תוכן עניינים

INSTALLING NC-BLOGAPP **2**

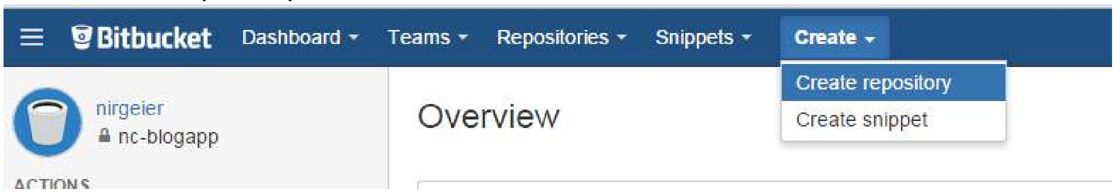
PREPARING THE PROJECT 2
ADDING THE SOURCE CODE 3
INSTALLING NODE DEPENDENCIES 3
TESTING THE PROJECT 4
ADDING OUR CODE 5
Adding angular support *5*
Add ng-app *6*
Setting the routes *6*
Adding the controllers for the route 6
Displaying the content of the templates in the browser 7

Installing nc-blogapp

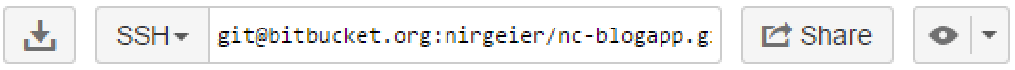
- Register to bitbucket

Preparing the project

- Create a new repository

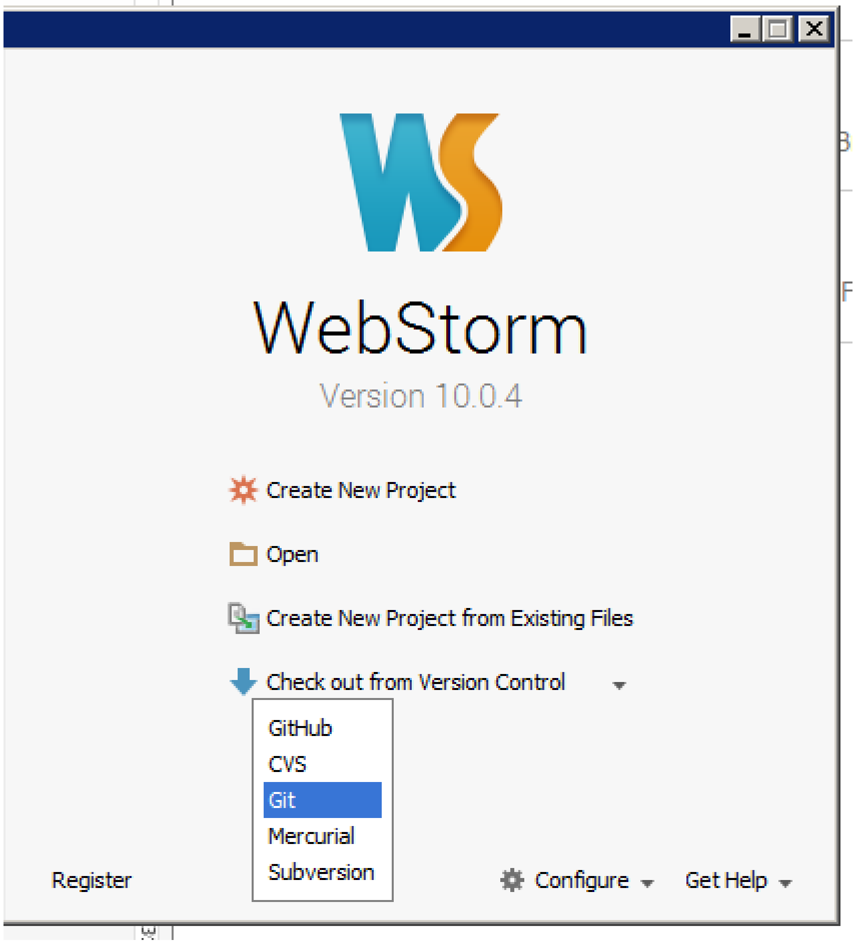


- Once the repository is created copy the clone url from the top left side of the screen

