Introduction

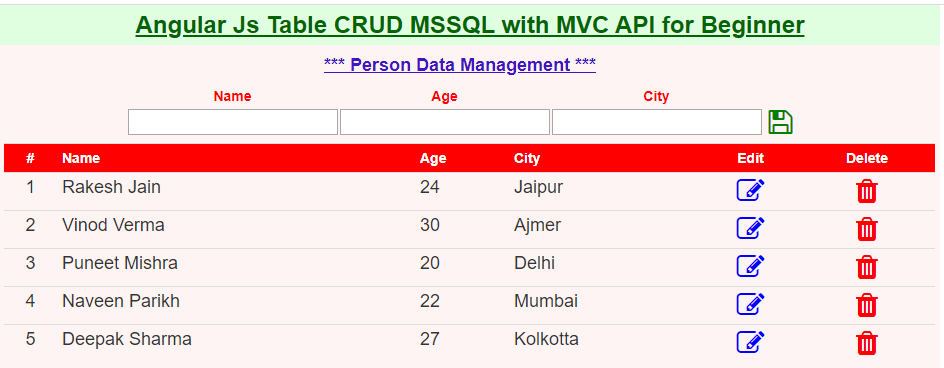
The article is related to managing data table CRUD operations in AngularJS using a SQL database table and ASP.NET MVC web API. This tutorial is for beginners or students.

We created a CRUD functionality in AngularJS. Managing the datatable is easy. Also, we will learn how to publish it on a server to access the web API. Furthermore, we will learn how to test the web API with the Postman client. Finally, we explained all data operations in one page of coding. I hope that it will be helpful for you to manage data tables in your program with advanced mode.

Angular JS code point

* ng-app - This directive is used to bootstrap the application. Normally, it is put in top-level elements like HTML or body tag.
* ng-controller - This is the main component of AngularJS, which contains the state and logic both. It acts as a bridge between services and views.
* $scope - Provides a link between the data and views. It holds all the required data for the views and used within the HTML template.
* {{expression}} - It is called expressions and JavaScript-like code can be written inside. It should be used for mainly small operations in an HTML page.
* ng-init - This directive provides us a way to initialize a variable on the page. ng-repeat-It actually repeats the element on which it is used based on the provided list of data. In our example, persons is a JSON object which contains a list of people.
* $http-It enables you to initiate the AJAX request to the server and get the response. This is a core angular service.

**Example ScreenShot**



Section 1

Section 1 shows a short view of the code part. We create *ng-app* myApp and in the body part we create *ng-controller* myTableCtrl. Both ng-app and ng-controller are the two main parts of AngularJS. In the header tag, we include angular.min.js and link bootstrap.min.css, font-awesome.css to run angular js functionality and design css. Also, we added a custom design class to create a good-looking output. In HTML, we add code to get an Angular app by using angular.module() function, so that we can access the controller by using "appName".controller() function. Next, we created an empty json object array named persons to fill the table. Also, add three functions named Save(),RecEdit() and RecDel() to manage the table record.

1. <html ng-app="myApp">
2. <head>
3. <link href="bootstrap.min.css" rel="stylesheet" />
4. <link href="fontawesome/css/font-awesome.css" rel="stylesheet"/>
5. <script src="angular.min.js"></script>
6. <title>My Data Table</title>
7. <style>
8. .container {text-align: center;background-color: rgb(255, 244, 244);}
9. .container h2{margin:0px;background-color: rgb(224, 255, 224);color:darkgreen;}
10. .row {text-align: center;}
11. .row h4 {color:rgb(67, 21, 192);}
12. .row table{margin: auto;}
13. .row table tr th{color:red;}
14. .row table #row\_header th{color:white;background-color:red;}
15. .row table #row\_header #th1,#th5,#th6{text-align: center;}
16. .row table tr #td1,#td5,#td6{text-align: center;}
17. .row table tr #td1,#td2,#td3,#td4 {font-size:large;}
18. </style>
19. <script>
20. **var** myApp = angular.module('myApp', []);
21. myApp.controller('myTableCtrl',**function**($scope,$http){
22. **var** mykey=0;
23. // create empty json object array.
24. $scope.persons=[];
25. // get all records, local function not related to scope
26. GetRecords = **function** (sqry) {/\*code below\*/ }
27. // save or update record, local function not related to scope
28. SaveRecords=**function**(sqry){/\*code below\*/ }
29. // get record data to update by key.
30. $scope.RecEdit = **function** (per) {/\*code below\*/}
31. // delete record by key
32. $scope.RecDel = **function** (PersonId) {/\*code below\*/}
33. // save or update record
34. $scope.Save = **function** () {/\*code below\*/ }
35. // default load record if available.
36. GetRecords("{'StrQry':'SELECT \* FROM [TblPerson]'}");
37. });
38. </script>
39. </head>
40. <body ng-controller="myTableCtrl">
41. <div **class**="container"><h2><u>Angular Js Table CRUD MSSQL **with** MVC API **for** Beginner</u></h2>
42. <div **class**="container" ng-init="per={}"></div>
43. <div **class**="container"></div>
44. </body>
45. </html>

