2.6 Testing Routers 

This section will guide you to:

* Set up MongoDB on your local system

This lab has two subsections, namely:

* + 1. Writing a router test
    2. Pushing the code to GitHub repositories

**Step 2.6.1:** Writing a router test

* Open Visual Studio Code
* Open the karma-recipe folder in it
* Open the terminal and type **npm install sinon**
* Go to the spec folder
* Modify the **recipeModelSpec.js** file
* Write the following code in it:

*describe('A recipe', function() {*

*it('should exist', function() {*

*expect(app.Models.Recipe).toBeDefined();*

*});*

*it('should have certain defaults', function() {*

*var recipe = new app.Models.Recipe();*

*expect(recipe.get('name')).toEqual('Unnamed');*

*expect(recipe.get('rating')).toEqual(0);*

*expect(recipe.get('ingredients').length).toEqual(0);*

*expect(recipe.get('instructions').length).toEqual(0);*

*expect(recipe.get('dateAdded')).toBeDefined();*

*});*

*it('should identify if it has nuts', function() {*

*var noRestrictionsRecipe = new app.Models.Recipe(FIXTURES.recipes.noRestrictions);*

*var withNutsRecipe = new app.Models.Recipe(FIXTURES.recipes.withNuts);*

*expect(noRestrictionsRecipe.containsNuts()).toBeFalsy();*

*expect(withNutsRecipe.containsNuts()).toBeTruthy();*

*});*

*it('should identify if it has eggs', function() {*

*var noRestrictionsRecipe = new app.Models.Recipe(FIXTURES.recipes.noRestrictions);*

*var withEggsRecipe = new app.Models.Recipe(FIXTURES.recipes.withEggs);*

*expect(noRestrictionsRecipe.containsEggs()).toBeFalsy();*

*expect(withEggsRecipe.containsEggs()).toBeTruthy();*

*});*

*it('should identify if it is vegetarian', function() {*

*var noRestrictionsRecipe = new app.Models.Recipe(FIXTURES.recipes.noRestrictions);*

*var withMeat = new app.Models.Recipe(FIXTURES.recipes.withMeat);*

*expect(noRestrictionsRecipe.isVegetarian()).toBeTruthy();*

*expect(withMeat.isVegetarian()).toBeFalsy();*

*});*

*it('should delegate to checkForIngredientType', function() {*

*var withEggsRecipe = new app.Models.Recipe(FIXTURES.recipes.withEggs);*

*var mySpy = spyOn(withEggsRecipe, 'checkForIngredientType').and.callThrough();*

*var hasEggs = withEggsRecipe.containsEggs();*

*expect(mySpy).toHaveBeenCalled();*

*expect(mySpy).toHaveBeenCalledWith('isEggs');*

*expect(hasEggs).toBeTruthy();*

*expect(mySpy.calls.count()).toBe(1);*

*});*

*});*

* Save the file
* Go to the src folder
* Modify the **recipeModel.js** file
* Write the following code in it:

*var app = window.app || {};*

*app.Models = app.Models || {};*

*app.Models.Recipe = Backbone.Model.extend({*

*defaults: function() {*

*return {*

*name: 'Unnamed',*

*rating: 0,*

*ingredients: [],*

*instructions: [],*

*dateAdded: Date.now()*

*};*

*},*

*checkForIngredientType: function(propertyName) {*

*var ingredients = this.get('ingredients');*

*for (var i = 0, l = ingredients.length; i < l; i ++) {*

*var ingredient = ingredients[i];*

*if(ingredient[propertyName]) {*

*return true;*

*}*

*}*

*return false;*

*},*

*containsNuts: function() {*

*return this.checkForIngredientType('isNut');*

*},*

*containsEggs: function() {*

*return this.checkForIngredientType('isEggs');*

*},*

*isVegetarian: function() {*

*return !this.checkForIngredientType('isMeat');*

*}*

*});*

* Save the file
* Go to the spec folder
* Create a new file named **recipeRouterSpec.js**
* Write the following code in it:

*describe('The application router', function() {*

*var router;*

*afterEach(function() {*

*Backbone.history.stop();*

*});*

*var routerSetup = function() {*

*router = new app.Routers.RecipeRouter();*

*Backbone.history.start();*

*};*

*it('should let us navigate to a recipe by id', function() {*

*var routerSpy = spyOn(app.Routers.RecipeRouter.prototype, 'openRecipe');*

*routerSetup();*

*router.navigate('recipes/3', { trigger: true });*

*expect(routerSpy).toHaveBeenCalled();*

*expect(routerSpy).toHaveBeenCalledWith('3', null);*

*});*

*var spyAndSetup = function(spyConfig) {*

*for(var method in spyConfig) {*

*spyOn(spyConfig[method], method);*

*}*

*routerSetup();*

*};*

*it('should let us create a variation based on another recipe', function() {*

*spyAndSetup({*

*'createVariation': app.Routers.RecipeRouter.prototype*

*});*

*router.navigate('recipe/3/variation', {trigger: true});*

*expect(router.createVariation).toHaveBeenCalled();*

*expect(router.createVariation).toHaveBeenCalledWith('3', null);*

*});*

*});*

* Save the file
* Go to the src folder
* Create a new file named **recipeRouter.js**
* Write the following code in it:

*var app = window.app || {};*

*app.Routers = app.Routers || {};*

*app.Routers.RecipeRouter = Backbone.Router.extend({*

*routes: {*

*'recipes/:id': 'openRecipe',*

*'recipe/:id/variation': 'createVariation'*

*},*

*openRecipe: function() {*

*},*

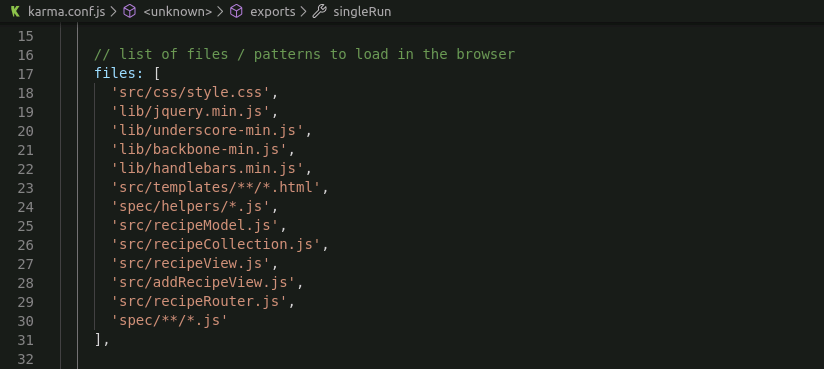
*createVariation: function () {*

*}*

*});*

* Save the file

* Open karma.conf.js using Notepad
* Modify the file as shown in the image
* Save the file

****

* Open a terminal
* Type karma test
* The tests will pass

**Step 2.6.2:** Pushing the code to GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add . 

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master