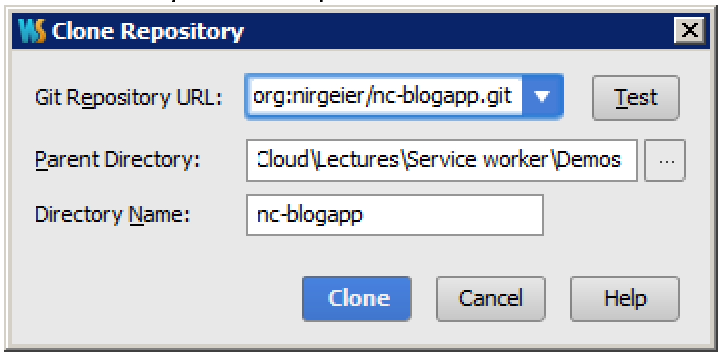


```
git@bitbucket.org:nirgeier/nc-blogapp.g
```

- Open web storm and in the opening screen select:



- Paste the url you have copied from bitbucket



Git Repository URL: org:nirgeier/nc-blogapp.git
Parent Directory: Cloud\Lectures\Service worker\Demos
Directory Name: nc-blogapp

- Once the repository is cloned open the project

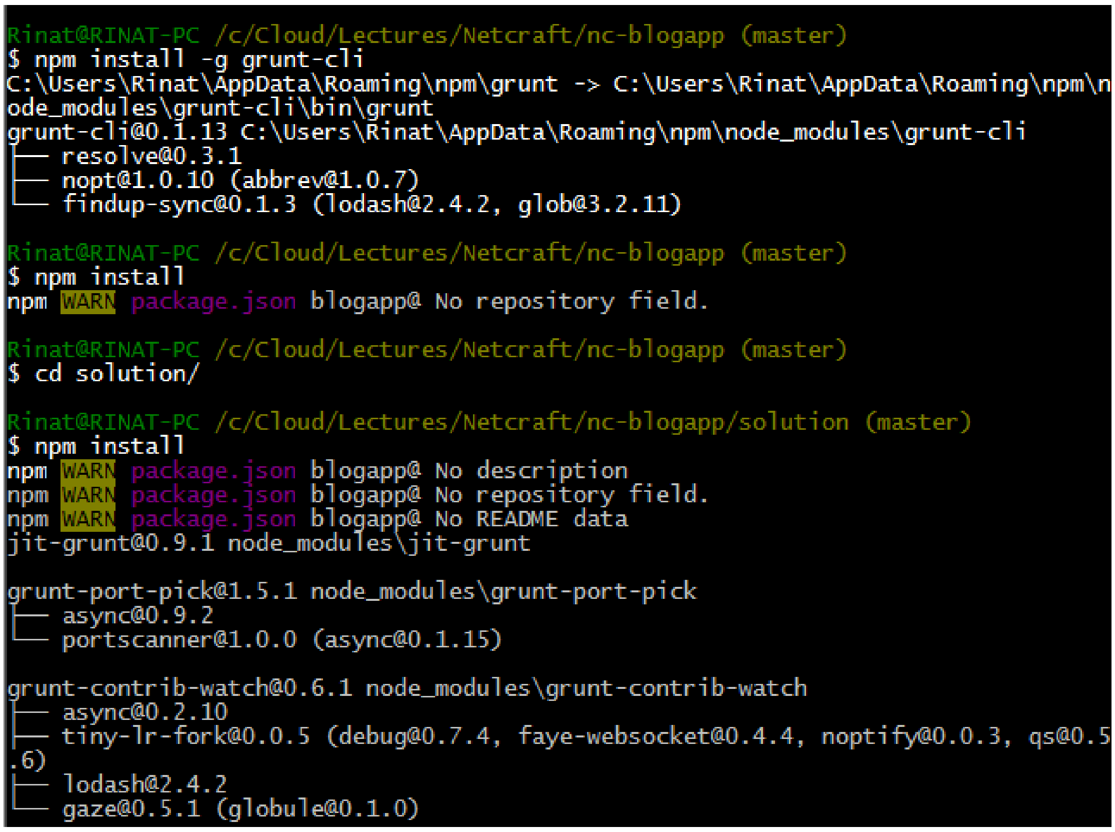
Adding the source code

- Download the zipped app
- Drag the content of the zip to webstorm or extract it to the project folder
- Add the all files to git (Using webstorm or from command line), commit and push

