When it comes to storing and retrieving large data on SharePoint, my approach (depending on scope though) is usually to store the data in an external database  where I can have full control over the data retrieval and totally avoid dealing with the large list issue due to SharePoint list throttling limit, then render the data on a SharePoint page using visual web part. But like I indicated above, there are a number of things you have to consider before deciding weather you want to store your data on a SharePoint list or directly on an SQL database table.  Important questions to answer are:

* Will end users be creating views based on different filtering parameter from time to time? if the answer is yes, you might want to consider using a SharePoint list as it already has administrative tools for power users to perform these tasks. This may be a quick win approach for you, particularly where deadline is against you. But you have to bear in mind that your users might have to deal with the large list issue when retrieving data via the views. One way to avoid list throwing error when retrieving data is to index a column on the list. Hope it works the way you want, good luck with that.  **Note:**In SharePoint online and SharePoint 2016, Microsoft has included auto indexing tool to manage lists that are reaching or getting over the throttling limit.  See [here](http://www.learningsharepoint.com/2016/03/29/office-365-how-sharepoint-online-handles-list-view-threshold/) for more info.
* Would the solution be required to export large result data to excel for reporting purpose? If yes, you might want to consider storing your data directly on sql database. One thing is certain here, You can always run a query against your table and export the result to an excel sheet no matter how large. SharePoint list tends to error out when the data is too large. I don't know if Microsoft has made any improvement on this in the latest version.
* Are there likely going to be an upgrade to SharePoint farm or Office 365 that can break the solution? Ensure you follow best practices when building your web part. Upgrade may not really break your code if care is taken to keep your artifacts in safe places. I have deployed a web part I built in 2010 to a 2013 environment and it worked just fine. Thankfully, with SharePoint app model/SharePoint framework and the support for RESTful APIs, things are now made easy.
* Another thing that can constrain you is your team or client's development approach. You have to conform to what the style is in your organization unless you can make them see convincing reasons why they have to adopt a new approach.

Enough said!!! Let's drop the debate and get back to the topic. Supposing there is an existing data in an sql database, be it azure or on premise database server and you are required to make the data available for CRUD operations in SharePoint for your users. This is a scenario where this article will be of help to you. In this article, I will show you how you can perform CRUD operations on Azure SQL database from a SharePoint web part. Note that the steps are the same for on-premise SQL database except for steps required in setting up azure database.  
  
In this article, I will be covering:

* How to set up SQL database in Azure
* Creating restful web API and connect to the azure sql database
* Publish the API to Azure
* Enable cross-origin in the Web API
* How to create SPFX web part
* Consume the Web API created in part 1 in the Spfx web part
* Implement CRUD operations in the Spfx web part
* Deploy to SharePoint Online

**Setting up SQL Azure database**

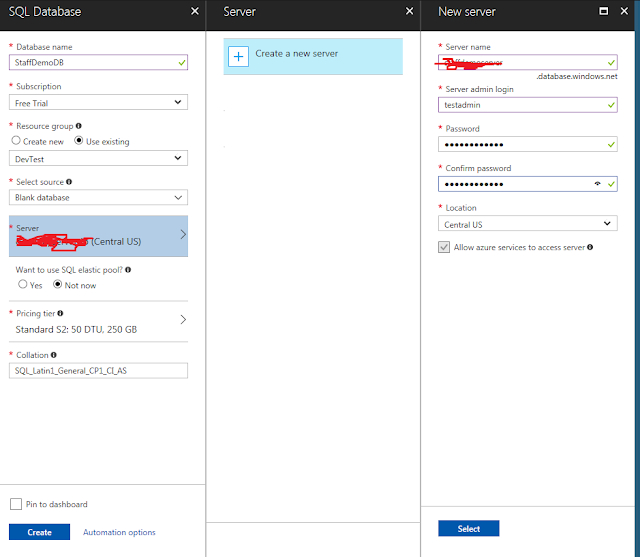
Log into your azure account and follow the steps below to create an SQL database (note that you can create a trial azure account for testing purpose). 

The steps are pretty straightforward.

1. Enter a name for your database as shown below

2. Create a new server, name it and click the**select** button to select the server on which to create the database.

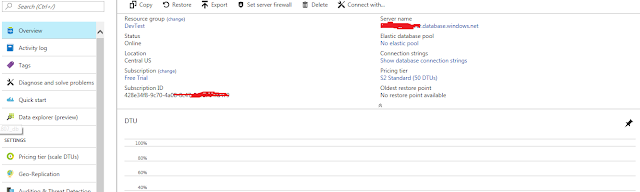
3. Then click**create** to create the database (note the creation process might take sometime, and you might have to refresh the page to see the new database).

