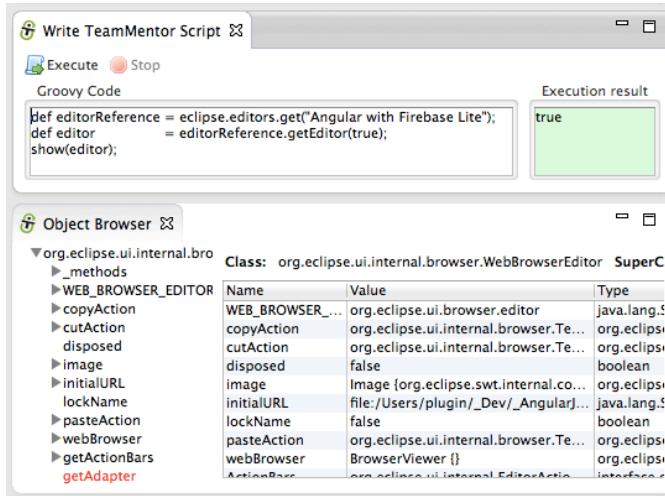
The screenshot shows the 'Write TeamMentor Script' dialog. In the 'Groovy Code' panel, the code 'return eclipse.editors.get("Angular with Firebase Lite");' is entered. In the 'Execution result' panel, the object reference is displayed:

```
org.eclipse.ui.internal.EditorReference@1b6f7c9
```

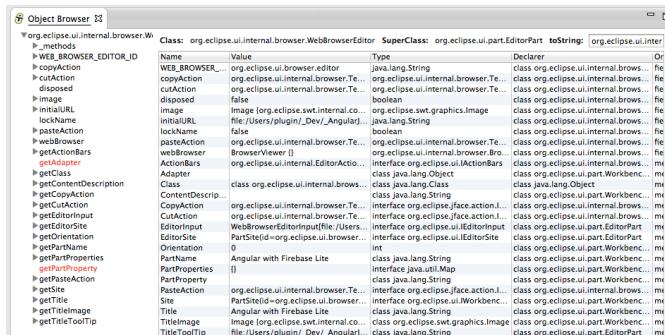
5) everytime I have an object reference that I want to take a look, I call the `_show({object})` `_viewer`, since that will give me a nice *TreeViewer* of all methods, fields and properties ( see [Viewing Eclipse and SWT objects \(Workbench, Display and Shell\)](#) using Groovy's ObjectBrowser and using TeamMentor's Plugin ObjectBrowser for more details how this works)



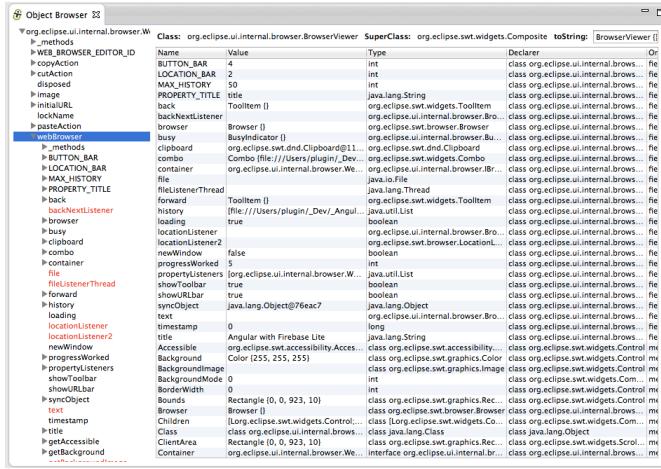
6) in this case we don't want the `_EditorReference` object. What we really want is the actually editor, which can be retrieved by invoking the `getEditor(true)` method (note how the object we are currently seeing in the `Object Browser` is the `org.eclipse.ui.internal.WebBrowserEditor`)



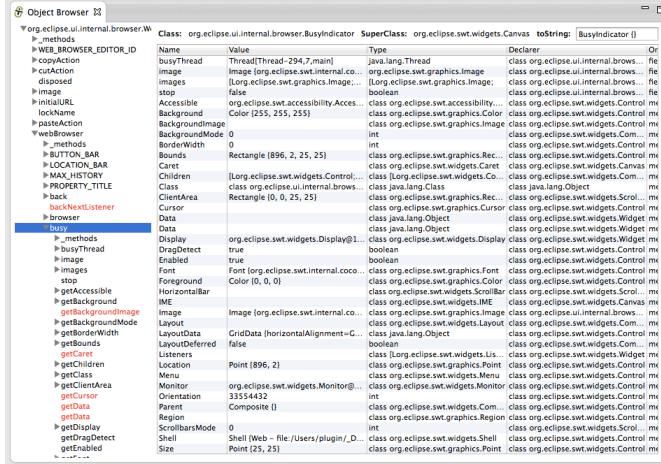
Here is a better view of this object:



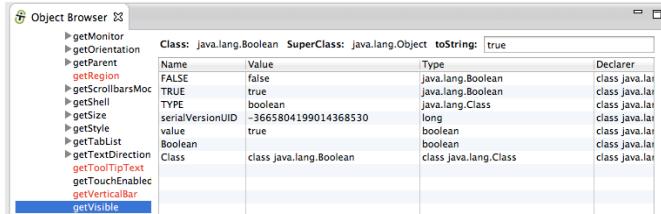
7) looking at the `org.eclipse.ui.internal.WebBrowserEditor` object, the one that called my attention was the `webBrowser` field (which is an `org.eclipse.ui.internal.browser.BrowserViewer` object)



8) Inside the **webBrowser** object I found the **busy** field (which is an *org.eclipse.ui.internal.browser.BusyIndicator* object)



9) and finally inside the\*\* busy object I found the \*\*visible property (ie the **getVisible** and **setVisible** methods)



10) back in the REPL, it was time to do the same thing programmatically.

First I got a reference to the **webBrowser** field:

The screenshot shows the "Write TeamMentor Script" interface. On the left, under "Groovy Code", the following script is written:

```
def editorReference = eclipse.editors.get("Angular with Firebase Lite");
def editor      = editorReference.getEditor(true);
def webBrowser  = editor.webBrowser;
return webBrowser;
```

On the right, under "Execution result", the output is shown as "BrowserViewer {}".

... and then the **busy** field:

The screenshot shows the "Write TeamMentor Script" interface. On the left, under "Groovy Code", the following script is written:

```
def editorReference = eclipse.editors.get("Angular with Firebase Lite");
def editor      = editorReference.getEditor(true);
def webBrowser  = editor.webBrowser;
def busy        = webBrowser.busy;
return busy;
```

On the right, under "Execution result", the output is shown as "BusyIndicator {}".

- 11) Once I had the reference to the **busy** field, it was easy to make it invisible (by just setting the **visible** property to **false**)

The screenshot shows the "Write TeamMentor Script" interface. On the left, under "Groovy Code", the following script is written:

```
def editorReference = eclipse.editors.get("Angular with Firebase Lite");
def editor      = editorReference.getEditor(true);
def webBrowser  = editor.webBrowser;
def busy        = webBrowser.busy;
return busy.visible = false;
```

On the right, under "Execution result", the output is shown as "false".

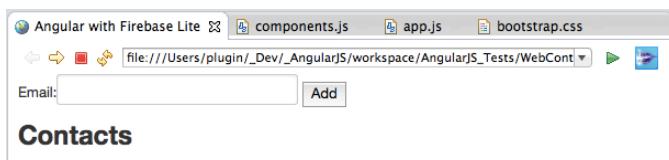
- 12) final step was to write a generic script that would work on all opened browser editor windows (a robust script would use the same trick I used in the integration with the Fortify plugin, where a callback is received on every new view opened (i.e. we could wait for new instances of the `org.eclipse.ui.browser.editor` to be opened, and apply the fix then))

The screenshot shows the "Write TeamMentor Script" interface. On the left, under "Groovy Code", the following script is written:

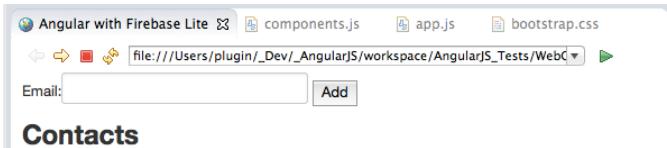
```
def visible = false;
for(editor in eclipse.editors.list())
    if (editor.id == "org.eclipse.ui.browser.editor")
        editor.getEditor(true).webBrowser.busy.visible = visible;
return "done";
```

On the right, under "Execution result", the output is shown as "done".

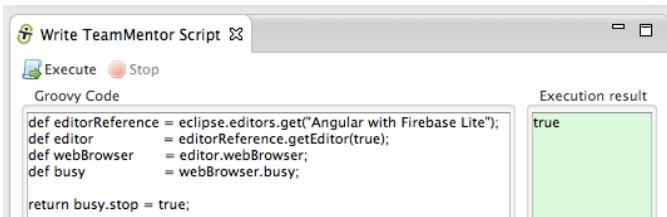
To demonstrate the script in action, here it is the browser before:



... and here it is after:



Note: another way to stop the constant animation was to just set the *stop* value to *false*, but this would only work for the current opened page (i.e. the animation would be back on page reload)



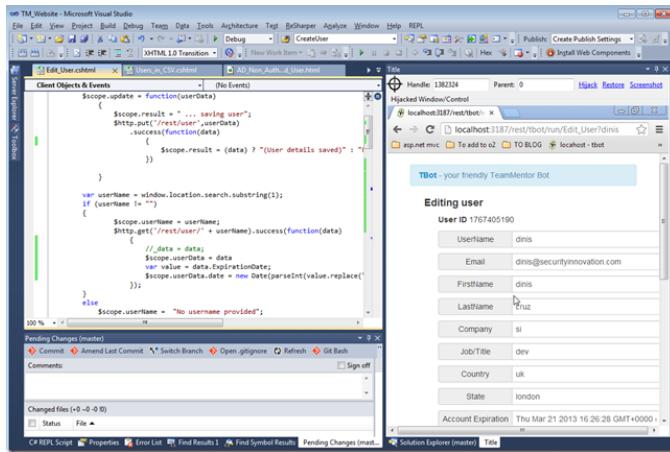
---

[Table of Contents](#) | [Code](#)

## 5.3 Using Chrome inside a native VisualStudio pane (using Window Handle Hijacking)

To help me debug and visualize an [AngularJS](#) page I was developing, I used the O2's [Window Handle Hijack](#) technique to insert an Chrome window inside VisualStudio 2010.

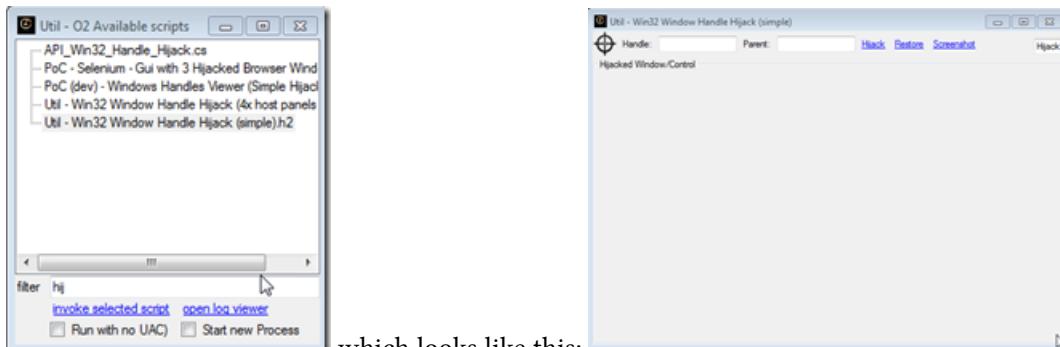
Here it is in action:



On the right you can see a full chrome window, inserted inside a VisualStudio dockable pane.