Here, we create an empty JSON object array named persons=[]. Also, we added two new functions to call the API, which is named GetRecords() and SaveRecords() which receives a parameter JSON object. Both functions have $http request with post method and call the web API URL for reading or saving table data. In our case, we used iis server localhost. The application's name is mywapi.

1. // get all records, local function not related to scope
2. GetRecords = **function** (sqry) {
3. $http({
4. method: 'POST',
5. url: 'http://localhost/mywapi/api/tcrud/recselect',
6. data: sqry
7. }).then(**function** mySuccess(response) {
8. **if** (response.status == 200) {
9. **var** data = response.data;
10. data = JSON.parse(data);
11. **if** (data.length > 0)
12. $scope.persons = data;
13. **else**
14. alert("No record found.");
15. } **else**
16. alert("Server error...");
17. })
18. }
19. // save or update record, local function not related to scope
20. SaveRecords=**function**(sqry){
21. $http({
22. method: 'POST',
23. url: 'http://localhost/mywapi/api/tcrud/recsave',
24. data: sqry
25. }).then(**function** mySuccess(response) {
26. **if** (response.status == 200)
27. GetRecords("{'StrQry':'SELECT \* FROM [TblPerson]'}");
28. **else**
29. alert("Server error...");
30. })
31. }

You have seen that in AngularJS, to access any function or variable we can use $scope. But the functions GetRecords() and SaveRecords() are not related to $scope, because it is local function used to call the web API. The function parameter in JSON is {"StrQry":"Query String"}. Example: "{'StrQry':'SELECT \* FROM [TblPerson]'}". We are going to explain the functions Save(), RecEdit() and RecDel(). The function Save() checks the key of row. If bigger than 0, then it updates the record. Otherwise, it adds a new record in the data table using web API functionality. Here, $scope.per is a single JSON object. This empty JSON object is part of ng-controller, which is available in person form ng-init='per={}' shows in the red-colored rectangle.

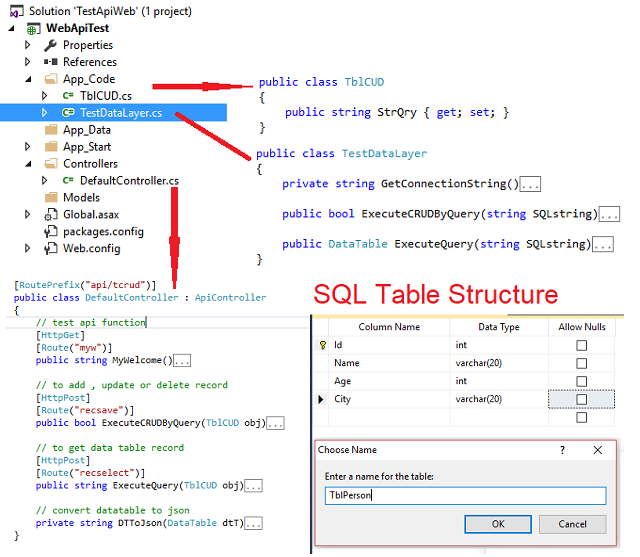
1. // get record data to update by key.
2. $scope.RecEdit = **function** (per) {
3. $scope.per.Name = per.Name;
4. $scope.per.Age = per.Age;
5. $scope.per.City = per.City;
6. mykey = per.Id;
7. }
8. // delete record by key
9. $scope.RecDel = **function** (PersonId) {
10. **var** sqry = { "StrQry": "DELETE FROM [TblPerson] WHERE [Id]='" + PersonId + "'" };
11. SaveRecords(sqry);
12. }
13. // save or update record
14. $scope.Save = **function** () {
16. //default new record save query
17. **var** sqry = { "StrQry": "INSERT INTO [TblPerson] ([Name],[Age],[City]) VALUES('" + $scope.per.Name + "','" + $scope.per.Age + "','" + $scope.per.City + "')" }
19. **if**(mykey>0){
20. sqry = { "StrQry": "UPDATE [TblPerson] SET [Name] = '" + $scope.per.Name + "',[Age] = '" + $scope.per.Age + "',[City] = '" + $scope.per.City + "' WHERE [Id]='" + mykey + "'" }
21. }
22. SaveRecords(sqry);
23. mykey = 0;
24. $scope.per.Name = ''; $scope.per.Age = ''; $scope.per.City = '';
25. }