[](https://1.bp.blogspot.com/-hFsvALjLcWY/Wkb4fkmR1GI/AAAAAAAAB4I/mtRtbJCLwaISsMNyAQhlP0tZIbLUgw1bACLcBGAs/s1600/sql_db.PNG)

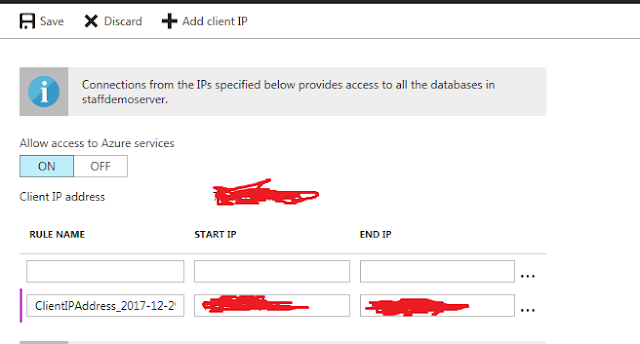
Ensure you remember your password\*\*\*

The next step is to create a firewall rule and add your current client IP address to it. This step is important for you to connect to the database server from your client system.

4. Click on the newly created database, you should see an interface similar to the one below.  Note down the server name (i.e. yourserver.database.windows.net). You will need it when you want to connect to the server.

[](https://4.bp.blogspot.com/-q0Ylih2kdj0/WkcCmNTaVbI/AAAAAAAAB4c/OCn8ap7VOgMF1iuYCrlRfm0_Sdz6y6tSwCLcBGAs/s1600/sql_db_done.PNG)

5. Click on **Set server firewall**in the toolbar  
6. Click **Add client IP** in the toolbar to add your current client IP address to the firewall. Note your public IP will be automatically retrieved.

[](https://1.bp.blogspot.com/-oJwao6Gv0jM/WkcCoXBZnsI/AAAAAAAAB4o/ruOrYnpRIfAiywkYQ3Izv5P7n_hoNPUzgCEwYBhgL/s1600/clientIP.PNG)

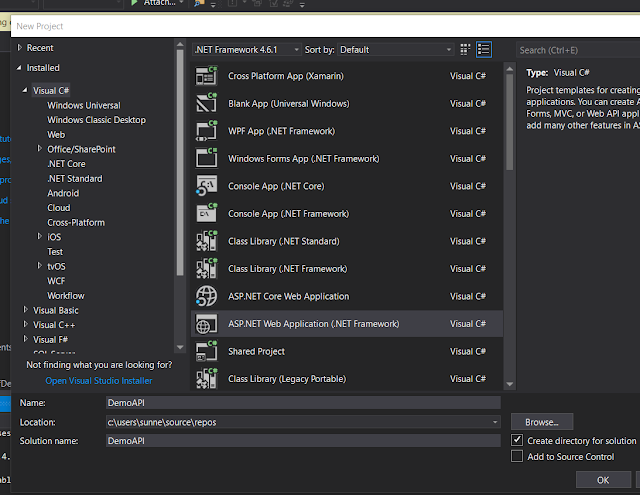
7. Click **Save**to finish the process.  
  
Now you are done with the first task. The next step is to create our web API solution.  
Let's fire up visual studio. I will be using visual studio 2017 community edition.

**Creating restful web API and connect to the azure sql database**

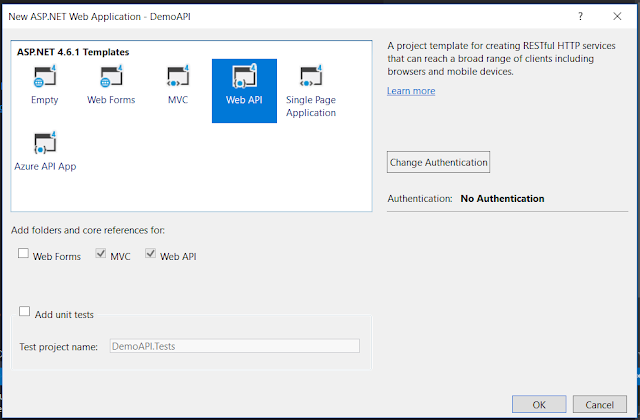
Before you start this step, ensure you download SQL server management studio [here](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms) if you don't already have it installed. We will need it to connect to our database. You could also connect to it within VS but I prefer using SSMS.

1. Launch visual studio and create a new ASP.NET Web Application project

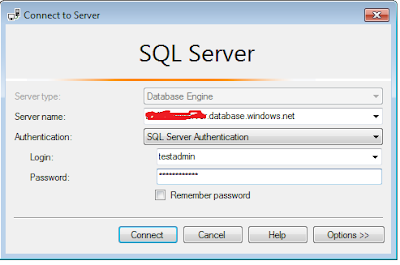
2. Name it**DemoAPI** or whatever you like

[](https://1.bp.blogspot.com/-UsExq_-NcHk/WkcLkldPjaI/AAAAAAAAB40/wWWHairHCocnj93XhLq77VHuhTNYs_YUACLcBGAs/s1600/AspNet.PNG)