On the left you can see the AngularJs file (rendered from a RazorSharp template) that I can edit and quickly view its output on the right-hand-side Chrome window (with no web recompilation needed)

To create this, I searched in [O2 Platform](#) for the [Util - Win32 Window Handle Hijack \(simple\).h2](#) script

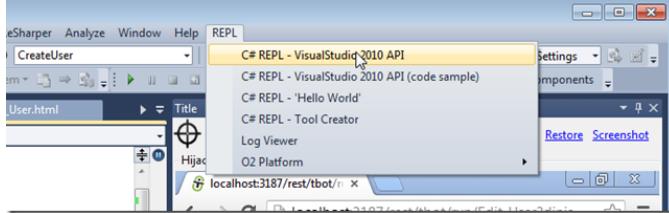


which looks like this:

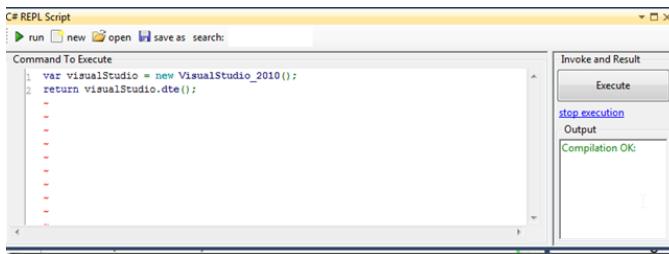
Had a look at its source code to see how it was created (note the Extension Method `add_Handle_HijackGui`):



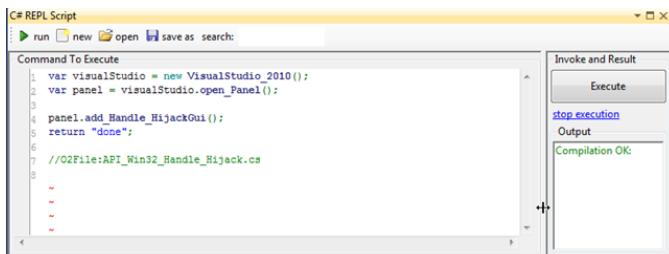
Then inside a VisualStudio, I opened a C# REPL script:



This gave me access to the **VisualStudio2010\_API**



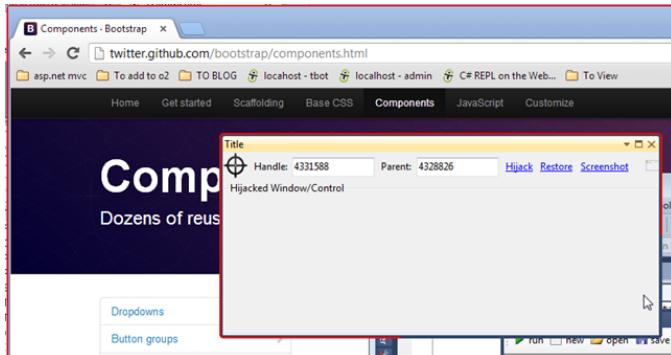
where I can use the Extension Method **add\_Handle\_HijackGui**



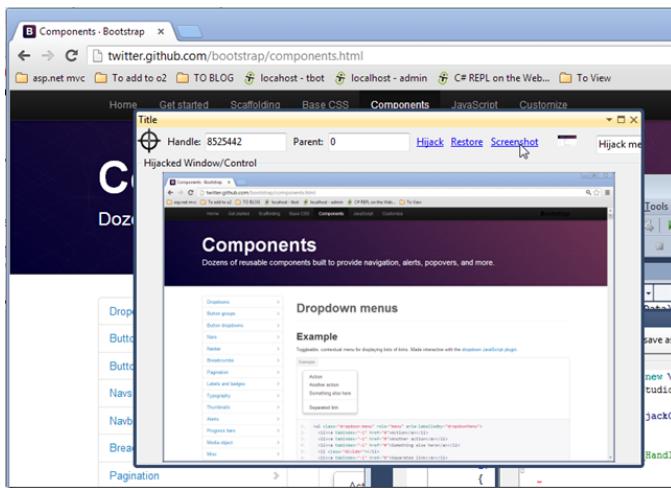
to create a native VisualStudio pane with the **Windows Handle Hijack Gui**



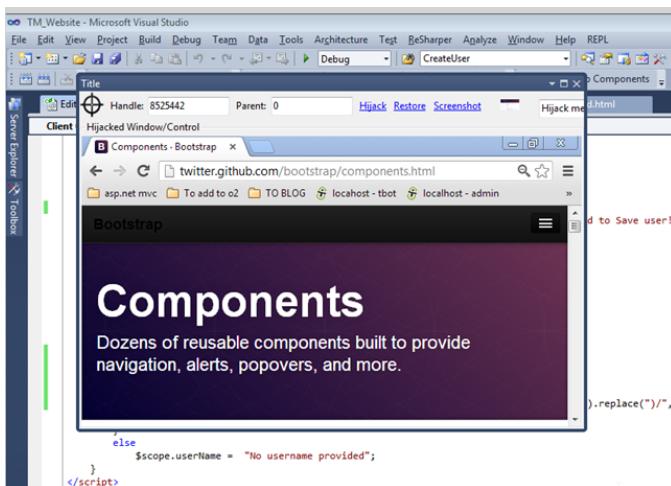
With this GUI, we can grab any Window's Window, by dragging the target icon (top left) into the window we want to use/hijack:



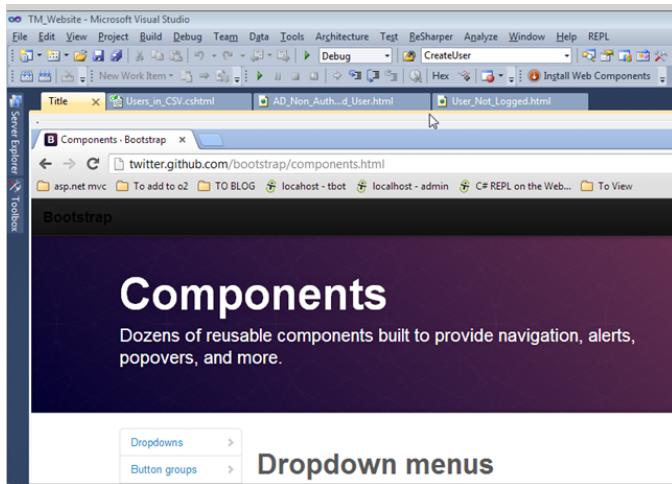
Tip: before Hikacking a window, it is a good idea to take a screenshot and see if we have the right one:



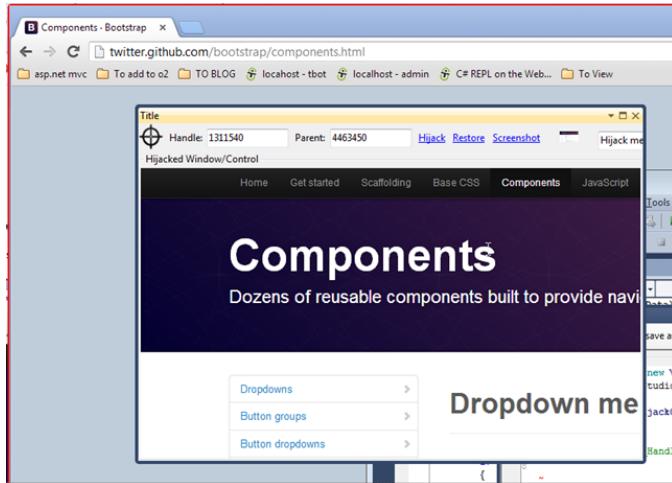
Once we're sure, just hit the Hijack link, and we will have have a fully functional Chrome window that we can place anywhere inside VisualStudio's GUI.



For example, we can place it in the documents area as one of the source code files  
(tip: double click on the 'Hijacked Window/Control' text to hide the hijack controls)



As a final example, here is what it looks like if we just Hijack the browser's website window (without the navigation and top bars)

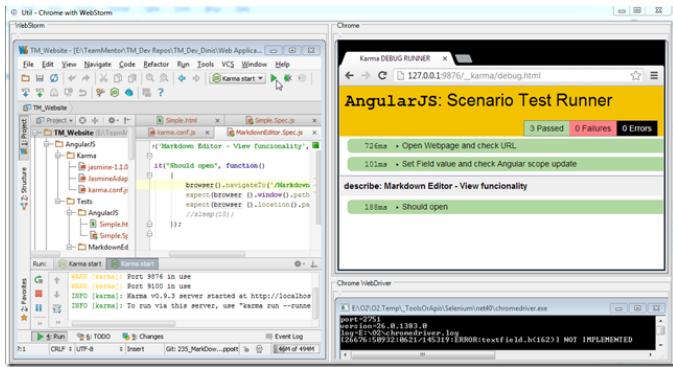


---

[Table of Contents](#) | [Code](#)

## 5.4 Using WebStorm with Chrome and ChromeDriver (to view KarmaJS execution results)

Following from the example described in [When the best way to automate Chrome is to use ... Chrome](#), here is a more practical scenario where I'm creating a GUI that has both WebStorm and Chrome running side-by-side. Here is what it looks like:



What makes this example practical is the KarmaJS auto execution on file change, just like it was described in [Adding KarmaJS support to WebStorm to automagically run tests on file changes](#)

Here is the code that creates this GUI (with some functionality [since the last example](#), but it still needs to a bit of work to make it more robust):

```

1 //var topPanel      = panel.clear().add_Panel();
2 var topPanel      = "Util - Chrome with WebStorm".popupWindow(1200,600);
3 var webStormPanel = topPanel.add_GroupBox("WebStorm").add_Panel();
4
5 //we need to stop any existing WebStorm processes
6 foreach(var process in Processes.getProcessesCalled("WebStorm"))
7 process.stop().WaitForExit();
8
9 // start webstorm
10 var webStorm = @"C:\Program Files (x86)\JetBrains\WebStorm 6.0.2\bin\WebStorm.exe".startPr\
11 ocess();
12
13
14 var chromeHijack = new API_Chrome_Hijack().open_ChromeDriver()
15 .add_Chrome_To_Panel(topPanel.insert_Right());
16 //add_WebDriver_ScriptMe_To(rep1Panel);
17
18 //wait for main window handle
19 webStorm.waitFor_MainWindowHandle();
20 //hack to deal with the splash screen
21 3000.sleep();
22

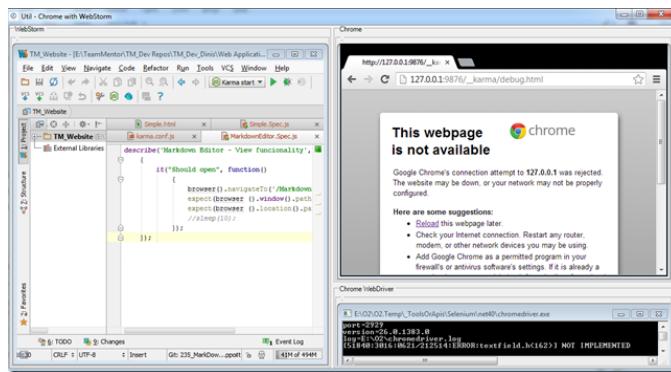
```

```

23 //hijack WebStorm window
24 chromeHijack.hijack_Process_Into_Panel(webStormPanel, webStorm);
25
26 chromeHijack.ChromeDriver.Navigate().GoToUrl("http://127.0.0.1:9876/_karma/debug.html");
27
28
29 //02File:API_Chrome_Hijack.cs
30 //02Ref:WebDriver.dll

