Angular Server - Restangular

In today's lesson we are going to learn about server communication, specifically we are going to use a popular angular service called **Restangular** to help us do so.
On the server side we have our todo application **REST** server located at the **URL:** **https://nztodo.herokuapp.com/**
Our server exposes the following CRUD api for our **task** model:

GET: /api/task/?format=json - will get all the tasks
GET: /api/task/<id>/?format=json - will get a single task
POST: /api/task/ - will create a new task
PUT: /api/task/<id>/?format=json - will update a single task
DELETE: /api/task/<id>/ - will delete a single task

The server returns a task model which looks like this:

```
{
   id: number
   title: string,
   description: string,
   group: string,
   date: Date
}
```

Bootstrapping our application

Let's start a new Angular application.
Create a root folder for you application and cd into that directory.
init bower in that directory:
> bower init

answer all bower questions or just press enter to choose the default values.
Now use bower to install **angular**
> bower install angular --save

Create a **src** folder to hold all our **js** code and in that folder create the file **app.module.js** to hold our **angular module** init code.
Place the following code in that file:

```
angular.module('RestDemo', []);
```

Our application will have a single controller that will be called **MainController**, in the **src** folder create a folder called **controllers** and in it create a file called **main.controller.js** with the following code:

```
angular.module('RestDemo').controller('MainController', function () {
  var self = this;

  self.greeting = 'Hello Rest Demo';
});
```

Our controller simply attach a greeting to the **scope** .

Let's create a template to display our greeting. Create a folder in the **src** folder called **templates** and in that folder place a file called **main.template.html** with the following code:

```
<h1>
  {{ctrl.greeting}}
</h1>
```

Now in the root folder of our app, create our **index.html** with the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Rest Demo</title>
</head>
<body ng-app="RestDemo">
  <div
     ng-controller="MainController as ctrl"
     ng-include="'src/templates/main.template.html'">

  </div>

  <script src="bower_components/angular/angular.js"></script>
  <script src="src/app.module.js"></script>
  <script src="src/controllers/main.controller.js"></script>
</body>
</html>
```

Activate your app and verify that the greeting message appears.
We now have a simple angular application with a root module, a controller, and a template for that controller, and we are attaching a greeting from the controller to our template.

That will be a good starting point for our journey to cover the **restangular service**.

Installing Restangular

Let's start by installing the **Restangular service.**
In the root folder type in the terminal:
> bower install restangular --save

Now we need to include the service in our **index.html** so add the following line at the bottom of the **index.html** just after the **angular** script and before the definition of the **module**:

```
<script src="bower_components/underscore/underscore.js"></script>
<script src="bower_components/restangular/dist/restangular.js"></script>
```

Now we need to include the **Restangular** module in our **module definition** so the **DI** will be able to inject the service.
In the **app.module.js** change it to look like this:

```
angular.module('RestDemo', ['restangular']);
```

We can now inject the **Restangular** service in our controller and start using it,

Using Restangular in our Controller

Let's use **Restangular** to create a call to our server to get all the tasks.
Modify the **main.controller.js**

```
angular.module('RestDemo').controller('MainController', ['Restangular', function
(Restangular) {
  var self = this;

  self.tasks = Restangular.allUrl('tasks',
'https://nztodo.herokuapp.com/api/task/?format=json')
  .getList().$object;


  self.greeting = 'Hello Rest Demo';
}]);
```

Notice that we injected the service **Restangular** and called the **allUrl** method which allows us to specify the **URL** where we are making the call to. We are then calling the **getList** which creates

the request and returns a **Promise** we can subscribe to using **then**, the **then** function will get the data from the server and we can then place that data to our scope, another way which is what we are doing here is to avoid the callback and simply get the **$object** which will have an empty array and will be filled when the response returns.
Now in the template **main.template.html** change it to iterate on the **tasks** list and print the title of each task.

```
<ul>
  <li ng-repeat="task in ctrl.tasks">
     {{task.title}}
  </li>
</ul>
```

Activate the application and you should see a call to the server and the list of tasks is displayed.

Configuring Restangular

Looking at our above request with **Restangular** there is some annoying things that we done there. For example we are using the absolute **URL** of our server and also adding **format=json** query param.
Let's modify the configuration of **Restangular** so we can stop doing that on each request.

Modify **app.module.js** to look like this:

```
angular.module('RestDemo', ['restangular'])
  .config(['RestangularProvider', function(RestangularProvider){
     RestangularProvider.setBaseUrl('https://nztodo.herokuapp.com/api');
     RestangularProvider.setDefaultRequestParams('get', {format: 'json'});
     RestangularProvider.setRequestSuffix('/');
  }]);
```

Now we can change our **main.controller.js** to this:

```
angular.module('RestDemo').controller('MainController', ['Restangular', function
(Restangular) {
  var self = this;

  self.tasks = Restangular.all('task').getList().$object;

  self.greeting = 'Hello Rest Demo';
}]);
```

Without writing long URL's and without callbacks we are simply calling **Restangular.all** passing the required resource and calling **getList** and by accessing **$object** we are getting an array that will be filled with the data we need.

OOP - Writing a Model and a Service for Task

Let's try to aim to be a bit more object oriented, we are going to create a **factory** for our **Task** class and also create a **service** for our server interaction with the task api.
In the **src** folder create a folder called **services**. Inside the **services** folder create another folder called **Task.**
In the **Task** folder create a file called **task.model.js** with the following code:

```
angular.module('RestDemo')
  .factory('Task', function(){

    return function Task(taskJson){
      var _id = taskJson.id || -1;
      var _title = taskJson.title || '';
      var _description = taskJson.description || '';
      var _group = taskJson.group || '';
      var _when = new Date(taskJson.when);

      this.getTitle = function getTitle(){
        return _title;
      }

    }

  });
```

We use **angular factory** to create our **Task class** and we are hiding all the properties of the class in private variables and exposing only the title through a getter called **getTitle.**
We are creating here encapsulation for our **Task** class.

Now let's create the **service** that will interact with our server, it's better to do the server interaction in the model part of our MVC application and not in the controller. Create a file called **task.service.js** in the **Task** folder with the following code:

```
angular.module('RestDemo')
  .service('TaskService', ['Restangular', 'Task', function(Restangular, Task){

    /**
```

```
     * Set a restangular that will return Task instances array
     */
    var _restangular = Restangular.withConfig(function(RestangularConfigurer) {
        RestangularConfigurer.setResponseExtractor(function(response){
            var newResponse = response;
            if (angular.isArray(response)) {
                angular.forEach(newResponse, function(value, key) {
                  newResponse[key] = new Task(value);
                });
            } else {
                newResponse = new Task(response);
            }
            return newResponse;
        });
    });

    this.getTasks = function getTasks(){
        return _restangular.all('task').getList().$object;
    }

  }]);
```

Looks much more elegant now. We are configuring a **Restangular** for our service that will intercept the response and return instances of our **Task** instead of interacting directly with server response.
Now to use our service in the **main.controller.js**

```
angular.module('RestDemo').controller('MainController', ['TaskService', function
(TaskService) {
  var self = this;

  self.tasks = TaskService.getTasks();

}]);
```

And also in the template we have to interact with a **Task** instance and not directly with a server response. so change your **main.template.html:**

```
<ul>
  <li ng-repeat="task in ctrl.tasks">
    {{task.getTitle()}}
  </li>
</ul>
```

CRUD

Let’s finish