3. Follow the screenshot below to complete the process. When you are done, click **Ok**

[](https://1.bp.blogspot.com/-c6LAgMq4IN4/WkcMauE9_sI/AAAAAAAAB48/czx-v4DYapIN0apBQIgW5EEjB94_lKhLgCLcBGAs/s1600/AspNet2.PNG)

Now let's connect to our database server and create a table in our database  
4. Open SQL server management studio and connect your server (the server name you saved earlier)

[](https://4.bp.blogspot.com/-TFRu4oDGj0I/WkcOu-47Q0I/AAAAAAAAB5I/nhFXc_1X8tQuqmE-JrOHXIEpOZrfRQkeACLcBGAs/s1600/ssms.PNG)

5. Put in the sever login Id and password you created when you set up the server in azure

6. Click **Connect**to open the the server

7. Run the script below to create a table in your database. Be sure to change the database name to match the database you created earlier.

Use StaffDemoDB

Go

Create table Employees

(

ID int primary key identity,

FirstName nvarchar(50),

LastName nvarchar(50),

Gender nvarchar(50),

Salary int

)

Go

Insert into Employees values ('Mark', 'Hastings', 'Males', 6000)

Insert into Employees values ('Jas', 'Hastings', 'Males', 6000)

Insert into Employees values ('Mark', 'Mikel', 'Males', 6000)

Insert into Employees values ('Mitchell', 'Hastings', 'Males', 6000)

Insert into Employees values ('Mark', 'Jones', 'Males', 6000)

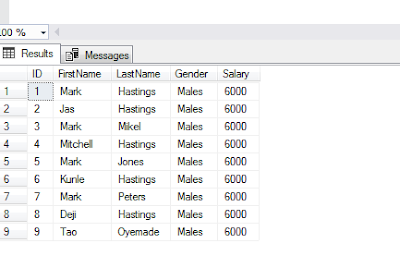
Insert into Employees values ('Kunle', 'Hastings', 'Males', 6000)

Insert into Employees values ('Mark', 'Peters', 'Males', 6000)

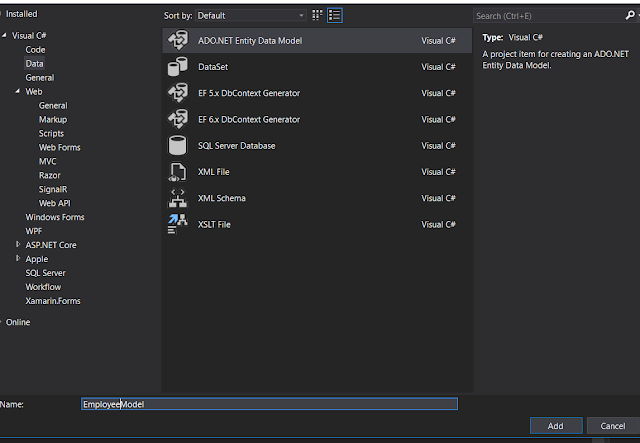
Insert into Employees values ('Deji', 'Hastings', 'Males', 6000)

Insert into Employees values ('Tao', 'Oyemade', 'Males', 6000)

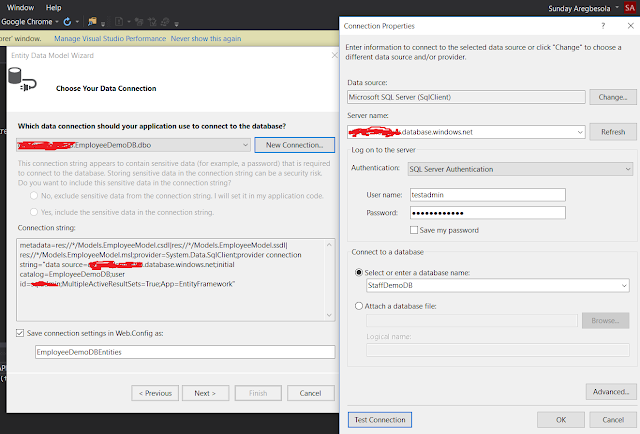
If everything works fine, your Employee table should be created and look like this:

[](https://4.bp.blogspot.com/-SJaq_VO7FpE/WkclkIVkGUI/AAAAAAAAB5Y/Dnk3j_Up6IwSdyIasgAyABJCUvfETkC4wCLcBGAs/s1600/database.PNG)