```

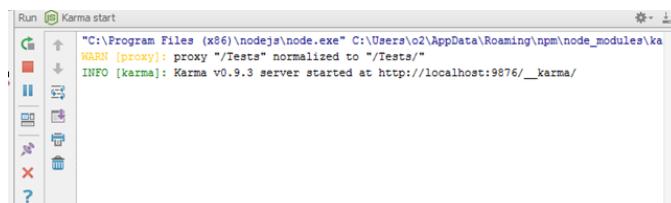
When executed for the first time, this GUI looks like this:



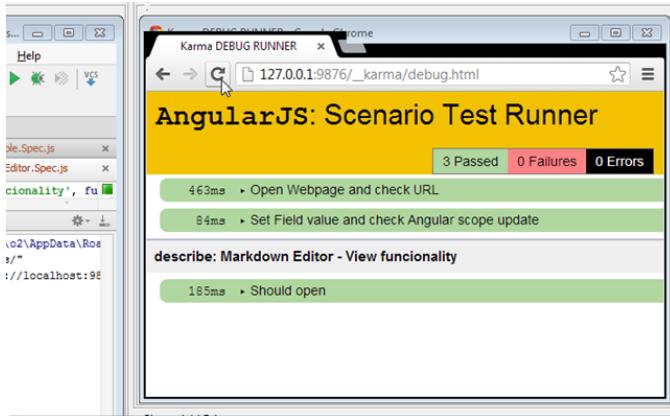
Then if we start KarmaJS:



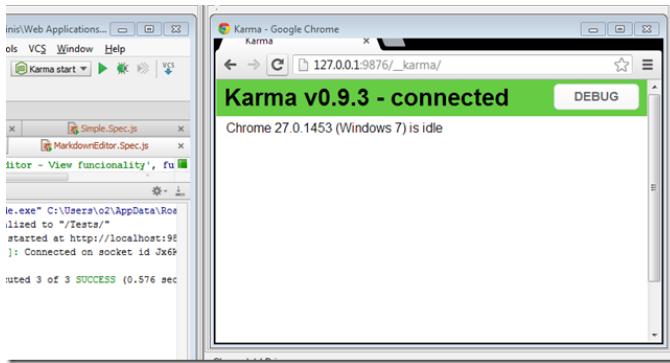
... a listener will start (waiting for browsers to connect):



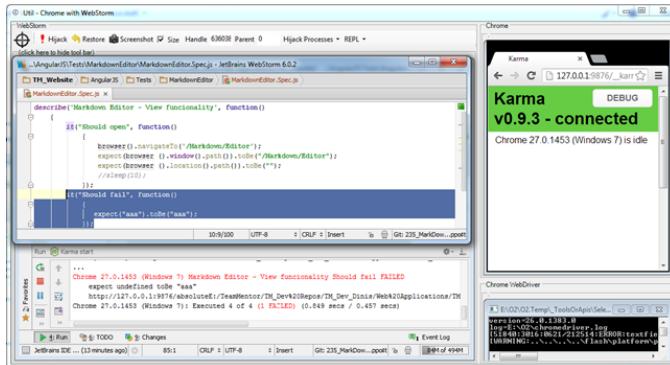
Refreshing the browser will ‘capture it’ and manually execute the tests:



Opening up the normal page will start the ‘auto test execution loop’:



... and if we add a new test and save the file, the unit test execution will occur



Note: there is some kind of weird event race-condition happening and the keyboard key's events are not being fired inside the captured WebStorm. So in the case shown above, I open the file to edit in a WebStorm popup window, where the keyboard events fire-up ok (and I was able to make the changes to the unit test file)

## 5.5 When the best way to automate Chrome is to use ... Chrome (with examples on Google search, direct AngularJS scope manipulation and ChromeDriver javascript access)

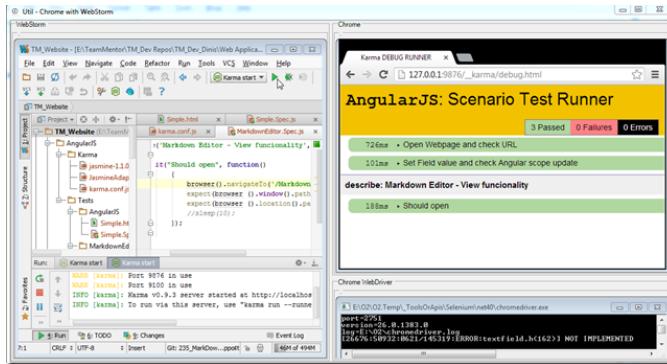
On the topic of Web Automation, I always wanted to have a REPL environment for Chrome like I have for IE ([using Watin](#)).

In the past I have explored multiple solutions, for example the use of CefSharp (see [here](#) and [here](#)). But that was never the real thing, and there was always a couple issues (caused by the fact that the ‘real’ chrome wasn’t being used).

For a while, in the back on my mind the solution was simple and obvious: *Use the real Chrome process in a way that it can be programmatically accessed from an O2’s C# repl environment!*

Well, the good news is that is exactly what I have done :)

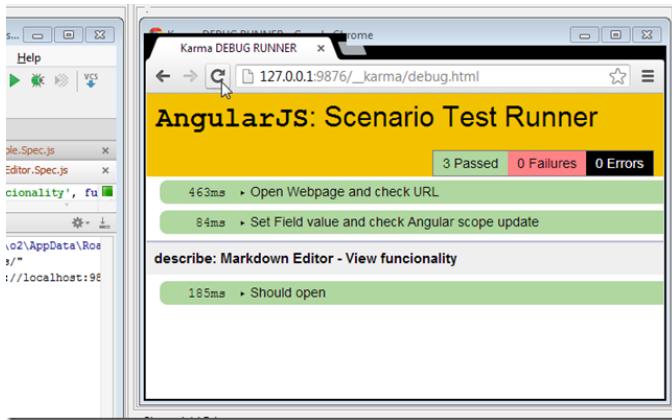
I just created the Gui you can see below, which uses the [Window-Hikacking](#) technique to inject an (Selenium’s ChromeDriver started) Chrome process’ window in a Panel, and pass its reference (as a variable) to an O2 REPL environment.



The script is called \_ Util - Chrome Browser with REPL.h2\_ and it has:

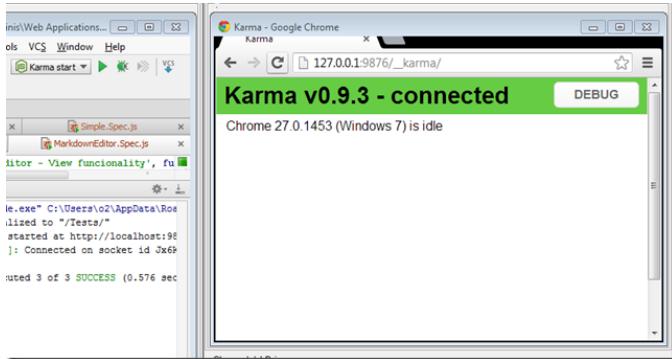
- The cmd.exe window from the ChromeDriver.exe process in the bottom right
- The Chrome window (started by the ChromeDriver) in the top right
- A C# REPL (with the ChromeDriver object passed in as a parameter) in the left

Now, on the left-hand-side [C# REPL](#), I can write a script like this:



Which will:

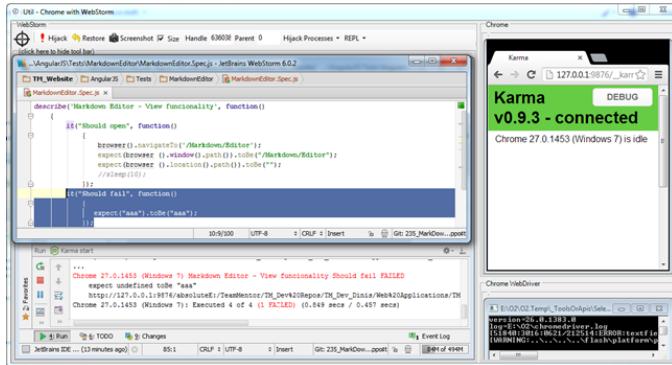
- Open (in Chrome) the <http://www.google.com> web page
- Find the 'q' element (the search text input)
- Send the Keys 'O2 Platform' to that element
- Find the element 'btnG' (the search button)
- Click on that element



## Automating an AngularJS page

\*\*

The next example is more interesting where we are going to open AngularJS page and programmatically change a \$scope variable

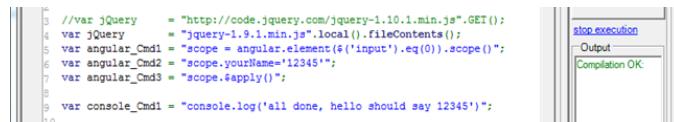


Lets look at the code to see what is going on.

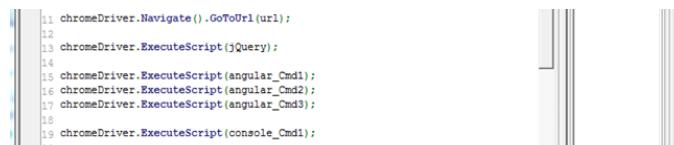
The first line opens up an the Url with the sample AngularJS example described [here](#) and [here](#)



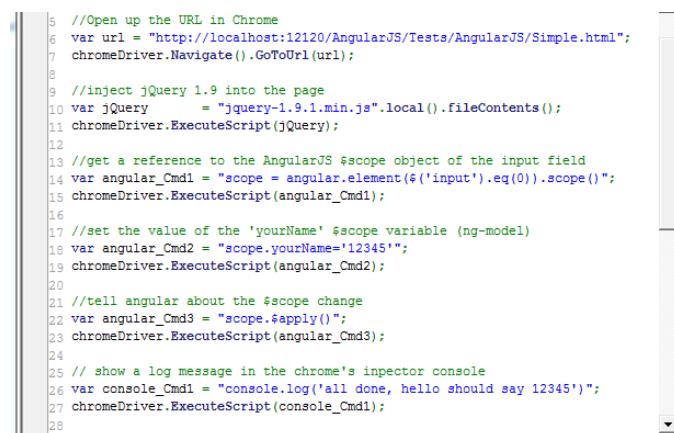
... next I create a number of variables with the url and commands to execute:



... here:



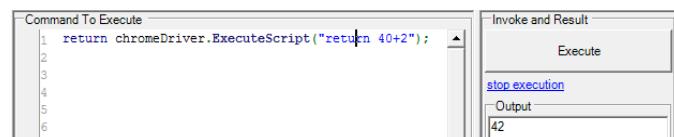
It will probably be easier if I break these commands individually:



And this is all programmed in a nice REPL environment, which makes it really easy to try thing out.

