How to Read and Create SharePoint List Items by AngularJS

Summary: Objective of this article is to help who is new to Angular development in SharePoint using SharePoint REST API.

Code demo will help you, to explore angular $http get and post methods using controller to read and create list items by REST service, which is simplified standard XMLHttpRequest service in angular framework.

Prerequisites: html, javascript, ajax calls to post data and get data, json, basic SharePoint development using javascript CSOM (Client side object modeling)

Libraries required to execute this demo code is angular.min.js or angular.js and create appScript.js file using below mentioned javascript code.

Steps to implement below code in SharePoint:

1. Create a page in SharePoint as index.aspx or name the page you wise
2. Upload the angular.min.js in SiteAssets library or reside some where you wise
3. Create a new js file as appScript.js in SiteAssets library
4. Edit index.aspx in SharePoint designer
5. Go to end of the line in edited page
6. Copy and paste the below mentioned html markup at before </asp:content> tag
7. Change the js src if it is different location and highlighted in yellow color for more clarity see the blow highlighted markup

<style>

#taskView {border-collapse: collapse;width: 100%;}

#taskView td, #taskView th {border: 1px solid black; padding: 5px;}

.action-container {width: 240px;text-align: right;padding: 10px 0px 10px 0px;

}

</style>

<script type="text/javascript" src="../SiteAssets/angular.min.js"></script>

<script type="text/javascript" src="../SiteAssets/appScript.js"></script>

<div ng-app="demoApp">

<div ng-controller="tasksController">

<div>

<table>

<tr><th>Task Name</th><td><input type="text" ng-model="newTask.Title" /></td></tr>

<tr>

<th>Start Date</th>

<td>

<input type="text" class="form-control" ng-model="newTask.StartDate" placeholder="Start Date dd/mm/yyyy" />

</td>

</tr>

<tr>

<th>Due Date</th>

<td>

<input type="text" class="form-control" ng-model="newTask.DueDate" placeholder="Due Date dd/mm/yyyy" />

</td>

</tr>

<tr><th>% Complete</th><td><input type="text" ng-model="newTask.PercentComplete" /></td></tr>

</table>

</div>

<div class="action-container"><button ng-click="createTask()">Add Task</button></div>

<table id="taskView">

<thead>

<tr>

<th>Task ID</th>

<th>Task Name</th>

<th>Start Date</th>

<th>Due Date</th>

<th>% Complete</th>

<th>Created Date</th>

<th>Modified Date</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="task in Tasks">

<td>**{{**task.ID**}}**</td>

<td>**{{**task.Title**}}**</td>

<td>**{{**task.StartDate | date: &#39;dd MMM yyyy&#39;**}}**</td>

<td>**{{**task.DueDate | date: &#39;dd MMM yyyy&#39;**}}**</td>

<td>**{{**task.PercentComplete**}}**</td>

<td>**{{**task.Created | date: &#39;dd MMM yyyy&#39;**}}**</td>

<td>**{{**task.Modified | date: &#39;dd MMM yyyy&#39;**}}**</td>

</tr>

</tbody>

</table>

</div>

</div>

1. Copy the below mentioned javascript and paste it in appScript.js code save and publish it if required

var demoApp = angular.module('demoApp', []);

demoApp.factory('dataService', function ($http) {

var actions = {};

var serviceURL = {

tasks: "http://localhost/\_api/lists/GetByTitle('MyTasks')/items",

}

var digest = document.getElementById('\_\_REQUESTDIGEST').value;

var requestHeader = {

get: {

'headers': {

'accept': 'application/json;odata=verbose'

}

},

post: {

'headers': {

'X-RequestDigest': digest,

'content-type': 'application/json;odata=verbose',

'accept': 'application/json;odata=verbose'

}

}

};

actions = {

get: function () {

return $http.get(serviceURL.tasks, requestHeader.get);

},

post: function (data) {

return $http.post(serviceURL.tasks, data, requestHeader.post)

}

}

return actions;

});

demoApp.controller('tasksController', function ($scope, $http, dataService) {

//limited fields are used for sample app

$scope.newTask = {

Title: '',

StartDate: '',

DueDate: '',

PercentComplete: '',

\_\_metadata: { type: 'SP.Data.MyTasksListItem' }

};

function getAllTasks() {

dataService.get().then(function (response) {

$scope.Tasks = response.data.d.results;

});

}

// Getting the item type for the list

function getListItemType(name) {

return "SP.Data." + name[0].toUpperCase() + name.substring(1) + "ListItem";

}

//To create new list item

$scope.createTask = function () {

var item = {

"\_\_metadata": { "type": getListItemType('MyTasks') }

};

dataService.post($scope.newTask).then(function (response) {

//alert(response);

return response;

});

};

function init() {

getAllTasks();

}

init();

});

1. Open the SharePoint site which you are looking to update markup and js
2. Create new list called MyTasks by choosing Tasks list template and make sure your MyTask list has fields which are highlighted in green color in the javascript such as Title, StartDate, DueDate, PercentageComplete
3. To execute your code navigate to index.aspx -> now you can see earlier added list items in page otherwise you can input fields -> click Add Task

Steps to add or remove fields in table as well as ‘Add Task’

1. Add new field in list, example ‘ProjectPhase’
2. Include same in $scope.newTask(to perform add action) variable and also in expression for binding (to show in table)
3. If you want to do it for other list, do the changes at red colored and green colored areas

Note: Don’t forget to include the angular.min.js or angular.js,

How the service call will perform against to SharePoint?

Initially get method will read SharePoint list items in JSON serialized format, with the help of header information <‘heders:{‘accept’: ’application/json:odata=verbose’>. You can minimize result fields by appending specific fields with the following query url syntax such as ?$select=[filedName1, fieldName2, …]) otherwise it will fetch all fields in the SharePoint List. You can read more about OData query at <http://www.odata.org/documentation/odata-version-3-0/url-conventions/>. $http get method haven’t return json object include the $http.get().error() part to determine issue by browser console or debugging the request network area in the developer tool.

In $http.post method we need to mention some of prerequisites like header with SharePoint list content type which is nothing but list content type. In example we are using "\_\_metadata": { "type": getListItemType('MyTasks') } to add list item and we reloaded the list items.

Hi folks next post I am planning on angular directives, please feel free to raise your questions if you are facing any challenges on executing this code.