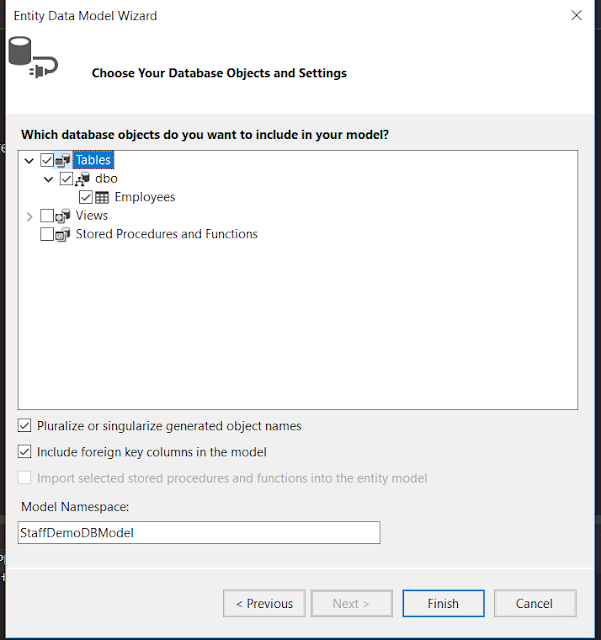
The next step is to create connection to our db using ADO.NET Entity Data Model  
  
8. In the solution explorer of your project, right click on **Models**folder, select **Add**> **New Item** > **Data** and click**ADO.NET Entity Data Model**  
9. Name it EmployeeModel

[](https://3.bp.blogspot.com/-V0B7egd5SoE/WkcoaukTMRI/AAAAAAAAB5k/gnBNEUBqSYIu7sq3XWaU-5X7MFbdh-H9ACLcBGAs/s1600/Model.PNG)

10. Click **Add,** then select **EF Designer from database** in the next screen and click **Next**  
11. In the **Choose your database connection**, click **New Connection** and supply details to connect to your database, click **Ok** and click **Next**

[](https://4.bp.blogspot.com/-HdXeWmdw5hA/WkcsTk56L3I/AAAAAAAAB58/NpcasVqDkk8qLjMqeZXw77AL3Va5tagEACLcBGAs/s1600/connection.PNG)

12.Select **Entity framework 6x**and click**Next**  
**13.**Select your table and click **Finish**

[](https://2.bp.blogspot.com/-TTmMCHDZXy0/WkctlOpJhcI/AAAAAAAAB6I/c7zz70bf3rccqX3g5Rez8SW1eBOa4TNnQCLcBGAs/s1600/finishCon.PNG)

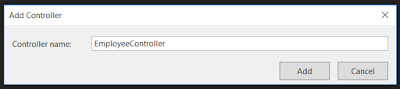
The next step is to publish our API to azure. Follow the steps below to do this

**Publish the API to Azure**

Before publishing the API, lets create a Get operation to fetch our list of employees from the database

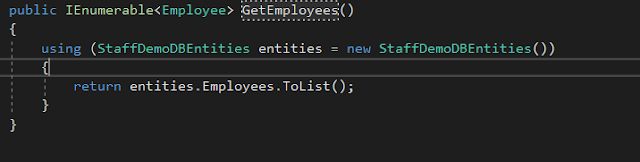
1. Right click on the Controller folder > **Add**> **Controller**.. > **Web API 2 Controller -Empty**

2. Name it **Employee.**Note you only have to rename the highlighted part. If you use Employee as advised, you will have **EmployeeController**as name. See the image below

[](https://1.bp.blogspot.com/-6bY4kw7f0Ks/WkcwQdL27uI/AAAAAAAAB6U/7UnwUe31XpwznBHuoNSKiHjoMWTaEd8sACLcBGAs/s1600/controller.PNG)

3. Click **Add**

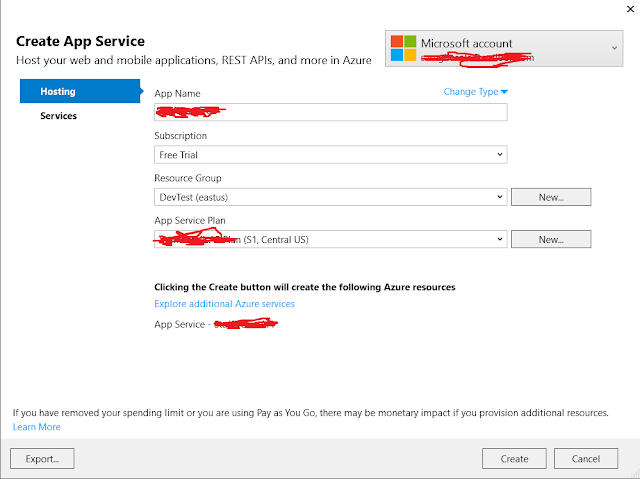
4. Type out the code below into your controller. If you have used the same database and table name, you should have no error

[](https://2.bp.blogspot.com/-dAM_W-hfM8k/WkczIm5r4TI/AAAAAAAAB6o/ZL0xNGoleaYz0xIqMipRHwAiAOw9lQQ_gCLcBGAs/s1600/code2.PNG)

5. Now let's publish. Right click the project name and click **publish**...

6. Click on **Microsoft Azure App Service** and select **Create New** 

7. Create an App Name with other settings as shown below. You can leave the default names if you wish. When you are done, click **Create**. This may take some time to complete.