For the HTML code, three div are available in this section. The first div tag is to show page header title, the second div tag show form of person, and the third div tag shows the table part. In the second div tag, we initialize the empty json object by using the AngularJS directive ng-init='per={}', which shows in the red-colored rectangular. Next, in for we use ng-model directive in input control in the form, and ng-click to get button event in angular js. Now finally, we are going to talk about the main data table in AngularJS. ng-repeat directive is used to show items in json object array. ng-repeat directive is like a loop. Using ng-repeat displays all items in an array in tabular format.

1. <div **class**="container" ng-init="per={}">
2. <div **class**="row">
3. <h4><u>\*\*\* Person Data Management \*\*\*</u></h4>
4. <table>
5. <tr><th>Name</th><th>Age</th><th>City</th></tr>
6. <tr>
7. <td> <input ng-model="per.Name" /></td>
8. <td> <input ng-model="per.Age" /> </td>
9. <td> <input ng-model="per.City" /> </td>
10. <td>
11. <i **class**= "fa fa-save fa-2x" style="color:green;margin-left: 5px;" data-ng-click="Save()"></i></td>
12. </tr>
13. </table>
14. </div>
15. </div>
16. <div **class**="container">
17. <div **class**="row">
18. <table **class**="table table-condensed table-hover" style="width:97%;margin-left:24px;margin-top: 5px;">
19. <tr id="row\_header">
20. <th id="th1">#</th><th>Name</th><th>Age</th><th>City</th><th  id="th5">Edit</th><th id="th6">Delete</th>
21. </tr>
22. <tr id="rdata" ng-repeat="person in persons">
23. <td id="td1">{{$index+1}}</td>
24. <td id="td2">{{person.Name}}</td>
25. <td id="td3">{{person.Age}}</td>
26. <td id="td4">{{person.City}}</td>
27. <td  id="td5"><i **class**= "fa fa-edit fa-2x" style="color:blue;" data-ng-click="RecEdit(person)"></i></td>
28. <td  id="td6"><i **class**= "fa fa-trash fa-2x" style="color:red;" data-ng-click="RecDel(person.Id)"></i></td>
29. </tr>
30. </table>
31. </div>
32. </div>

Section 2 (ASP.NET MVC WEB API and DataTable)

Now, we create an ASP.NET MVC API application. The ASP.NET Web API solution structure is below. To add App\_Code folder, right-click on the project ---> Add ---> Add ASP.NET folder --> Select App\_Code. We created two CS classes in the App\_Code folder, the name is TblCUD.cs and TestDataLayer.cs. TblCUD.cs has only one get, set property which is named StrQry to receive a JSON string. Next, the TestDataLayer.cs has three functions, the first is related to the get connection string from web.config. You need to change as per your local environment. The other two functions are related to accessing the SQL client and connection for the table data record operation.