For example, here are a couple interesting two way data exchange between the C# script and the Chrome javascript:

### 1) get an int value



### 2) get an string value:

Command To Execute	Invoke and Result
1    return chromeDriver.ExecuteScript("return '40+2';")	<div style="border: 1px solid black; padding: 5px; width: fit-content;">Execute</div>
2	
3	
4	
5	
6	
	<a href="#">stop execution</a>
	<a href="#">Output</a>
	40+2

3) get an simple object (converted into dictionary)

# Repl

run new open save as search:

Command To Execute

```
1 return chromeDriver.ExecuteScript(
2     @"return {
3         key : 'value',
4         1 : 2,
5         another: 'example'
6     }");
```

Invoke and Result

Execute

[stop execution](#)

Output

```
[1, 2]
[another, example]
[key, value]
```

Misc

- Key another
- Val example

4) get an array

Command To Execute

```
1 return chromeDriver.ExecuteScript(  
2         @"return [ 'a',  
3             'b',  
4             12,  
5             {  
6                 key : 'value',  
7                 1 : 2,  
8                 another: 'exam  
9             }  
10            ]");  
11  
12 //using OpenQA.Selenium;  
13 //O2Ref:WebDriver.dll
```

Invoke and Result

Execute

Stop execution

Output

```
12  
a  
b  
System.Collections.Generic.Dictionary`2[[
```

Misc

Compare  
Count 3  
Keys (Collection)  
Values (Collection)

## 5) get the document.location object

Command To Execute

```
1 return chromeDriver.ExecuteScript(  
2     @"return document.location");  
3  
4 //using OpenQA.Selenium;  
5 //O2Ref:WebDriver.dll  
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  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2089  
2090  
2091  
20
```

6) get the 'this' object

Command To Execute

```
1 return chromeDriver.ExecuteScript(  
2     @"return this");  
3  
4 //using OpenQA.Selenium;  
5 //O2Ref:WebDriver.dll  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Invoke and Result

Execute

[Stop execution](#)

Output

— [WINDOW .wdc:1371809534500]

Misc	
Key	WINDOW
Value	wdc:1371809534500

\*\*7) get the first document.element \*\*

```
Command To Execute
1 return chromeDriver.ExecuteScript(
2     @"return document.all[0];"
3
4 //using OpenQA.Selenium;
5 //O2Ref:WebDriver.dll
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Invoke and Result

Execute	
<a href="#">stop execution</a>	<a href="#">Output</a>
<pre>Coordinates Displayed True Enabled True Location 0,0 LocationOnScreen 0,0 Selected False Size 387,124 TagName Html Text Name:Hello ! WrappedDriver</pre>	

### 8) get the html of the first document.element

```
Command To Execute
1 return chromeDriver.ExecuteScript(
2     @"return document.all[0].innerHTML"
3
4 //using OpenQA.Selenium;
5 //O2Ref:WebDriver.dll
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Invoke and Result

Execute	
<a href="#">stop execution</a>	<a href="#">Output</a>
<pre>&lt;head&gt; &lt;meta charset="utf-8"&gt; &lt;meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9"&gt; &lt;title&gt;Simple AngularJS page&lt;/title&gt; &lt;script src="/Scripts/angular.js"&gt;&lt;/script&gt; &lt;script type="text/css"&gt;@charset "UTF-8";&lt;/script&gt; &lt;ng-cloak&gt;&lt;div&gt;&lt;label&gt;Name:&lt;/label&gt; &lt;input type="text" ng-model="yourName" placeholder="Enter a name here" class="ng-valid ng-dirty"&gt; &lt;br&gt; &lt;h1&gt;Hello some text!&lt;/h1&gt; &lt;/div&gt;</pre>	

### 9) show an alert

```
Command To Execute
1 chromeDriver.ExecuteScript("alert(12)");
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

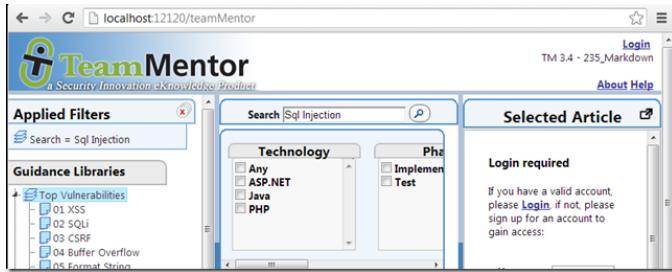
Invoke and Result

Execute	
<a href="#">stop execution</a>	<a href="#">Output</a>

### 10) getting and using an Html element via ExecuteScript

```
Command To Execute
1 //open Url
2 /*chromeDriver.ExecuteScript(
3     "document.location= 'http://localhost:12120'; */
4
5 //Getting the SearchTextBox object via ExecuteScript
6 var searchElement = (RemoteWebElement)chromeDriver.ExecuteScript(
7     "return document.getElementById('SearchTextBox')");
8 searchElement.Clear();
9 searchElement.SendKeys("Sql ");
10
11 //Getting the SearchTextBox object via Selenium selector
12 chromeDriver.FindElement(By.Id("SearchTextBox"))
13     .SendKeys("Injection");
14
15 //Click on search Button
16 chromeDriver.FindElement(By.Id("ctl00_ContentPlaceHolder1_SearchControl1_SearchButton"))
17     .Click();
18
19 return "done";
20
21 //using OpenQA.Selenium.Remote
22 //using OpenQA.Selenium;
23 //O2Ref:WebDriver.dll
```

which will populate the search text value and click on the search icon



What is cool about all these examples is that we are running the code on the local installation of Chrome, ie it is the actually chrome process that we are using, which is great for testing, debugging and developing :)

### Scripts used in this post

#### A) C# code that created the GUI (start chromeDriver and chrome processes and hijack they main window)

```

1 //var topPanel      = panel.clear().add_Panel();
2 var topPanel = "Util - Chrome with REPL".popupWindow(1200,600);
3
4 var replPanel = topPanel.add_GroupBox("C# REPL").add_Panel();
5 var chromePanel = topPanel.insert_Right(replPanel.width() / 2 , "Chrome");
6 var chromeDriver = chromePanel.parent().insert_Below(150,"Chrome WebDriver");
7
8 var firstScript =
9 @"chromeDriver.open(""http://www.google.com"");
10 chromeDriver.FindElement(By.Name("q"))
11 .SendKeys("O2 Platform");
12 chromeDriver.FindElement(By.Name("btnG"))
13 .Click();
14 return ""done"";
15
16 //using OpenQA.Selenium;
17 //O2Ref:WebDriver.dll
18 //O2File:API_ChromeDriver.cs";
19
20 var chromeHijack = new API_Chrome_Hijack()
21 .open_ChromeDriver();
22
23 chromeHijack.ChromeDriver.script_Me(replPanel).set_Code(firstScript);
24 var hijacked_Chrome = chromePanel.add_Handle_HijackGui(false)
25 .hijackProcessMainWindow(chromeHijack.ChromeProcess);
26 var hijacked_ChromeDriver = chromeDriver.add_Handle_HijackGui(false)
27 .hijackProcessMainWindow(chromeHijack.ChromeDriverProcess);
28
29 //O2File:API_Chrome_Hijack.cs
30 //O2File:API_Win32_Handle_Hijack.cs

```

```
31
32 //O2Ref:WebDriver.dll
33
34 **B) first code example (open Google and do a search)**
35
36
37     chromeDriver.open("http://www.google.com");
38     chromeDriver.FindElement(By.Name("q"))
39         .SendKeys("O2 Platform");
40     chromeDriver.FindElement(By.Name("btnG"))
41         .Click();
42
43     return "done";
44
45 //using OpenQA.Selenium;
46 //O2Ref:WebDriver.dll
47 //O2File:API_ChromeDriver.cs
48
49 **C) 2nd code example, open AngularJS page and programmatically change a $scope variable**\
50
51
52     var url          = "http://localhost:12120/AngularJS/Tests/AngularJS/Simple.html";
53
54 //var jQuery = "http://code.jquery.com/jquery-1.10.1.min.js".GET();
55 var jQuery = "jquery-1.9.1.min.js".local().fileContents();
56 var angular_Cmd1 = "scope = angular.element($('input').eq(0)).scope()";
57 var angular_Cmd2 = "scope.yourName='12345'";
58 var angular_Cmd3 = "scope.$apply()";
59
60 var console_Cmd1 = "console.log('all done, hello should say 12345')";
61
62 chromeDriver.Navigate().GoToUrl(url);
63
64 chromeDriver.ExecuteScript(jQuery);
65
66 chromeDriver.ExecuteScript(angular_Cmd1);
67 chromeDriver.ExecuteScript(angular_Cmd2);
68 chromeDriver.ExecuteScript(angular_Cmd3);
69
70 chromeDriver.ExecuteScript(console_Cmd1);
71
72 return "done";
73
74 // doesn't work to open chrome's inspector
75 // chromeDriver.Keyboard.SendKeys(OpenQA.Selenium.Keys.F12);
```

```
76
77 //using OpenQA.Selenium;
78 //O2Ref:WebDriver.dll
79 **D) last example where search filed was retrieved using two different techniques:**
80
81
82     //open Url
83     /*chromeDriver.ExecuteScript(
84         "document.location= 'http://localhost:12120'"); */
85
86 //Getting the SearchTextBox object via ExecuteScript
87 var searchElement = (RemoteWebElement)chromeDriver.ExecuteScript(
88 "return document.getElementById('SearchTextBox')");
89 searchElement.Clear();
90 searchElement.SendKeys("Sql ");
91
92 //Getting the SearchTextBox object via Selenium selector
93 chromeDriver.FindElement(By.Id("SearchTextBox"))
94 .SendKeys("Injection");
95
96 //Click on search Button
97 chromeDriver.FindElement(By.Id("ctl00_ContentPlaceHolder1_SearchControl1_SearchButton"))
98 .Click();
99
100 return "done";
101
102 //using OpenQA.Selenium.Remote
103 //using OpenQA.Selenium;
104 //O2Ref:WebDriver.dll
```

---

[Table of Contents](#) | [Code](#)

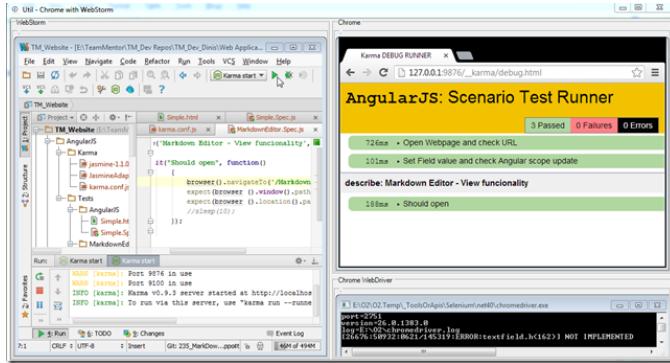
## 5.6 Adding KarmaJS support to WebStorm to automagically run tests on file changes (and test UI with SublimeText, Chrome and Cmd.exe)

On the *AngularJs* \*\*and *KarmaJS*\*\* theme (see [A small AngularJS Jasmine test executed by KarmaJS](#) and the related posts linked at the bottom), here is my first attempt at using Karma to test AngularJS code inside [TeamMentor](#).

I'm using WebStorm instead of VisualStudio, since for Javascript coding [WebStorm](#) is MUCH better/faster/-cleverer, specially since it has good support for AngularJs and Jasmine (with KarmaJS support easily added, as we are about to see).