[](https://2.bp.blogspot.com/-8wnYoRbdApg/Wkc42lbPEhI/AAAAAAAAB7A/utH-FFb_g8AQ1oMviDmHZ_un3tRQDe9bwCLcBGAs/s1600/publish3.png)

8. When it is finished, your web API will be published to Azure and opens in browser

9. To test your web API, append /api/employee to the end of the url displayed in the browser. You should have something like this "http://apiname.azurewebsites.net/api/employee". Notice the last parameter in the url-"employee". If you did not name your controller employee, this url won't work for you. You have to use the name of your controller.

10. When you open the url, you should see your data in XML format, or JSON if you have formatted your output.

The last step in this article is to enable cross origin resource sharing

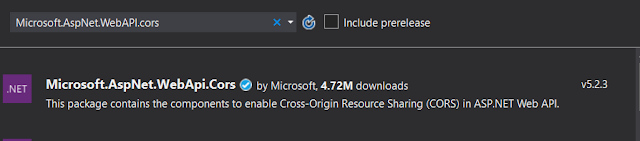
**Enable cross-origin in the Web API**

To be able to call this web API from another domain other than azure, you need to enable cross origin attribute in your code. Since we are going to be calling this API in SharePoint, we will add this attribute to our code.

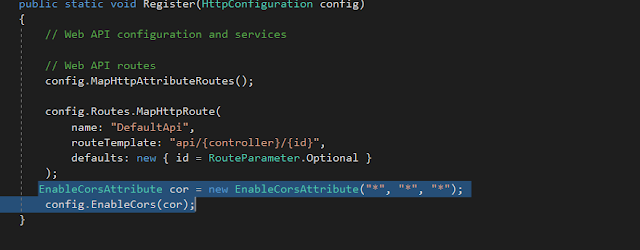
1. Right click on your project > **Manage NuGet Packages > Microsoft.ASPNET.WebAPI.Cors**

2. Click **Install**to add it to your project.

3. Click **Ok** to the warning and**accept** the installation

[](https://3.bp.blogspot.com/-zp5kWsDtPvI/Wkc-wrm1O4I/AAAAAAAAB7Y/poDsyV_wGpIIGpEkETBtd0i00qYIOmEaQCLcBGAs/s1600/origin.PNG)

4. Once installation completes, open WebAPIConfig.cs in**App\_Start** folder and add the following highlighted code inside the **Register**method

[](https://4.bp.blogspot.com/-N-6q69_NKGg/Wkc_802BncI/AAAAAAAAB7k/mdDEGZ2aBtgrJRrc9Da9GDOFe8oU_cZeQCLcBGAs/s1600/originEnabled.PNG)

**Note:**The \* in the above code means I want to accept all domains, apply it to all headers and methods. This may not be what you want to do in production. Be sure to understand what you are doing.

5. Right click your project and click **publish**to publish your changes to Azure.

Agenda for this post are:

* Update our Web API to include Post, Update and Delete methods
* Creating SPFX web part
* Consume the Web API and Implement CRUD operations in Spfx web part
* Test the web part

**Updating the API**

If you have created a Web API solution as described in part 1, go ahead and copy the code below to update your solution, otherwise go back to part 1 to create a web API solution.

 Your final code should look like this:

|  |  |
| --- | --- |
|  | using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Net; |
|  | using System.Net.Http; |
|  | using System.Web.Http; |
|  | using DemoAPI.Models; |
|  |  |
|  | namespace DemoAPI.Controllers |
|  | { |
|  | public class EmployeeController : ApiController |
|  | { |
|  | public IEnumerable<Employee> GetEmployees() |
|  | { |
|  | using (StaffDemoDBEntities entities = new StaffDemoDBEntities()) |
|  | { |
|  | return entities.Employees.ToList(); |
|  | } |
|  | } |
|  |  |
|  | public HttpResponseMessage GetEmployeeById(int id) |
|  | { |
|  | try |
|  | { |
|  | using (StaffDemoDBEntities entities = new StaffDemoDBEntities()) |
|  | { |
|  | var entity= entities.Employees.FirstOrDefault(e => e.ID == id); |
|  | //entities.Employees.Where(e => e.ID == id).FirstOrDefault(); |
|  | if (entity != null) |
|  | { |
|  | return Request.CreateResponse(HttpStatusCode.OK, entity); |
|  | } |
|  | else |
|  | return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Employee with id " + id + "was not found"); |
|  |  |
|  | } |
|  | } |
|  | catch(Exception ex) |
|  | { |
|  | return Request.CreateErrorResponse(HttpStatusCode.BadGateway, ex); |
|  | } |
|  | } |
|  |  |
|  | public HttpResponseMessage Post([FromBody] Employee employee) |
|  | { |
|  | try { |
|  | using (StaffDemoDBEntities entities = new StaffDemoDBEntities()) |
|  | { |
|  | entities.Employees.Add(employee); |
|  | entities.SaveChanges(); |
|  | var message = Request.CreateResponse(HttpStatusCode.Created, employee); |
|  | message.Headers.Location = new Uri(Request.RequestUri + "/" + employee.ID.ToString()); |
|  | return message; |
|  | } |
|  | } |
|  | catch (Exception ex) |
|  | { |
|  | return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex); |
|  |  |
|  | } |
|  | } |
|  | public HttpResponseMessage Put(int id, [FromBody] Employee employee) |
|  | { try |
|  | { |
|  | using (StaffDemoDBEntities entities = new StaffDemoDBEntities()) |
|  | { |
|  | var entity = entities.Employees.FirstOrDefault(e => e.ID == id); |
|  |  |
|  | if (entity != null) |
|  | { |
|  | entity.FirstName = employee.FirstName; |
|  | entity.LastName = employee.LastName; |
|  | entity.Gender = employee.Gender; |
|  | entity.Salary = employee.Salary; |
|  | entities.SaveChanges(); |
|  | return Request.CreateResponse(HttpStatusCode.OK, entity); |
|  | } |
|  | else |
|  | return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Employee with id " + id + "was not found"); |
|  | } |
|  | } |
|  | catch(Exception ex) |
|  | { |
|  | return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex); |
|  | } |
|  | } |
|  |  |
|  | public HttpResponseMessage Delete(int id) |
|  | { |
|  | try |
|  | { |
|  | using (StaffDemoDBEntities entities = new StaffDemoDBEntities()) |
|  | { |
|  | var entity = entities.Employees.FirstOrDefault(e => e.ID == id); |
|  | if (entity != null) |
|  | { |
|  | entities.Employees.Remove(entity); |
|  | entities.SaveChanges(); |
|  | return Request.CreateResponse(HttpStatusCode.OK, entity); |
|  | } |
|  | else |
|  | return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Employee object with id " + id + "was not found."); |
|  | } |
|  | } |
|  | catch (Exception ex) |
|  | { |
|  | return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/saregbesola/9504f0ada7fcfa3366651d2472a8b8a8/raw/ce64914f1c4e0bd7e93f326aa9f33b1f5c8983e3/WebAPIDemoWithSharePoint)[**WebAPIDemoWithSharePoint**](https://gist.github.com/saregbesola/9504f0ada7fcfa3366651d2472a8b8a8#file-webapidemowithsharepoint) hosted with ❤ by [**GitHub**](https://github.com/)

**Right-click**on your project and publish to Azure as described in part 1.  
 **Creating SPFX web part**

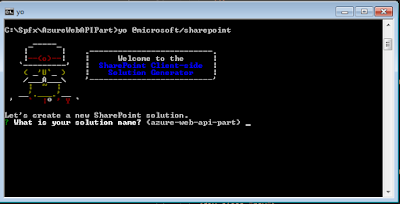
The first step is to set up your dev environment for SharePoint framework development. if you have not already done so, follow this link to prepare environment [Set up your SharePoint Framework development environment](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/set-up-your-development-environment).

Now that you have finished setting up your environment, follow the steps below to create an spfx web part:

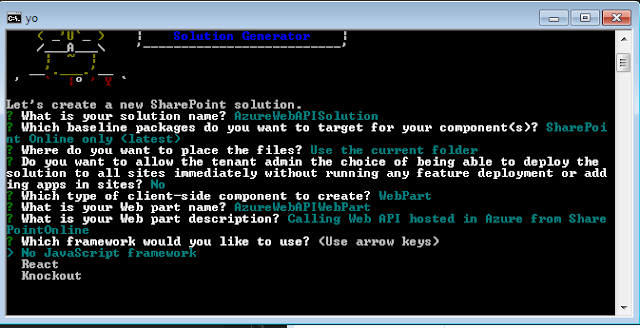
1. Open command line, cmd

2. Create a folder using the command md, call it spfx (i.e. md spfx)

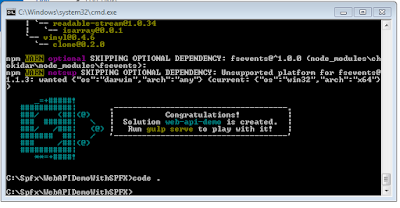
3. Navigate to the folder you just created using "cd spfx" and type yo @microsoft/sharepoint

[](https://2.bp.blogspot.com/-yBFdjNMCbs0/WlE_4m86Q_I/AAAAAAAAB9o/ht-a6JTYWFE1_8K7L4-1et7DiS2-QHLxACLcBGAs/s1600/azurewebpart1.PNG)

4. Follow the screenshot below to create a solution for your web part. **Note that you can name your web part AzureWebAPIWebPart as shown below or anything you wish. Just ensure that when you are copying the code from here, you replace your web part name where necessary.**

[](https://3.bp.blogspot.com/-YIm39_RTK9Q/WlFAIFKEYeI/AAAAAAAAB9s/LHILUHmJQHUrUidte64LU_bvcrJo72NHgCLcBGAs/s1600/azurewebpart2.PNG)

5. If all is well, you will seen a congratulations screen that looks like below. If there is any issue, ensure you fix it before going on to the next step.