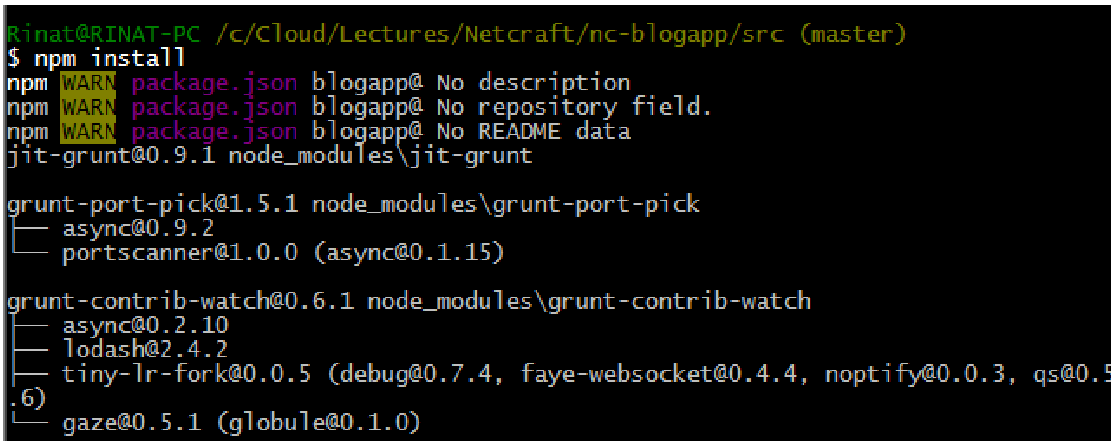
Installing node dependencies

- Open command line or gitbash in the project root folder
  - Run:

    ```
    npm install –g grunt-cli
    npm install
    cd solution
    npminstall
    cd ../src
    npm install
    ```