Also shown below is a cool tool I created that hijacks windows from SublimeText, Chrome and Cmd.exe windows into the same UI (an O2 Platform .NET Script)

Here is the directory structure:



... with *Karma.config.js* looking like this:

```

1 module.exports = function(karma)
2 {
3     karma.configure(
4         {
5             frameworks: ['ng-scenario'],
6             files:
7                 [
8                     '../Tests/**/*.Spec.js'
9                 ],
10            urlRoot: '/__karma/',
11            autoWatch: true,
12            proxies: {
13                '/' : 'http://localhost:12120/'
14            }
15        }
16    );
17}

```

```
18          '/Tests': 'http://localhost:12120/AngularJs/Tests/'  
19      },  
20  
21      //browsers: ['Chrome'],  
22  
23      reporters: ['dots'],    //reporters: ['progress'],  
24      plugins: [  
25          'karma-ng-scenario',  
26          'karma-chrome-launcher'  
27          //'karma-firefox-launcher'  
28          ],  
29      //logLevel: karma.LOG_DEBUG  
30      logLevel: karma.LOG_INFO  
31  );  
32};  
33};
```

... \*\*Simple.html \*\*like this:

```
1 <!DOCTYPE html>  
2 <html xmlns:ng="http://angularjs.org" id="ng-app" ng-app>  
3 <head>  
4     <meta charset="utf-8">  
5     <meta http-equiv="X-UA-Compatible" __content="IE=EmulateIE9" />  
6     <title>Simple AngularJS page</title>  
7     <script src="/Javascript/angularJS/1.0.5/angular.min.js"></script>  
8 </head>  
9 <body>  
10 <div>  
11     <label>Name:</label>  
12     <input type="text" ng-model="yourName" placeholder="Enter a name here">  
13     <hr>  
14     <h1>Hello {{yourName}}!</h1>  
15 </div>  
16 </body>  
17 </html>
```

... Simple.Spec.js like this:

```
1 describe('Test Angular Simple page', function()
2 {
3     beforeEach(function()
4     {
5         browser().navigateTo('/Tests/AngularJS/Simple.html');
6     });
7     it('Open Webpage and check URL', function()
8     {
9         expect(browser().window().path()).toBe("/Tests/AngularJS/Simple.html");
10        expect(browser().location().path()).toBe("");
11        browser().navigateTo('/Tests/AngularJS/aaa.html');
12        expect(browser().window().path()).not().toBe("/Tests/AngularJS/Simple.htm\\
13 1");
14        expect(browser().window().path()).toBe("/Tests/AngularJS/aaa.html");
15    });
16
17    it('Set Field value and check Angular scope update', function()
18    {
19        var testValue = "This is a value";
20        var expectedValue = "Hello " + testValue + "!";
21        input('yourName').enter(testValue);
22        expect(element('.ng-binding').text()).toEqual(expectedValue);
23    });
24
25    it('get path value manually (using setTimeout)',function()
26    {
27        var path = browser().window().path();
28        console.log("(before timeout) path value = " + path.value );
29        setTimeout(function()
30        {
31            console.log("(after timeout) path value = " + path.value );
32            //expect("value").toBe("/123");
33
34        },200);
35        sleep(1);
36    });
37});
```

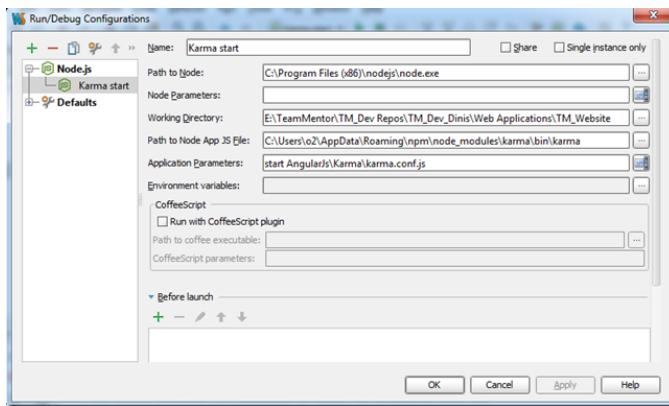
... and **MarkDown.Editor.Spec.js** currently with just

```

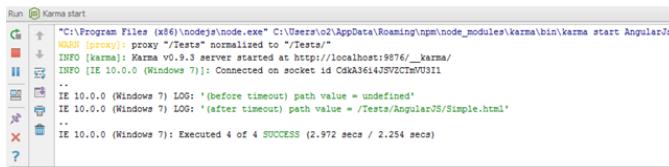
1 describe('Markdown Editor - View functionality', function()
2 {
3     it("Should open", function()
4     {
5         browser().navigateTo('/Markdown/Editor');
6         expect(browser().window().path()).toBe("/Markdown/Editor");
7         expect(browser().location().path()).toBe("");
8         //sleep(10);
9     });
10 });

```

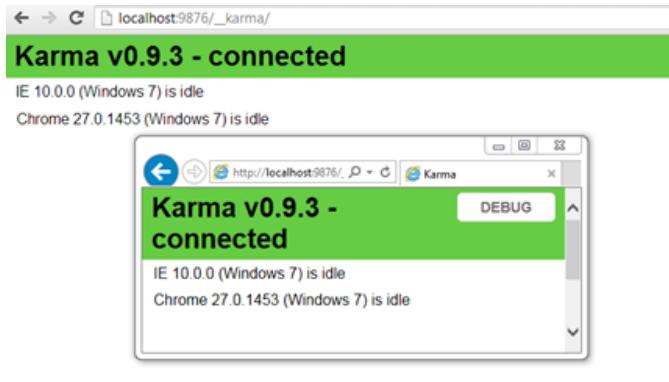
This is how I configured Karma to run on WebStorm:



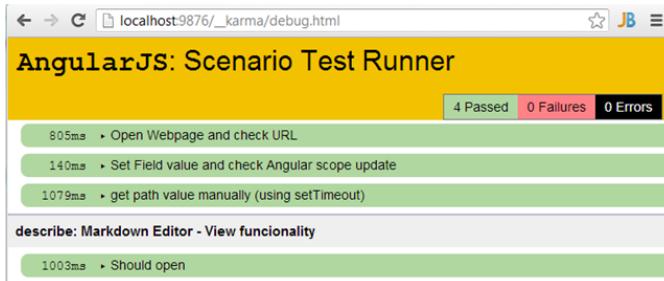
Here is KarmaJS execution log:



Here are two connected KarmaJS Runners (one in IE and one in Chrome)



And finally, here is the (super useful) **AngularJS: Scenario Test Runner** view, in a collapsed state:



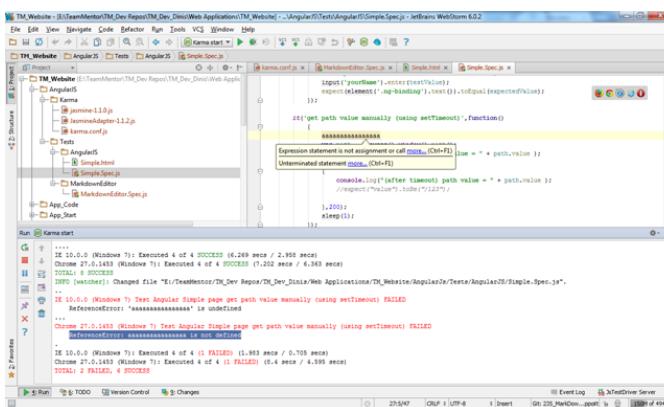
... and in an expanded state:



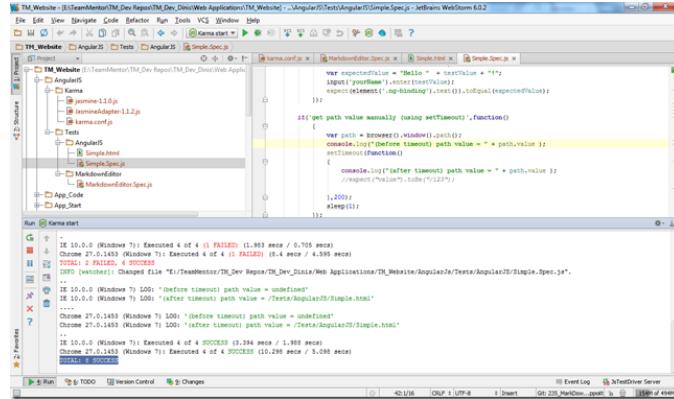
With this set-up KarmaJS is configured to run continuously and to monitor any file changes, which is really powerful, since it allows for continuous development and testing.

For example (to see the automagically execution in action), here is what the WebStorm UI looks like:

\*\* ... with an Javascript error:\*\* (KarmaJS execution triggered on Save) :



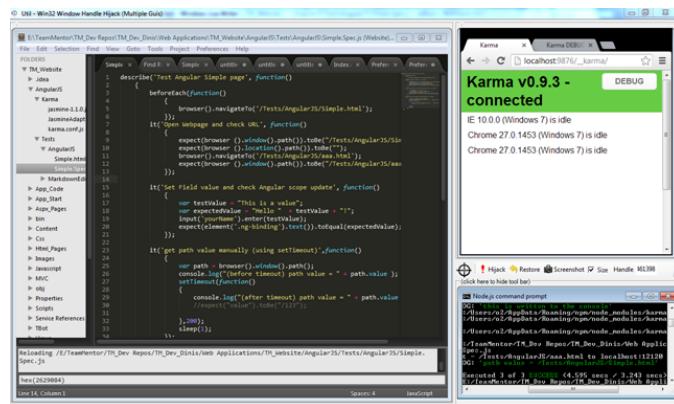
\*\*... without the error: \*\* (KarmaJS execution triggered on Save)



This is a really nice environment to quickly develop AngularJS apps, specially since the tests are super quick to execute and we can control what tests are executed (for example by creating multiple karma.conf.js files)

## O2 Platform Test GUI (created by hijacking 3 processes' windows)

I also created a test GUI using the O2 Platform's window-handle Hijack capabilities, which looked like this:



In the image above:

- The host process is the O2 Platform Util - Win32 Window Handle Hijack (4x host panels).h2 script/tool (which is a .Net app)
  - The left-hand side is the **Sublime Text editor** (which is a C++, Python app)
  - The top-right is **Chrome** (which is C++ app)
  - The bottom-right is cmd.exe (C++ app) running the command **karma start karma.conf.js** **\*\*in the \*\*E:TeamMentorTM\_Dev ReposTM\_Dev\_DinisWeb ApplicationsTM\_WebsiteAngularJSKarma folder**

All in a nice, self-contained process/environment

Note that there are 4 separate process at play here (O2 Platform, Sublime, Chrome and Cmd.Exe w/ Karma)

Related AngularJS and KarmaJS posts:

- A small AngularJS Jasmine test executed by KarmaJS
  - KarmaJS AngularJS Scenario Test Runner execution variations in IE 7,8,9 and 10 when using AngularJS
  - Debugging a weird case of missing module in AngularJS and KarmaJS
  - Running KarmaJS's AngularJS example test/e2e/angular-scenario (on Chrome)
  - AngularJS code editor using UI-Bootstrap and CodeMirror (done without using jQuery)
  - Hubspot current.js code includes JQuery on it
  - Submitting TM users to HubSpot via TBOT interface (using Angular JS)
  - Using Chrome inside a native VisualStudio pane (using Window Handle Hijacking)
  - If AngularJS doesn't work on your O2 Platform IE scripts (the fix is to change browser compatibility mode)
- 

[Table of Contents](#) | [Code](#)

