Communicating with $http

In Angular, the basic building block for communication with server is $http service. The $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequestobject.

This service provides basic functionalities like:

* GET
* POST
* HEAD
* DELETE
* PUT
* JSONP

Angular API uses the Promise interface based approach for server side communication. This ensures that all the server related calls are non-blocking/asynchronous calls, which means that the response from the server is returned at some time in future. The Promise interface ensures that the response will be handled accordingly as and when it will arrive. There are two method names, success and error for handling the response from the server.

$http.get("empMgr", {params: {op: 'get'}})

.success(function(data, status, headers, config) {

$scope.employee = data; *// Sets the employee object in $scope*

})

.error(function(data, status, headers, config) {

alert("failure");

});

};

In the above example, an AJAX GET request is fired to find the details of an employee with id as 100. This call doesn’t wait for the response from the server. When the response from the server is received sometime in future, appropriate function success or error is called depending on the response from the server.

$http.post

In case we need to submit a data to the server using POST method, we can use $http.post API. Following is an example of posting the data to the server.

var dataToPost = {firstName: "Allen", lastName: "John"}; */\* PostData\*/*

var queryParams = {params: {op: 'saveEmployee'}};*/\* Query Parameters\*/*

$http.post("empMgr" */\* URL \*/*, dataToPost, queryParams)

.success(function(serverResponse, status, headers, config) {

*// Updating the $scope postresponse variable to update theview*

$scope.postresponse = serverResponse.data.firstName + " " + serverResponse.data.lastName;

}).error(function(serverResponse, status, headers, config) {

alert("failure");

}

);

The above example POSTs the employee data to the server. This call too is asynchronous. When the server is done with handling the save request, the response returns and the relevant method success/error is invoked.

**Handling POST in Angular**

Please note that $http.post of Angular is different from jQuery.post. Angular posts the data as JSON in the http body with content-type set as application/json, where as jQuery posts the values with content-type as 'application/x-www-form-urlencoded;charset=utf-8' so we need to put some codebase at the server to parse the http body to fetch to request body or we need to configure the Angular codebase to send the request as key value pair.

Configuring the $http Service Request

Although default implementation of $http.get,$http.post etc. is good enough to serve most of the purpose, there may be some specific use cases where we need to customize the default APIs. For example, we may have to set some custom headers, transform request and response, set custom timeout, enable/disable cache and set the response type.

$http service provides an inbuilt mechanism to customize the http requests using the “config” object. All the APIs of $http.X take the last parameter as config object to customize the requests.

For example, we can set the authorization token in HTTP header for get request using the header property ofconfig object.

$http.get("empMgr",

{

params: {op: 'get'},

headers: {"AUTH\_TOKEN": "1234567890"} *// Custom headers put in.*

})

.success(function(data, status, headers, config) {

$scope.employee = data; *// Sets the employee object in $scope*

})

.error(function(data, status, headers, config) {

alert("failure");

});

};

});

$http service can be invoked by passing only the config object too. The config object supports the following properties:

The supported options are :

$http({

method: string, // GET, POST, PUT, HEAD etc

url: string, // URL of resource being requested

params: object, // Query parameters, if not string goes as JSON string

data: string or object, // Data to be sent as request message data

headers: object, // Map representing the HTTP header

cache: boolean or Cache object, // Boolean to turn cache on or off

timeout: number, // Timeout to wait for response from server

withCredentials: boolean

});