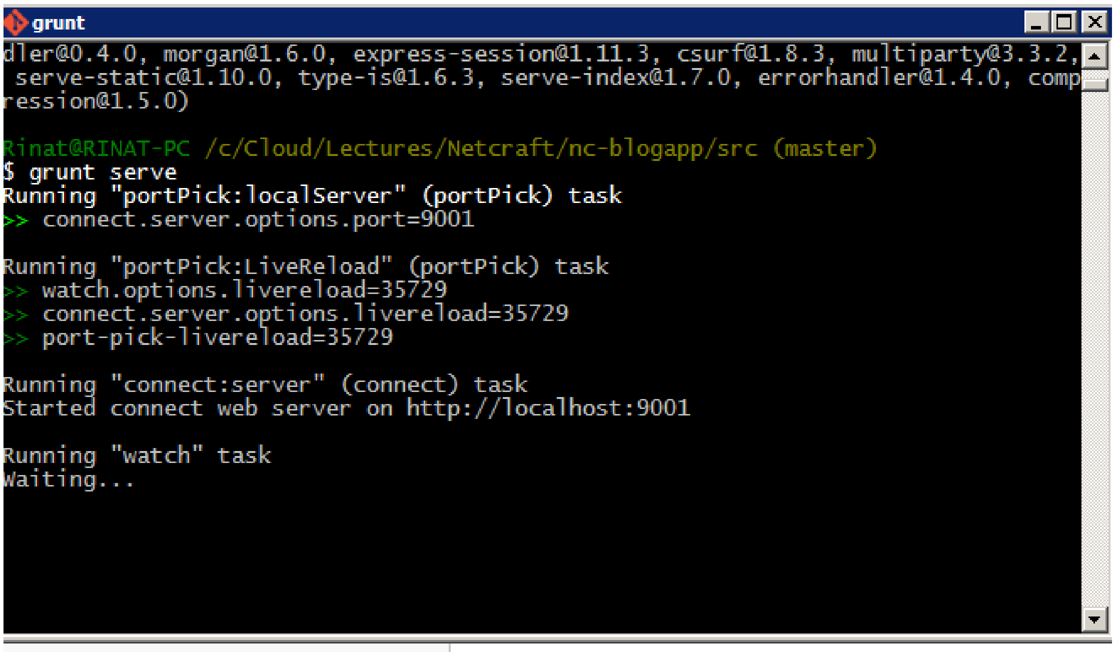
```
Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp (master)
$ npm install -g grunt-cli
C:\Users\Rinat\AppData\Roaming\npm\grunt -> C:\Users\Rinat\AppData\Roaming\npm\n
ode_modules\grunt-cli\bin\grunt
grunt-cli@0.1.13 C:\Users\Rinat\AppData\Roaming\npm\node_modules\grunt-cli
├── resolve@0.3.1
├── nopt@1.0.10 (abbrev@1.0.7)
└── findup-sync@0.1.3 (lodash@2.4.2, glob@3.2.11)

Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp (master)
$ npm install
npm WARN package.json blogapp@ No repository field.

Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp (master)
$ cd solution/

Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp/solution (master)
$ npm install
npm WARN package.json blogapp@ No description
npm WARN package.json blogapp@ No repository field.
npm WARN package.json blogapp@ No README data
jit-grunt@0.9.1 node_modules\jit-grunt

grunt-port-pick@1.5.1 node_modules\grunt-port-pick
├── async@0.9.2
└── portscanner@1.0.0 (async@0.1.15)

grunt-contrib-watch@0.6.1 node_modules\grunt-contrib-watch
├── async@0.2.10
├── tiny-lr-fork@0.0.5 (debug@0.7.4, faye-websocket@0.4.4, noptify@0.0.3, qs@0.5
.6)
├── lodash@2.4.2
└── gaze@0.5.1 (globule@0.1.0)
```



```
Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp/src (master)
$ npm install
npm WARN package.json blogapp@ No description
npm WARN package.json blogapp@ No repository field.
npm WARN package.json blogapp@ No README data
jit-grunt@0.9.1 node_modules\jit-grunt

grunt-port-pick@1.5.1 node_modules\grunt-port-pick
├── async@0.9.2
└── portscanner@1.0.0 (async@0.1.15)

grunt-contrib-watch@0.6.1 node_modules\grunt-contrib-watch
├── async@0.2.10
├── lodash@2.4.2
├── tiny-lr-fork@0.0.5 (debug@0.7.4, faye-websocket@0.4.4, noptify@0.0.3, qs@0.5
.6)
└── gaze@0.5.1 (globule@0.1.0)
```

Testing the project

- Go to the <root>/src folder
- Run: *grunt serve*



```
dler@0.4.0, morgan@1.6.0, express-session@1.11.3, csurf@1.8.3, multiparty@3.3.2,
 serve-static@1.10.0, type-is@1.6.3, serve-index@1.7.0, errorhandler@1.4.0, comp
ression@1.5.0)

Rinat@RINAT-PC /c/Cloud/Lectures/Netcraft/nc-blogapp/src (master)
$ grunt serve
Running "portPick:localServer" (portPick) task
>> connect.server.options.port=9001

Running "portPick:LiveReload" (portPick) task
>> watch.options.livereload=35729
>> connect.server.options.livereload=35729
>> port-pick-livereload=35729

Running "connect:server" (connect) task
Started connect web server on http://localhost:9001

Running "watch" task
Waiting...
```

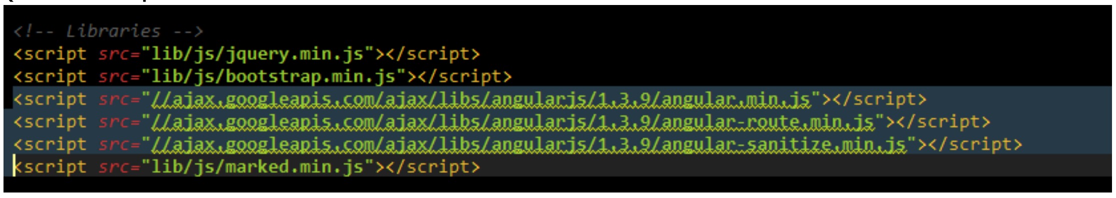
- Now the browser should be opened with the index file
  - Note: The grunt is starting an internal node server and opens the index.html under the src folder.