# 6 Troubleshooting

This section has the following chapters:

- KarmaJS AngularJS Scenario Test Runner execution variations in IE 7,8,9 and 10 when using AngularJS
  - If AngularJS doesn't work on your O2 Platform IE scripts (the fix is to change browser compatibility mode)
  - Debugging a weird case of missing module in AngularJS and KarmaJS
- 

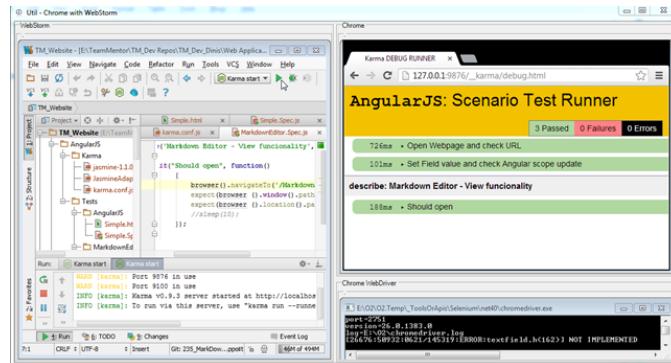
[Table of Contents](#) | [Code](#)

## 6.1 KarmaJS AngularJS Scenario Test Runner execution variations in IE 7,8,9 and 10 when using AngularJS

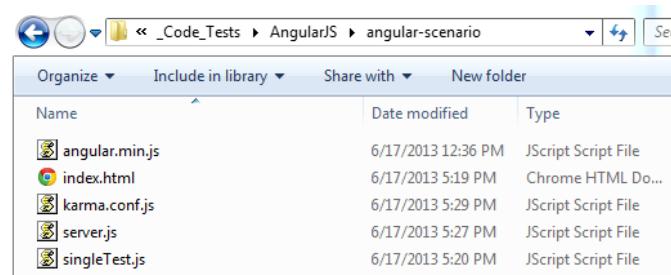
While trying to get [Karma JS](#) to work, I found a number of different behaviours for its AngularJS Scenario Test Runner in IE's multiple ‘compatibility modes’.

TLDR: some of the Jasmine and AngularJS test apis don't work (although Angular does seem to work ok)

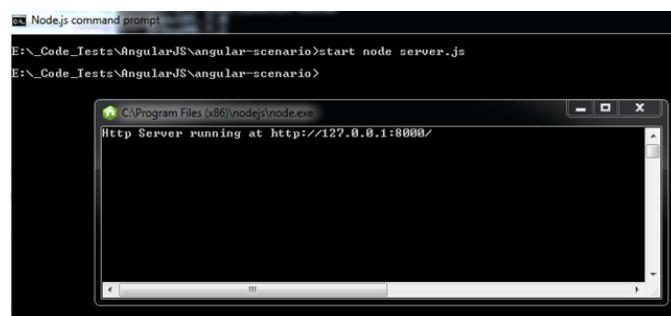
Here is the default web page I was using:



Here is the test executed



Here is KarmaJS starting and successfully executing the tests

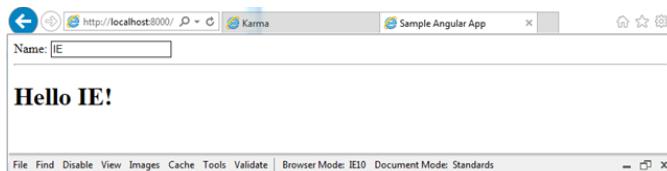


... in this captured IE browser session:

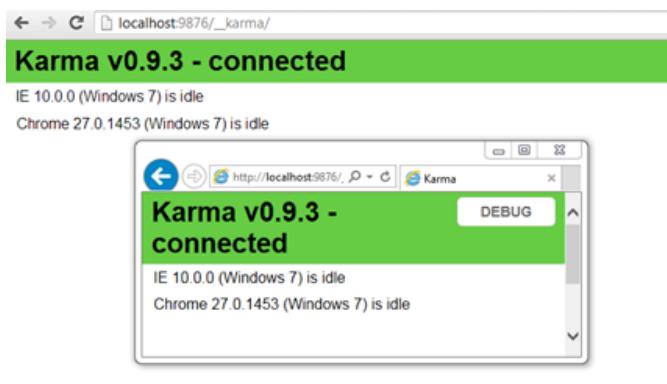


Just to confirm that the target page works in the multiple IE configurations, here it is running in:

IE 10 , IE9, IE 8:

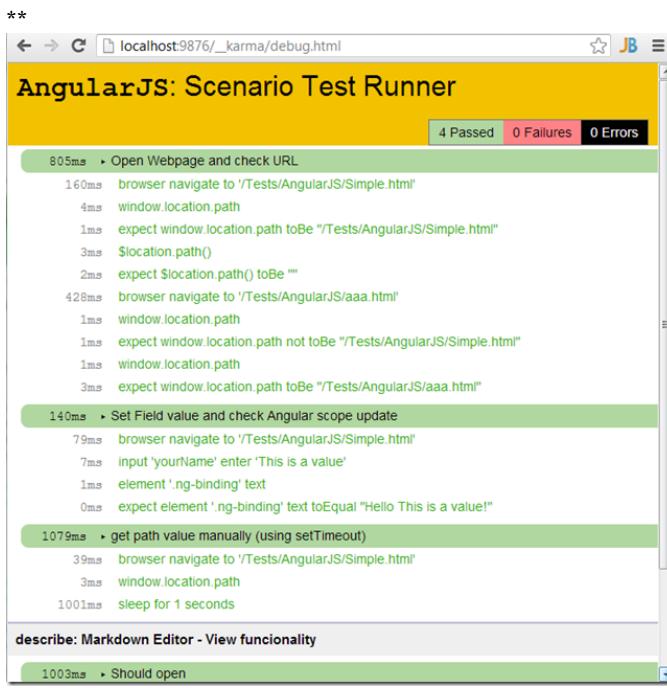


... and even in IE 7:



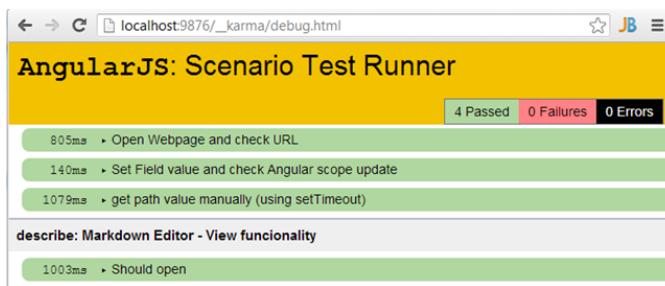
Now lets click on the \_DEBUG \_button to open the KarmaJS's\_ AngularJS Scenario Test Runner\_ view and see what happens in multiple IE compatibility modes.

**IE 10 Works:**



\*\*\*\*IE 10 Compatibility View

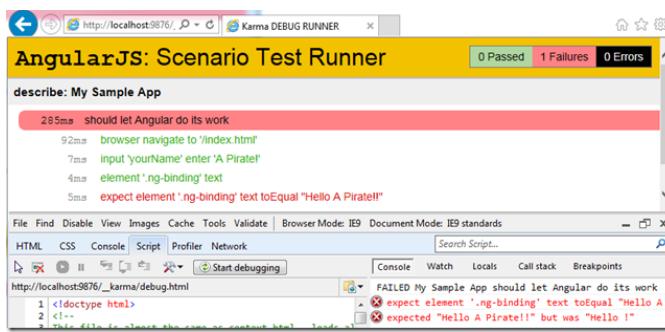
\*\*



\*\*

**\*\*\*\*IE 9 Fails:**

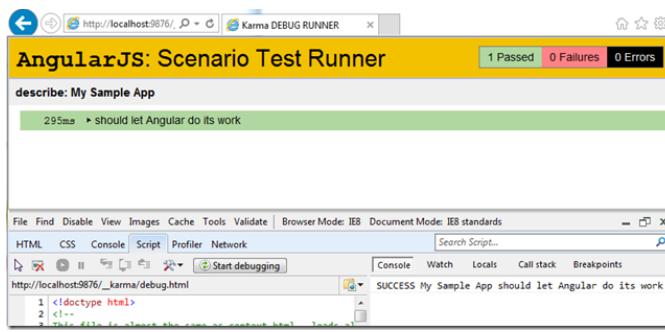
\*\*



\*\*

**\*\*\*\*IE 8 Works (WTF!!)**

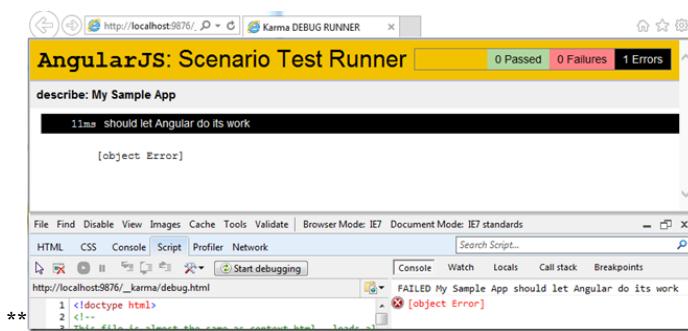
\*\*



\*\*

**\*\*\*\*IE 7 Fails**

\*\*



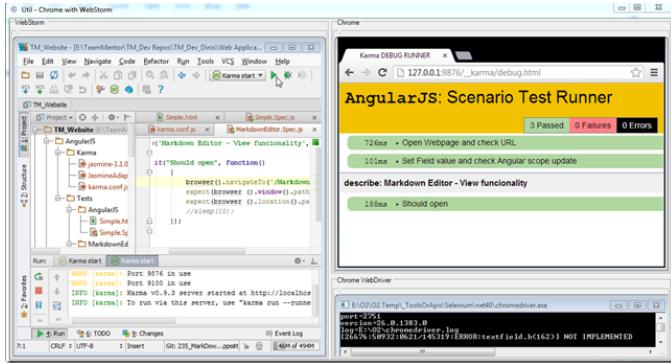
So unfortunately it looks like this technique can't be used to run e2e (end-to-end) tests on AngularJS apps using KarmaJS

---

[Table of Contents](#) | [Code](#)

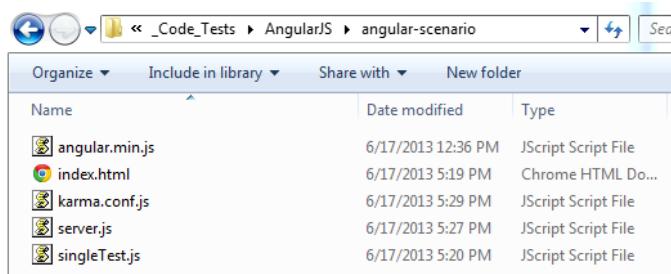
## 6.2 If AngularJS doesn't work on your O2 Platform IE scripts (the fix is to change browser compatibility mode)

If when trying to open an AngularJS page inside an O2 Platform script, you see:

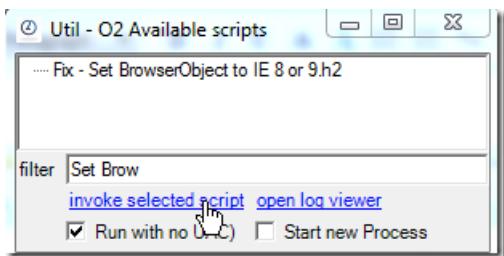


... this means that the IE browser embedded in that .NET process is set to run under IE 7