[](https://3.bp.blogspot.com/-XBrJVrc1V98/WlFEr9c5DuI/AAAAAAAAB94/7GWm3evX2-QCzv-h0KjJe902zWZMr0QtgCLcBGAs/s1600/Congratulations_screen_gulp.PNG)

6. You can run **gulp serve**to ensure everything is okay. This should launch your default browser. Click the plus sign and add your web part. If all is fine, you should see your web part with default look.   
 Now let's install jquery and Typings for jquery as we require jquery ajax to make calls to our web API.  
7. To add jquery to your project, on command line type:   
    *npm jquery --save*  
*npm @types/jquery --save*

8. Then type "**code .**" without the quote to open visual studio code editor.

9. Click the drop down arrow next to **src >Webparts>YourWebpartName**  
10. Verify that jquery library was added successfully to your project by clicking the arrow next to **nodes\_modules > jquery > dist.**Verify **jquery.min.js** is there.  
Let's un-bundle the jquery library by externalizing it. This step is optional but good for page load performance.  
11. Open **Config.json**. **YourWebpartName > Config > Config.json**  
12. Modify it as shown below

"externals": {

"jquery": {

"path": "node\_modules/jquery/dist/jquery.min.js",

"globalName": "jQuery"

}

13. Go back to **src**and open **YourWebpartName.ts.**Let's import the jquery and load css file from a SharePoint library in the **YourWebpartName.ts.**  
**Note:**Normally you would want to put your css classes in the **.scss** file (i.e. **YourWebpartName.scss**)  and import it into the **YourWebpartName.ts**file (this is actually done automatically during project creation). But I decided to put it in a SharePoint list because my css selectors are grouped and I could not reference them using "**styles.selector**". Let me explain this a bit further:

|  |  |
| --- | --- |
|  | <div class="${ styles.webApiDemoWebPart }"> |
|  | <div class="${ styles.container }"> |
|  | <div class="${ styles.row }"> |
|  | <div class="${ styles.column }"> |
|  | <span class="${ styles.title }">Welcome to SharePoint!</span> |
|  | <p class="${ styles.subTitle }">Customize SharePoint experiences using Web Parts.</p> |
|  | <p class="${ styles.description }">${escape(this.properties.description)}</p> |
|  | <a href="https://aka.ms/spfx" class="${ styles.button }"> |
|  | <span class="${ styles.label }">Learn more</span> |
|  | </a> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |

[**view raw**](https://gist.github.com/saregbesola/a31f346694bd1d6f5b3c62a3c35d38db/raw/3f1742a64f6ae74b50e990092e3f0c49b71c3d35/Example%20Styles)[**Example Styles**](https://gist.github.com/saregbesola/a31f346694bd1d6f5b3c62a3c35d38db#file-example-styles) hosted with ❤ by [**GitHub**](https://github.com/)

"from the above code, **class="${ styles.container }"**means selector **.container** is being referenced from the stylesheet file (i.e. **YourWebpartName.scss** file). But when your css class selectors are grouped like:  
***.container .row .column{***  
***width: 70%;***  
***}***  
then using**styles.selector** becomes difficult as you cannot have something like **class="${styles.container .row .column}".**This will not work. If you know how to achieve it, kindly share it in the comment section.  
To load our css file from a list or an external location, we have to import **SPComponentLoader**into our project  
14. Add the code line below in the import section at the top of your code inside **YourWebpartName.ts**file

*import { SPComponentLoader } from '@microsoft/sp-loader';*

15. We also need to import jquery into the file. Copy and paste the following line of code above the **SPComponentLoader.**No special reason for putting it above **SPComponentLoader**, it's just my practice.