Adding our code

- Now that we have the project running and ready our next step is to start adding our code to the project.

Adding angular support

- Open the index.htm under the solutions folder and copy the required angular scripts:
  (all the scripts under the libraries section



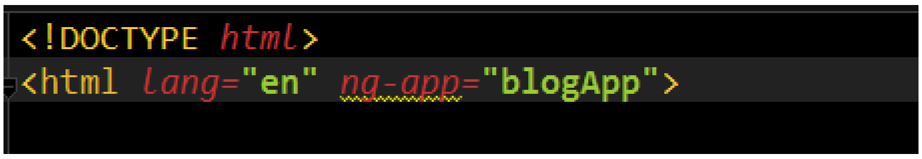
```
<!-- Libraries -->
<script src="lib/js/jquery.min.js"></script>
<script src="lib/js/bootstrap.min.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular.min.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular-route.min.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular-sanitize.min.js"></script>
<script src="lib/js/marked.min.js"></script>
```

- Add those script to the index file under the src folder (src/index.html)

Now we are all set to go and we can start adding our angular code

Add ng-app

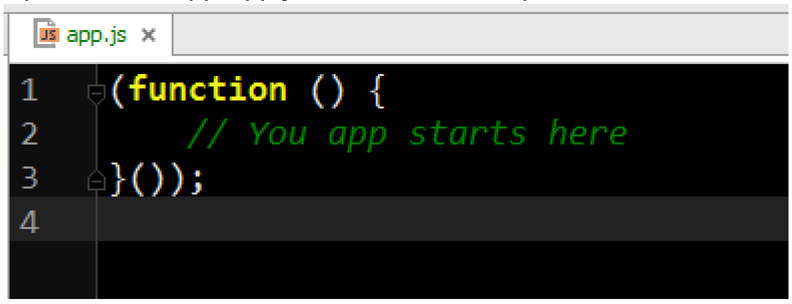
- Add the ng-app to your html tag (src/index.html)



```
<!DOCTYPE html>
<html lang="en" ng-app="blogApp">
```

Writing the angular code

- Open the src/app/app.js and start to add your code



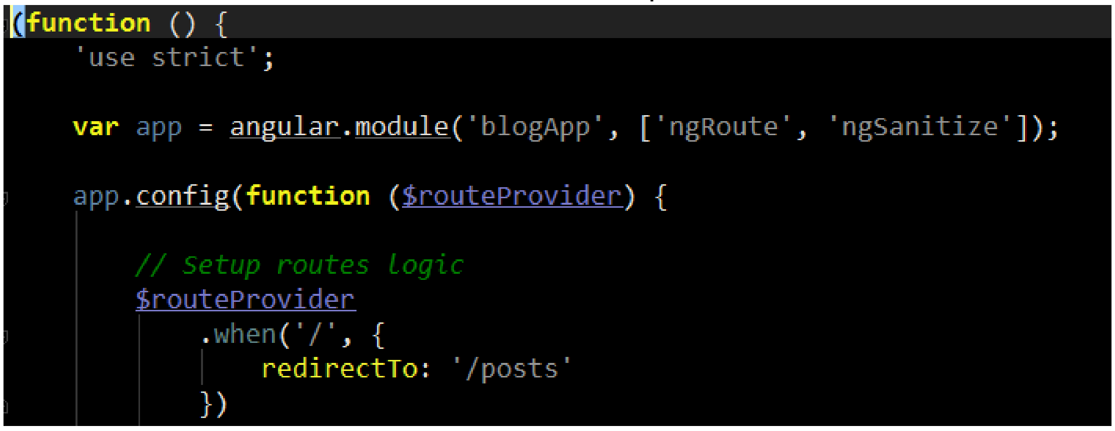
app.js

```
(function () {
    // You app starts here
}());
```