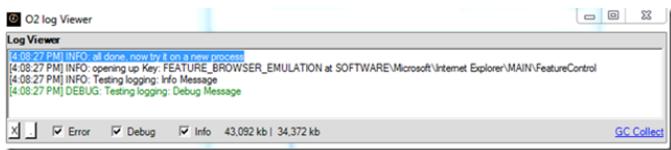
To confirm it, try opening the <http://www.whatismybrowser.com> and you should see something like:



As mentioned in the [set .NET WebBrowser Control to use latest version of IE](#) post to. change it on your system, run this script



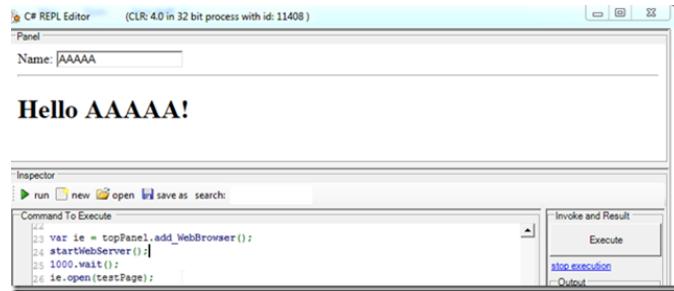
... as admin with no UAC



...and now after restarting the O2 Platform process, IE should be on Internet Explorer 9 compatibility mode  
\*\*



... and AngularJS should work:



\*\*Note 1: \*\*if you control the site you are testing, you can also add this also works to make it work (with the advantage that it is not exe specific)

1    \_<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9" />\_

---

[Table of Contents](#) | [Code](#)

## 6.3 Debugging a weird case of missing module in AngularJS and KarmaJS

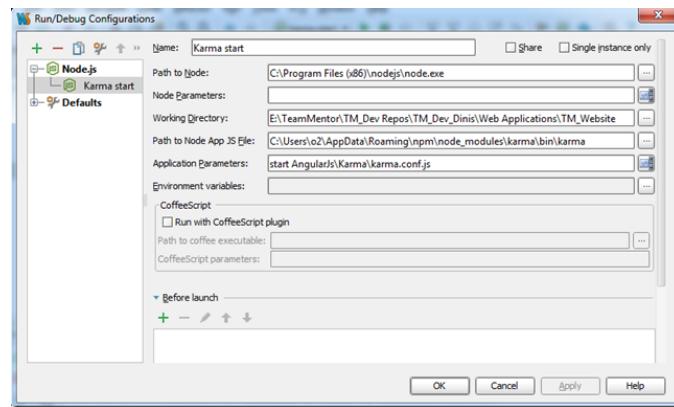
When I was trying the [Running KarmaJS's AngularJS example test/e2e/angular-scenario \(on Chrome\)](#) I hit on the the following weird behaviour.

TLDR; the solution was to run `npm install -g **** karma@canary`

### Setup

\*\*

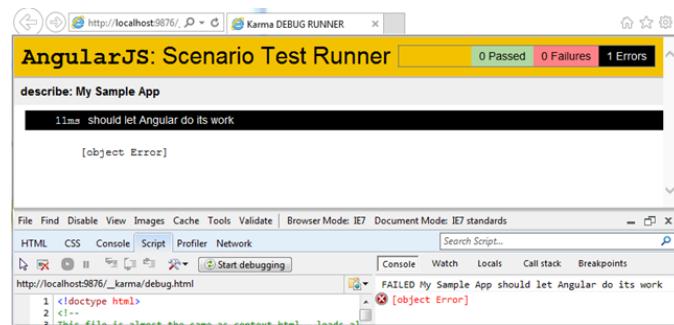
\*\*Chrome window opened in:



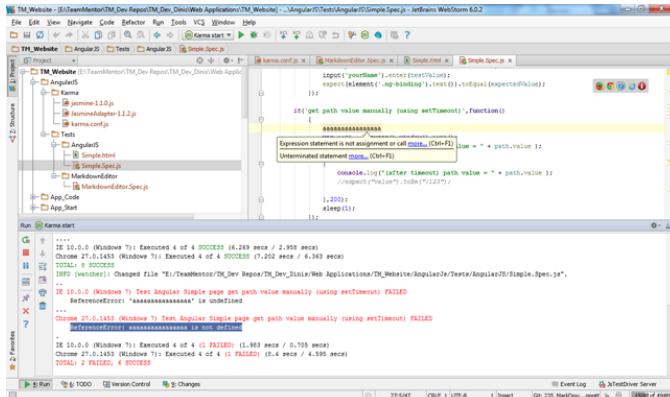
... a local website at port 8000:

```
E:\Code\Tests\SingularJS\angular-scenario>karma start karma.conf.js
INFO [Karma]: Karma v0.14.3 serve started at http://localhost:9876/
INFO [Chromedriver]: Chromedriver v2.14.1453 (Windows 7) Connected on socket id 5WJYzMu60uMjZ-2v
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (0.368 secs / 0.184 secs)
```

... which is a nodeJS powered website, started using `node server.js` (below)



A simple (as I could make it) `_ Karma.config.js _file`



.... and AngularJS test

```
1  /** A Sample Angular E2E test */
2
3  describe('My Sample App', function() {
4
5      it('should let Angular do its work', function() {
6          browser().navigateTo('/index.html');
7          input('yourName').enter('A Pirate!');
8          expect(element('.ng-binding').text()).toEqual('Hello A Pirate!!!');
9      });
10 })
```

\* \*

\*\*\*\*Scenario A) Running from folder with karma clone (and npm install)\*\*\*\*  
\*\*

\*\*\*\* karma start ..\angular-scenario\karma.conf.js \*\*works OK

```
E:\Code\Tests\AngularJS\karma>karma start ..\angular-scenario\karma.conf.js
DEBUG [config]: autoWatch set to false, because of singleRun
DEBUG [plugin]: Loading plugin karma-ng-scenario.
INFO [karma]: Karma v0.9.3 server started at http://localhost:9876/_karma/
DEBUG [watcher]: Resolved files:
  E:\Code\Tests\AngularJS\karma\node_modules\karma-ng-scenario\lib\angular-s
  cenario.js
  E:\Code\Tests\AngularJS\karma\node_modules\karma-ng-scenario\lib\adapter.j
  S
  E:\Code\Tests\AngularJS\angular-scenario\e2eSpec.js
DEBUG [karma]: New browser has connected on socket #9aNFLIX_UCPEn_NPleui-
INFO [Chrome 27.0.1453 (Windows 7)]: Connected on socket #9aNFLIX_UCPEn_NPleui-
DEBUG [karma]: All browsers are ready, executing
  user's server-side test
  E:\Code\Tests\AngularJS\karma\node_modules\karma\stats
DEBUG [proxy]: proxying request - /index.html to localhost:8000
DEBUG [proxy]: proxying request - /angular.min.js to localhost:8000
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (5.388 secs / 4.173 secs)
DEBUG [launcher]: Disconnecting all browsers
```

**Scenario B) running from parent folder**

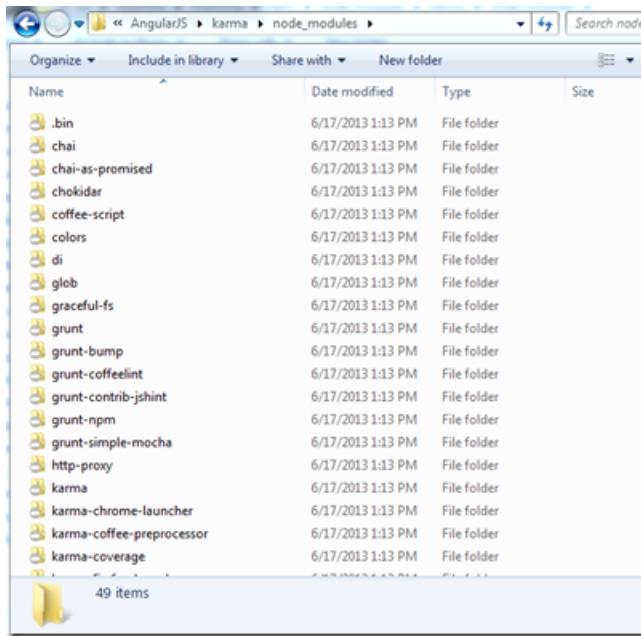
\* \*

\*\*\*\* karma start angular-scenario\\karma.conf.js \_fails with a **\*\* module is not defined error**

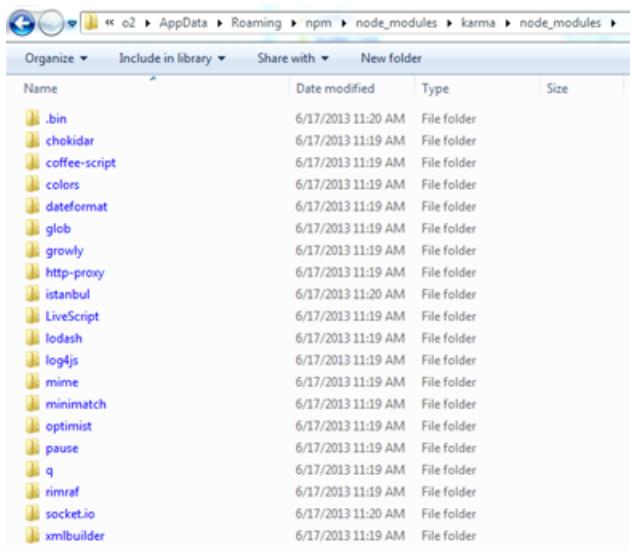
```
E:\_Code_Tests\AngularJS>karma>cd ..
E:\_Code_Tests\AngularJS>karma start angular-scenario\karma.conf.js
[2013-06-17 15:14:49Z +349] (ERROR) config - Invalid config file!
[ReferenceError: module is not defined]
ReferenceError: module is not defined
    at eval (eval at parseConfigFile (C:\Users\o2\appData\Roaming\npm\node_modules\karma\lib\config.js:158:8)
    at Object.parseConfigFile (C:\Users\o2\appData\Roaming\npm\node_modules\karma\lib\config.js:177:17)
    at Object.exports.star (C:\Users\o2\appData\Roaming\npm\node_modules\karma\lib\server.js:1)
    at Object. (C:\Users\o2\appData\Roaming\npm\node_modules\karma\bin\karma:20:39)
    at Module._extensions..js (module.js:390:20)
    at Object.Module._extensions.(anonymous function) [as .node] (C:\Users\o2\appData\Roaming\npm\node_modules\karma\lib\server.js:1)
    at Module.load (module.js:343:32)
    at Function.Module._load (module.js:312:12)
    at Function.Module.runMain (module.js:497:10)
```

So what I think is happening is that because I run `npm install **` on the karma folder (the one I got from a GitHub clone), there are more modules in there than in the global karma (which I got when I installed karma using `npm install -g karma`)

At the moment there are 49 modules in the GitHub karma:



And 20 modules in the global karma install



So let's try running npm install on this folder

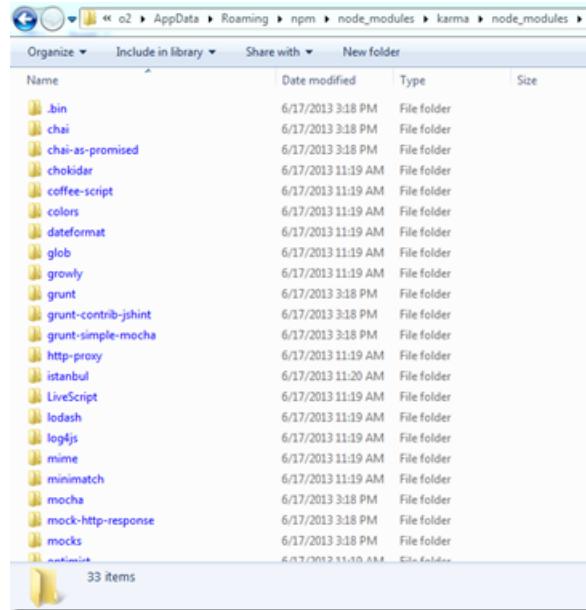
```
C:\Users\o2\AppData\Roaming\npm\node_modules\karma>npm install
npm http package.json download failed, reason: ECONNREFUSED. No repository field.
npm http GET package.json pause@0.1.22 No repository field.
npm http GET package.json pause@0.1.22 No repository field.
npm http GET https://registry.npmjs.org/mock-http-response
npm http GET https://registry.npmjs.org/timer-shim
npm http GET https://registry.npmjs.org/which
npm http GET https://registry.npmjs.org/which
npm http GET https://registry.npmjs.org/sinon
npm http GET https://registry.npmjs.org/chai
npm http GET https://registry.npmjs.org/nocks
npm http GET https://registry.npmjs.org/chai-as-promised
npm http GET https://registry.npmjs.org/nocha
npm http GET https://registry.npmjs.org/graceful-fs
npm http GET https://registry.npmjs.org/chai
npm http GET https://registry.npmjs.org/grunt
npm http 304 https://registry.npmjs.org/which
```

```

nock@http-response@0.1.1 node_modules\mock-http-response
└── which@0.5 node_modules\which
  └── sinon-chai@2.3.1 node_modules\sinon-chai
    ├── chai-as-promised@3.2.5 node_modules\chai-as-promised
    ├── grunt-simple-mocha@0.4.0 node_modules\grunt-simple-mocha
    └── mocks@0.0.11 node_modules\mocks
      └── chai@1.5.0 node_modules\chai
        └── timer-shim@0.2.3-1 node_modules\timer-shim
          └── linkedlist@1.0.1
            └── q@0.8.4
              └── q@0.8.4
    mocha@1.8.2 node_modules\mocha
      ├── growl@1.7.0
      ├── debug@0.7.2
      ├── errorhandler@0.6.1
      ├── diff@1.0.2
      ├── mkdirp@0.3.3
      ├── ms@0.3.1
      └── jsonfile@0.26.3 (mkdirp@0.3.0)
  grunt@0.4.1 node_modules\grunt
    ├── nock@0.5
    ├── async@0.1.22
    ├── eventemitter2@0.4.12
    ├── nomnom@1.0.1 (abbrev@1.0.4)
    ├── rimraf@2.0.3 (glob@3.2.11, graceful-fs@1.1.14)
    ├── coffee-script@1.3.3
    ├── underscore.string@2.2.0rc
    ├── jscolor@2.1.10
    ├── lodash@0.9.2
    ├── findup-sync@0.1.2 (lodash@0.1.0)
    ├── is-yaml@2.0.5 (esprima@1.0.3, argparse@0.1.15)
    └── sinon@1.6.0 node_modules\sinon
      └── buster-format@0.5.5 (cluster-core@0.6.4)
  grunt-contrib-jshint@0.3.8 node_modules\grunt-contrib-jshint
    └── jshint@1.1.0 (peakle@0.0.1, cli@0.4.4-2, underscore@1.4.4, shelljs@0.1.4, esprima@1.1.0-dev)

```

And now there are 33 modules installed:



but that still didn't work :(

At this stage I remembered reading something about installing the latest version of karma (globally), which is probably what I'm getting from the github clone, and that could explain the different number of modules.

So I executed **npm install -g karma@canary**

```

E:\_Code\Tests\AngularJS\angular-scenario>npm install -g karma@canary
npm http GET https://registry.npmjs.org/karma
npm http 304 https://registry.npmjs.org/karma
npm http GET https://registry.npmjs.org/_ai
npm http GET https://registry.npmjs.org/_chokidar
npm http GET https://registry.npmjs.org/_coffee-script
npm http GET https://registry.npmjs.org/_rimraf
npm http GET https://registry.npmjs.org/_glob
npm http GET https://registry.npmjs.org/_minimatch
npm http GET https://registry.npmjs.org/_socket.io
npm http GET https://registry.npmjs.org/_optimist
npm http GET https://registry.npmjs.org/http-proxy
npm http GET https://registry.npmjs.org/_q
npm http GET https://registry.npmjs.org/_pause/0.0.1
npm http GET https://registry.npmjs.org/_mime

```

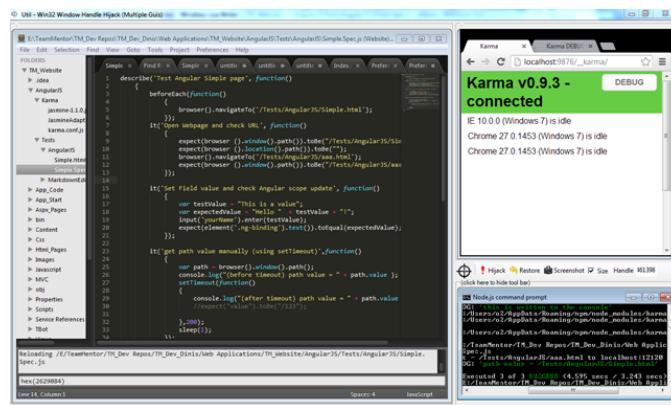
which installed ok:

```
karma@9_3 C:\Users\o2\AppData\Roaming\npm\node_modules\karma
└── pause@0.1.1
  ├── graceful-fs@1.2.2
  └── graceful@1.1.4
    ├── readable-stream@0.10.6
    └── inherits@1.2.0
      └── qs@0.9.6
        └── coffee-script@1.6.3
          ├── es5-shim@2.12.12
          └── underscore@1.3.1
        └── glob@3.1.21
          └── inherits@1.0.0
        └── optimist@0.3.5
          ├── wordwrap@0.0.2
          └── wordwrap@0.0.5
        └── log4js@0.6.6
          └── degauss@0.1.3
            └── server@1.1.4
          └── readable-stream@0.1.15
        └── http-proxy@0.10.2
          └── pkginfo@0.2.3
            └── util@0.1.7
        └── socket.io@0.9.16
          └── base64id@0.1.0
            └── policyfile@0.0.4
            └── redis@0.7.3
            └── socket.io-client@0.9.16
```

Now runnig the \*\*scenario B) \*\*case throws a different error:

```
E:\Code\Tests\AngularJS>karma start angular-scenario\karma.conf.js
DEBUG [config]: autoWatch set to false, because of singleRun
DEBUG [plugin]: Loading plugin karma-ng-scenario.
WARN [plugin]: Cannot find plugin "karma-ng-scenario".
Did you forget to install it?
npm install karma-ng-scenario --save-dev
C:\Users\o2\AppData\Roaming\npm\node_modules\karma\node_modules\di\lib\injector.js:9
  throw error`No provider for '' + name + ''`;
Error: No provider for 'framework:ng-scenario'! <Resolving: framework:ng-scenario>
  at Error (C:\Users\o2\AppData\Roaming\npm\node_modules\karma\node_modules\di\lib\in
```

It looks that we also need install the *karma-ng-scenario* module



And now it works :)

```
E:\Code\Tests\AngularJS>karma start angular-scenario\karma.conf.js
DEBUG [config]: autoWatch set to false, because of singleRun
DEBUG [plugin]: Loading plugin karma-ng-scenario.
INFO [karma]: Karma v0.9.3 server started at http://localhost:9876/_karma/
DEBUG [watcher]: Resolved files:
  C:/Users/o2/AppData/Roaming/npm/node_modules/karma-ng-scenario/lib/angular-scenario.js
  C:/Users/o2/AppData/Roaming/npm/node_modules/karma-ng-scenario/lib/angular-scenario/lib/adapter.js
  E:/Code\Tests\AngularJS\angular-scenario/e2eSpec.js
DEBUG [karma]: New browser has connected on socket 4-p7Xg16B9ux6y_kd2z
DEBUG [karma]: Connected on socket id 4-p7Xg16B9ux6y_kd2z
DEBUG [karma]: All browsers are ready, executing tests...
DEBUG [web server]: serving: C:/Users/o2/AppData/Roaming/npm/node_modules/karma/static/context.html
DEBUG [web server]: serving: C:/Users/o2/AppData/Roaming/npm/node_modules/karma-ng-scenario/lib/adapter.js
DEBUG [web server]: serving: C:/Users/o2/AppData/Roaming/npm/node_modules/karma-ng-scenario/lib/angular-scenario.js
DEBUG [proxy]: proxying request ~ index.html to localhost:8000
DEBUG [proxy]: proxying request ~ angular.min.js to localhost:8000
Chrome 27.0.1453 (Windows 7): Executed 1 of 1 SUCCESS (0.416 secs / 0.235 secs)
DEBUG [launcher]: Disconnecting all browsers
```

---

[Table of Contents](#) | [Code](#)

# **Appendix Post's Details**

This is the chronological order of the posts:

## **March 2013**

- Using Chrome inside a native VisualStudio pane (using Window Handle Hijacking)

## **April 2013**

- Submitting TM users to HubSpot via TBOT interface (using Angular JS)
- Hubspot current.js code includes JQuery on it

## **June 2013**

- AngularJS code editor using UI-Bootstrap and CodeMirror (done without using jQuery)
- Running KarmaJS's AngularJS example test/e2e/angular-scenario (on Chrome)
- Debugging a weird case of missing module in AngularJS and KarmaJS
- If AngularJS doesn't work on your O2 Platform IE scripts (the fix is to change browser compatibility mode)
- KarmaJS AngularJS Scenario Test Runner execution variations in IE 7,8,9 and 10 when using AngularJS
- A small AngularJS Jasmine test executed by KarmaJS
- Adding KarmaJS support to WebStorm to automagically run tests on file changes (and test UI with SublimeText, Chrome and Cmd.exe)
- When the best way to automate Chrome is to use ... Chrome (with examples on Google search, direct AngularJS scope manipulation and ChromeDriver javascript access)
- Using WebStorm with Chrome and ChromeDriver (to view KarmaJS execution results)

## **February 2014**

- Creating an Eclipse UI to run AngularJS e2e tests using Karma
- Trying out Firebase (Beta) hosting solution and good example of Firebase Security rules
- First PoC of sending TeamMentor's server-side request URLs to Firebase (and seeing it in realtime in an AngularJS page)
- Eclipse Groovy REPL script to sync a Browser with file changes (with recursive folder search via Java's WatchService)
- A really SIMPLE and clean AngularJS+Firebase example
- Using AngularJS in Eclipse, Part 1) The Basics
- Using AngularJS in Eclipse, Part 2) Add Some Control
- Using AngularJS in Eclipse, Part 3) Wire up a Backend
- Using AngularJS in Eclipse, Part 4) Create Components

## March 2014

- Eclipse Groovy script to remove the ‘busy’ image from the WebBrowser Editor
  - Programmatically changing an AngularJS scope variable and adding Firebug Lite to an AngularJs app
- 

[Table of Contents](#) | [Code](#)