1. **public** **class** TestDataLayer
2. {
3. **private** **string** GetConnectionString()
4. {
5. **return** ConfigurationManager.ConnectionStrings["ConnString"].ConnectionString;
6. }
8. **public** **bool** ExecuteCRUDByQuery(**string** SQLstring)
9. {
10. SqlConnection sqlCon = **null**;SqlCommand sqlCmd = **null**;
11. **bool** retVal = **false**;
12. **try**
13. {
14. //Setup command object
15. sqlCmd = **new** SqlCommand(SQLstring);
16. sqlCmd.CommandType = CommandType.Text;
17. sqlCmd.CommandTimeout = 100000;
18. sqlCon = **new** SqlConnection(GetConnectionString());
19. sqlCmd.Connection = sqlCon;
20. sqlCon.Open();
21. //Execute the command
22. **int** iR = sqlCmd.ExecuteNonQuery();
23. retVal = iR > 0 ? **true** : **false**;
24. }
25. **catch** { retVal = **false**;**throw**; }
26. **finally**
27. {
28. **if** (sqlCmd != **null**) sqlCmd.Dispose();
29. **if** (sqlCon.State == ConnectionState.Open) sqlCon.Close();
30. }
31. **return** retVal;
32. }
34. **public** DataTable ExecuteQuery(**string** SQLstring)
35. {
37. SqlConnection sqlCon = **null**; SqlCommand sqlCmd = **new** SqlCommand();
38. SqlDataAdapter da = **new** SqlDataAdapter();DataTable dt = **new** DataTable();
39. **try**
40. {
41. //Setup command object
42. sqlCmd = **new** SqlCommand(SQLstring);
43. sqlCmd.CommandType = CommandType.Text;
44. sqlCmd.CommandTimeout = 10000000;
45. da.SelectCommand = (SqlCommand)sqlCmd;
46. sqlCon = **new** SqlConnection(GetConnectionString());
47. sqlCmd.Connection = sqlCon;
48. sqlCon.Open();
49. //Fill the dataset
50. da.Fill(dt);
51. }
52. **catch** {**throw**;}
53. **finally**
54. {
55. **if** (da != **null**) da.Dispose();
56. **if** (sqlCmd != **null**) sqlCmd.Dispose();
57. **if** (sqlCon.State == ConnectionState.Open) sqlCon.Close();
58. }
59. **return** dt;
60. }
61. }

Next, add a new web API controller, in our case, its name is DefaultController. It has four API functions. The first one is for test purposes with a get request. The other two functions have a post request to receive operations related to your data table. The last function is DTToJson to convert a datatable to JSON output. The function is using MSSQL data table to hold the record. Also, you need to create a data table in your SQL server database. DefaultController is derived from ApiController and also we have set attribute [RoutePrefix("api/tcrud")] on top of the controller. We created two API functions ExecuteCRUDByQuery and ExecuteQuery. Both functions use HttpPost, also set attribute Route for both which is [Route("recselect")] and [Route("recsave")] to call an API with a new name in the URL. If you publish on localhost iis server, at the time of testing, the URL will look like "localhost/mywapi/api/tcrud/recselect". Here, localhost is your machine. In my case, mywapi is the application name, api/tcrud is the routeprefix name of the API controller, and finally recselect,recsave is the API function route name.

1. [RoutePrefix("api/tcrud")]
2. **public** **class** DefaultController : ApiController
3. {
4. // test api function
5. [HttpGet]
6. [Route("myw")]
7. **public** **string** MyWelcome()
8. {
9. **return** "Welcome Api world!";
10. }
12. // to add , update or delete record
13. [HttpPost]
14. [Route("recsave")]
15. **public** **bool** ExecuteCRUDByQuery(TblCUD obj)
16. {
17. TestDataLayer objMdl = **new** TestDataLayer();
18. **bool** bResult = objMdl.ExecuteCRUDByQuery(obj.StrQry);
19. **return** bResult;
20. }
22. // to get data table record
23. [HttpPost]
24. [Route("recselect")]
25. **public** **string** ExecuteQuery(TblCUD obj)
26. {
27. **string** sJson = "";
28. TestDataLayer objMdl = **new** TestDataLayer();
29. DataTable dt = **null**;
30. dt = objMdl.ExecuteQuery(obj.StrQry);
31. **if** (dt != **null**)
32. sJson = DTToJson(dt);
33. **return** sJson;
34. }
36. // convert datatable to json
37. // code available in source code
38. **private** **string** DTToJson(DataTable dtT){ }
39. }