Setting the routes

In the app.js file start to

- Create the app module with 2 dependencies (those dependencies are the ones we added the to the index.html with the script file

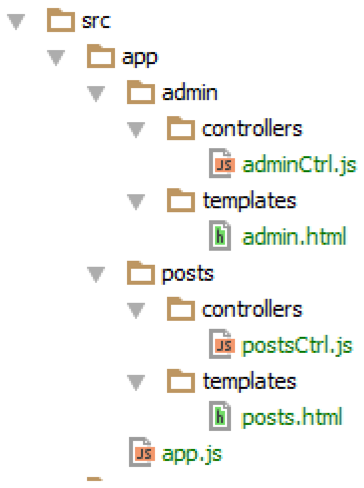


```
(function () {
    'use strict';

    var app = angular.module('blogApp', ['ngRoute', 'ngSanitize']);

    app.config(function ($routeProvider) {

        // Setup routes logic
        $routeProvider
            .when('/', {
                redirectTo: '/posts'
            })
```

Adding the controllers for the route

- Write 2 empty controllers for the moment. One for admin and one for posts
- Write 2 html templates for posts and admin

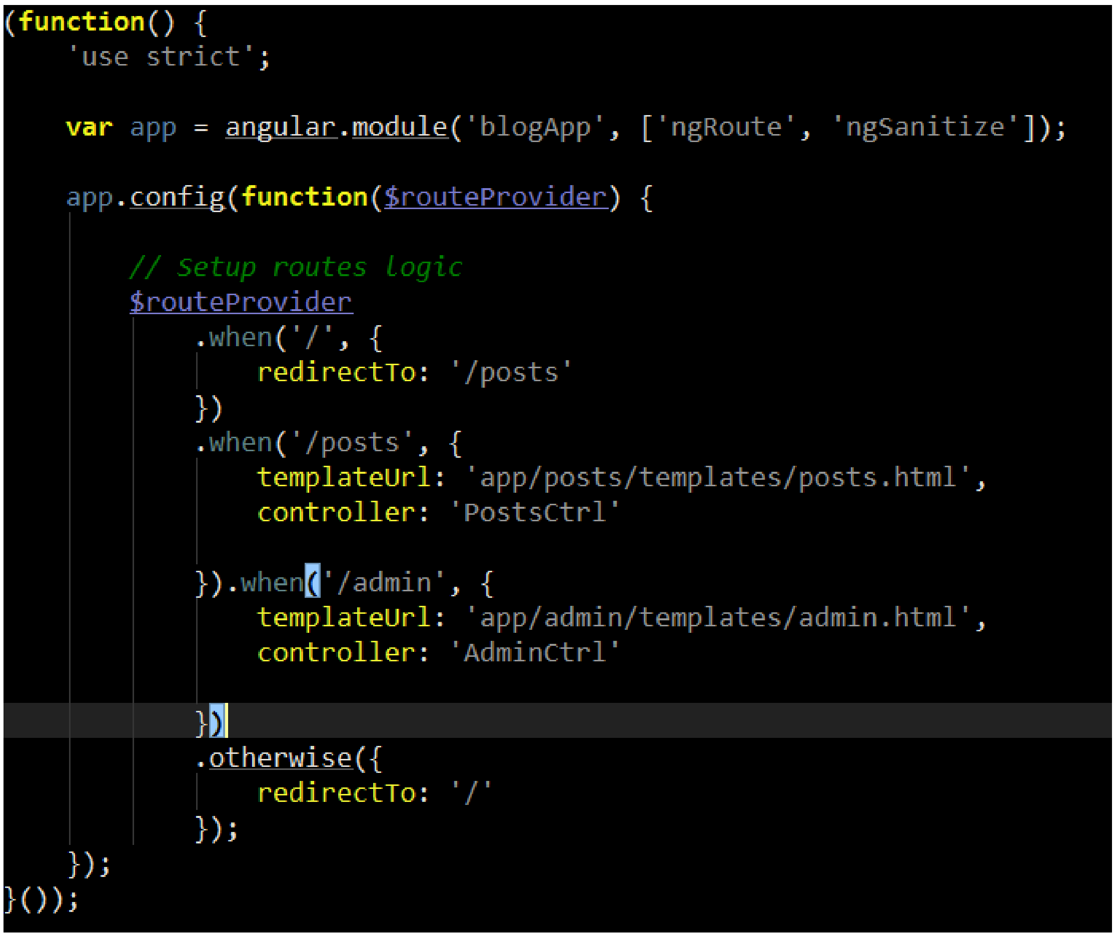
- Set the name of the controller + templates in the route (inside your app.js)



