Communicating with RESTful Services

As already discussed, $http is a low level API for server side communication used to communicate with the server. In real world applications, we have to deal with data objects which represent an entity in the software ecosystem. In a normal scenario, we have to write codebase for common functionality like create, read, update, delete of these objects using $http service.

myAppModule.factory('EmployeeService', ['$http', function($http) {

var url = 'rest/employee';

return {

get: function(empID) {

return $http.get(url + '/' + empId);

},

save: function(employeeObj) {

var url = employeeObj.id ? url + '/' + employeeObj.id : url;

return $http.post(url, employeeObj);

},

query: function(empID) {

return $http.get(url + '/' + empID.id + '/reportees');

},

calculateSalary: function(employee) {

return $http.post(url + '/' + employee.id, employee, {params: {admin: true}});

}};

}

]);

In order to address this repetition of tasks, Angular provides built in support to create a service using $resourceobject. This object provides common functionalities like:

* Get
* Save
* Query
* Remove
* Delete

out of the box, without even writing a single line. It also provides the functionality to create new methods in the service. The following code snippet explains the concept:

*// Service to fetch EmployeeListService*

angularModule.factory("EmployeeListService", ['$resource', function($resource){

return $resource('rest/employee/:empId',

{empId: '@id'},

{calculateSalary: {method: 'POST', params: {charge: true}, isArray: false}});

}]) ;

Now the EmployeeListService Service has the following methods out of the box without writing a single line of code:

* EmployeeListService.get()
* EmployeeListService.save
* EmployeeListService.save
* EmployeeListService.query()
* EmployeeListService.remove()
* EmployeeListService.delete()

Custom Methods

In addition to the above methods, $resource service provides us the functionality to create new methods. This helps us to make custom RESTful calls the server. The following sample shows how to define a custom method in$resource service:

*// Service to fetch Reportees*

angularModule.factory("EmployeeReporteeService", ['$resource', function($resource){

return $resource('rest/employee/:empId/reportees',

{empId: '@id'},

*/\*\**

*\* Custom method to find contact numbers.*

*\**

*\* Method type is GET*

*\* Additional Query parameter admin=true is sent*

*\* The expected return type is an Array*

*\*/*

{contactNumbers: {method: 'GET', params: {admin: true}, isArray: true}});

}]);

How to Use Include $resource Object?

Let’s have a look at how to include this $resource service in codebase:

*//Step 1 : $resource is an optional module in Angular defined in a separate file,*

*//so we have to include its JS file in our HTML file.*

<script src=” angular-resource.js”></script>

// Step 2: Include ngResource dependency in the module where we want to use it. This code will be controller.js

angular.module(‘samplemodule’, [‘ngResource’]);

// Step 3: Then inject the $resource in the service where you want to use it.

myAppModule.factory('EmployeeService', ['$resource', function($resource) {

The $resource service accepts 1 required parameter and 2 optional parameters:

* URL; required: The URL where the resource is available
* Query params: Parameters to be passed with URL additional functions: Additional methods we want to add to the service.