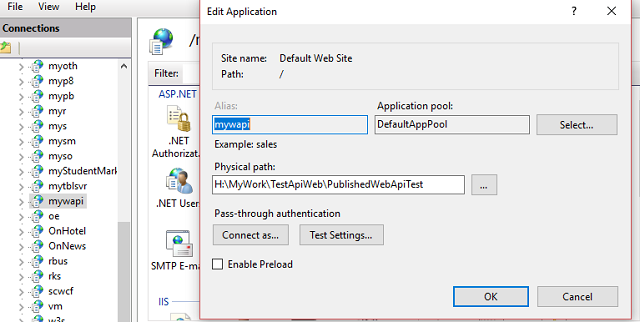
In our case, the table name is TblPerson. The structure is shown below:

1. **SET** ANSI\_NULLS **ON**
2. GO
3. **SET** QUOTED\_IDENTIFIER **ON**
4. GO
5. **SET** ANSI\_PADDING **ON**
6. GO
7. **CREATE** **TABLE** [dbo].[TblPerson](
8. [Id] [**int**] IDENTITY(1,1) NOT NULL,
9. [**Name**] [**varchar**](20) NOT NULL,
10. [Age] [**int**] NOT NULL,
11. [City] [**varchar**](20) NOT NULL,
12. **CONSTRAINT** [PK\_TblPerson] **PRIMARY** **KEY** CLUSTERED
13. (
14. [Id] **ASC**
15. )**WITH** (PAD\_INDEX = **OFF**, STATISTICS\_NORECOMPUTE = **OFF**, IGNORE\_DUP\_KEY = **OFF**,
16. ALLOW\_ROW\_LOCKS = **ON**, ALLOW\_PAGE\_LOCKS = **ON**) **ON** [**PRIMARY**]
17. ) **ON** [**PRIMARY**]
18. GO
19. **SET** ANSI\_PADDING **OFF**
20. GO

Section 3(Published and Test API on Postman)

To save your time, I recommend downloading our application source code and publishing it on your local IIS server. Open the Web API application, right-click on the project, then click on publishing action and follow the instructions. Just open the IIS server, then next right-click on the default website icon --> add application ---> Give the application an Alias Name--> select your published path location --> OK.

Repeat the same process to publish the AnguarJS app on the local IIS server.



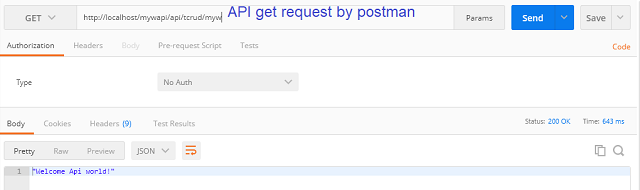
**Key Point**

Also, check your DefaultAppPool advance setting, if the application has any issue after publishing, then enable enable 32-bit Application mode set to true. Also, if your database server has not connected after publishing, then add a new login and user by using SQL server management studio.

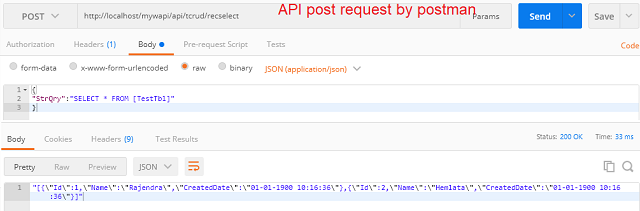
**Postman Client**

Our next step is to test our API via Postman. You need to download your postman client from the web and install it in Google Chrome. Then, you have to add a request GET or POST and API URL. In our case, first, we check the GET request method of our published API. The Get request method is shows a welcome message. So in the Postman client, you need to follow some steps below

Ppen postman --> select method get --> then add API URL --> click on send



Next, we check the post method to get a record in the data table. Open postman --> select method post --> then add API URL --> select Body --> radio button raw --> input type JSON (application/JSON) --> write JSON object data --> click on send



**Note**

In this article, we have not discussed any type of web API security. This article's purpose is for publishing a web API, using the web API, postman testing the API, and creating an API for learning purposes.

* Uploaded source code of the web API, TestApiWeb\_SourceCode.zip
* Uploaded source code of AngularJS, mydatatableserver\_angularjs.zip
* Uploaded the web API published code, PublishedWebApiTest.zip