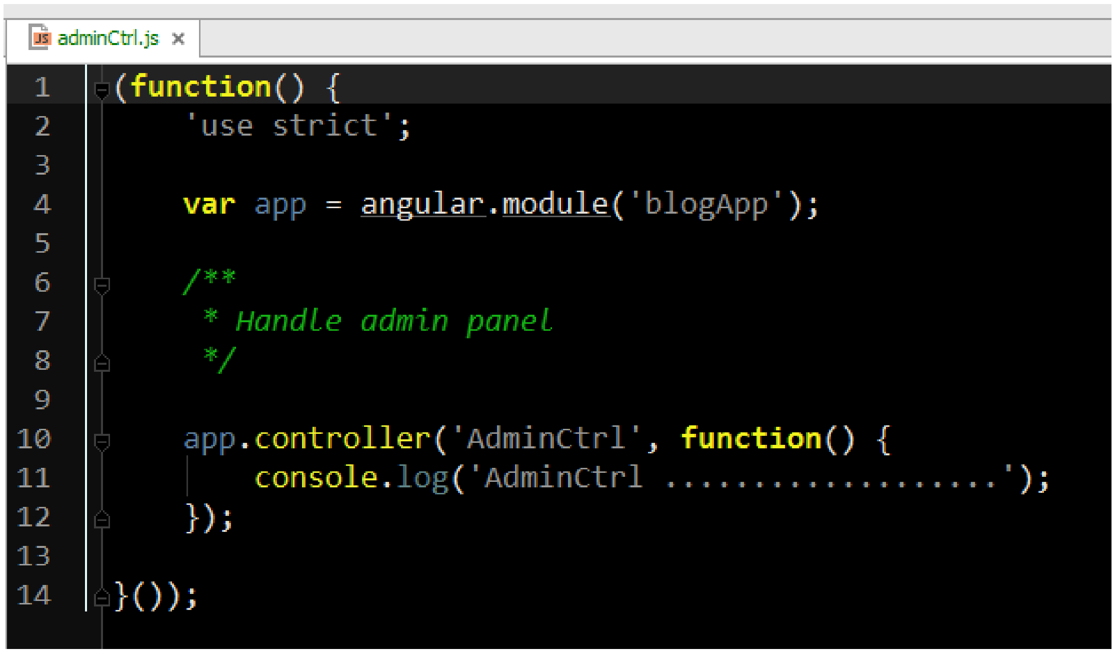
```
src
  app
    admin
      controllers
        adminCtrl.js
      templates
        admin.html
    posts
      controllers
        postsCtrl.js
      templates
        posts.html
    app.js
```

Displaying the content of the templates in the browser

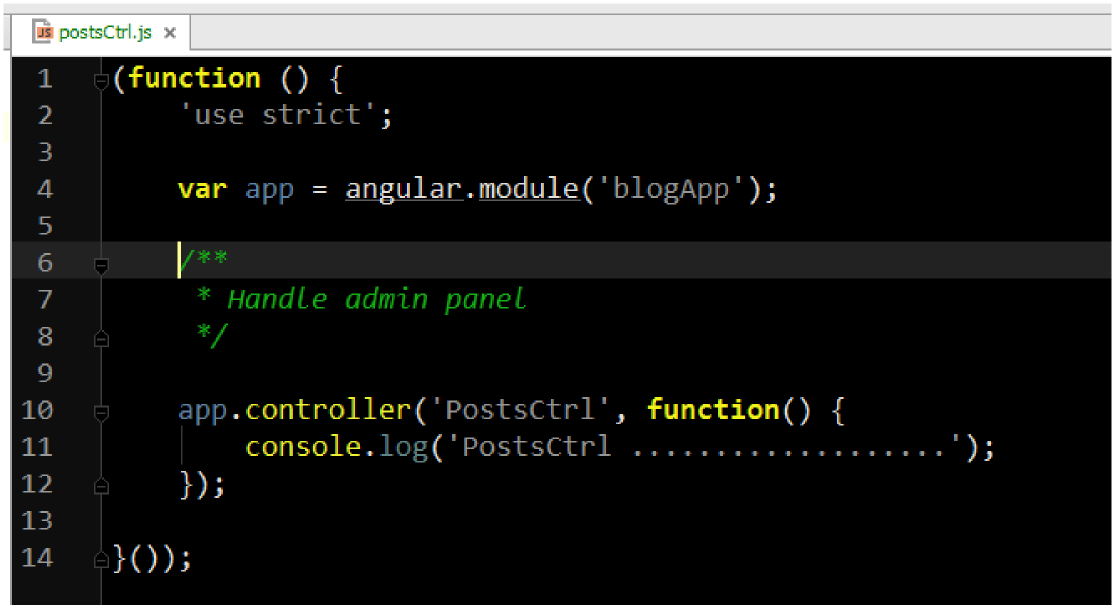
- Now that we all set to go and we have the route, controllers and templates we want to display the content in our application.
- Here is the code that we have so far:
  - app.js



```
(function() {
    'use strict';

    var app = angular.module('blogApp', ['ngRoute', 'ngSanitize']);

    app.config(function($routeProvider) {

        // Setup routes logic
        $routeProvider
            .when('/', {
                redirectTo: '/posts'
            })
            .when('/posts', {
                templateUrl: 'app/posts/templates/posts.html',
                controller: 'PostsCtrl'

            }).when('/admin', {
                templateUrl: 'app/admin/templates/admin.html',
                controller: 'AdminCtrl'

            })
            .otherwise({
                redirectTo: '/'
            });
    });
}());
```

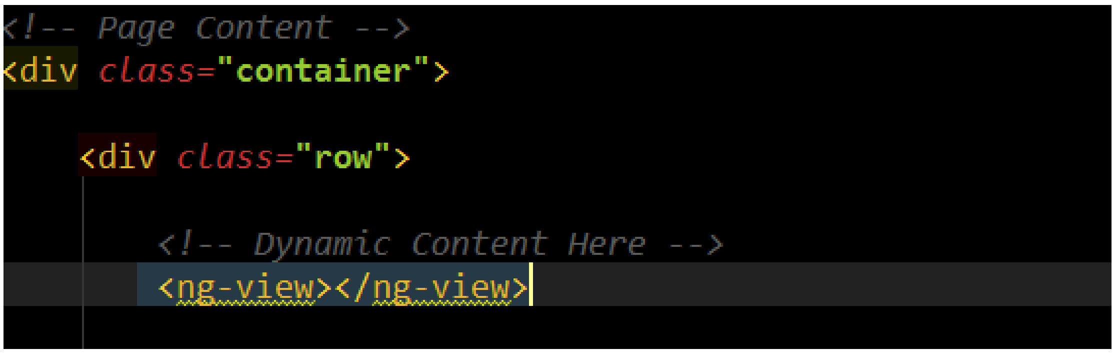


```
(function() {
    'use strict';

    var app = angular.module('blogApp');

    /**
     * Handle admin panel
     */

    app.controller('AdminCtrl', function() {
        console.log('AdminCtrl ...................');
    });

}());
```



```
(function () {
    'use strict';

    var app = angular.module('blogApp');

    /**
     * Handle admin panel
     */

    app.controller('PostsCtrl', function() {
        console.log('PostsCtrl ..................');
    });

}());
```

- Now we need to add the placeholder for displaying the content in the application. This is done by adding the following angular code to our inde.html
    - **<ng-view></ng-view>**



```
<!-- Page Content -->
<div class="container">

    <div class="row">

        <!-- Dynamic Content Here -->
        <ng-view></ng-view>
```

- Add any content to your html templates and click on the navigations url.
- The page will display the content of the appropriate template.