*import \* as jquery from 'jquery';*

16. Copy and paste the css file below in notepad and save as **styles.css**

|  |  |
| --- | --- |
|  | div.blueTable { |
|  | border: 1px solid #1C6EA4; |
|  | background-color: #EEEEEE; |
|  | width: 100%; |
|  | text-align: left; |
|  | border-collapse: collapse; |
|  | } |
|  | .divTable.blueTable .divTableCell, .divTable.blueTable .divTableHead { |
|  | border: 1px solid #AAAAAA; |
|  | padding: 3px 2px; |
|  | } |
|  | .divTable.blueTable .divTableBody .divTableCell { |
|  | font-size: 13px; |
|  | } |
|  | .divTable.blueTable .divTableRow:nth-child(even) { |
|  | background: #D0E4F5; |
|  | } |
|  | .divTable.blueTable .divTableHeading { |
|  | background: #1C6EA4; |
|  | background: -moz-linear-gradient(top, #5592bb 0%, #327cad 66%, #1C6EA4 100%); |
|  | background: -webkit-linear-gradient(top, #5592bb 0%, #327cad 66%, #1C6EA4 100%); |
|  | background: linear-gradient(to bottom, #5592bb 0%, #327cad 66%, #1C6EA4 100%); |
|  | border-bottom: 2px solid #444444; |
|  | } |
|  | .divTable.blueTable .divTableHeading .divTableHead { |
|  | font-size: 15px; |
|  | font-weight: bold; |
|  | color: #FFFFFF; |
|  | border-left: 2px solid #D0E4F5; |
|  | } |
|  | .divTable.blueTable .divTableHeading .divTableHead:first-child { |
|  | border-left: none; |
|  | } |
|  |  |
|  | .blueTable .tableFootStyle { |
|  | font-size: 14px; |
|  | font-weight: bold; |
|  | color: #FFFFFF; |
|  | background: #D0E4F5; |
|  | background: -moz-linear-gradient(top, #dcebf7 0%, #d4e6f6 66%, #D0E4F5 100%); |
|  | background: -webkit-linear-gradient(top, #dcebf7 0%, #d4e6f6 66%, #D0E4F5 100%); |
|  | background: linear-gradient(to bottom, #dcebf7 0%, #d4e6f6 66%, #D0E4F5 100%); |
|  | border-top: 2px solid #444444; |
|  | } |
|  | .blueTable .tableFootStyle { |
|  | font-size: 14px; |
|  | } |
|  | .blueTable .tableFootStyle .links { |
|  | text-align: right; |
|  | } |
|  | .blueTable .tableFootStyle .links a{ |
|  | display: inline-block; |
|  | background: #1C6EA4; |
|  | color: #FFFFFF; |
|  | padding: 2px 8px; |
|  | border-radius: 5px; |
|  | } |
|  | .blueTable.outerTableFooter { |
|  | border-top: none; |
|  | } |
|  | .blueTable.outerTableFooter .tableFootStyle { |
|  | padding: 3px 5px; |
|  | } |
|  | /\* DivTable.com \*/ |
|  | .divTable{ display: table; } |
|  | .divTableRow { display: table-row; } |
|  | .divTableHeading { display: table-header-group;} |
|  | .divTableCell, .divTableHead { display: table-cell;} |
|  | .divTableHeading { display: table-header-group;} |
|  | .divTableFoot { display: table-footer-group;} |
|  | .divTableBody { display: table-row-group;} |
|  |  |
|  |  |
|  | /\* Css styles for the form \*/ |
|  |  |
|  | \* { |
|  | box-sizing: border-box; |
|  | } |
|  |  |
|  | input[type=text], select, textarea{ |
|  | width: 100%; |
|  | padding: 12px; |
|  | border: 1px solid #ccc; |
|  | border-radius: 4px; |
|  | box-sizing: border-box; |
|  | resize: vertical; |
|  | } |
|  |  |
|  | label { |
|  | padding: 12px 12px 12px 0; |
|  | display: inline-block; |
|  | } |
|  |  |
|  | input[type=submit] { |
|  | background-color: #4CAF50; |
|  | color: white; |
|  | padding: 12px 20px; |
|  | border: none; |
|  | border-radius: 4px; |
|  | cursor: pointer; |
|  | float: right; |
|  | } |
|  |  |
|  | input[type=submit]:hover { |
|  | background-color: #45a049; |
|  | } |
|  |  |
|  | .container { |
|  | border-radius: 5px; |
|  | background-color: #f2f2f2; |
|  | padding: 20px; |
|  | } |
|  |  |
|  | .col-25 { |
|  | float: left; |
|  | width: 25%; |
|  | margin-top: 6px; |
|  | } |
|  |  |
|  | .col-75 { |
|  | float: left; |
|  | width: 75%; |
|  | margin-top: 6px; |
|  | } |
|  |  |
|  | /\* Clear floats after the columns \*/ |
|  | .row:after { |
|  | content: ""; |
|  | display: table; |
|  | clear: both; |
|  | } |
|  |  |
|  | /\* Responsive layout - when the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other \*/ |
|  | @media (max-width: 600px) { |
|  | .col-25, .col-75, input[type=submit] { |
|  | width: 100%; |
|  | margin-top: 0; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/saregbesola/cfebe7cc656d5a5056ec901fc21aa301/raw/c2ab4d4d69efa7d34df1e364286ab47df30f02b8/Styles)[**Styles**](https://gist.github.com/saregbesola/cfebe7cc656d5a5056ec901fc21aa301#file-styles) hosted with ❤ by [**GitHub**](https://github.com/)

17. Upload it to **Site Asset** library in your SharePoint site. **Note**you can use any library of your choice.  
18. Right-click on the **styles.css** inside the library and click **Copy link.** follow the instruction to copy the link. You can get the short link to the file by pasting the link in the browser and hit enter. This will give you something like *https://yourspsite.sharepoint.com/sites/dev/SiteAssets/styles.css?slrid=65513e9e-e0b3-4000-c649-ece51bb97dbc.*  
19. Copy*https://yourspsite.sharepoint.com/sites/dev/SiteAssets/styles.css* out of the link and go back to **YourWebpartName.ts**  
20. Add the following line of code just under **SPComponentLoader**, but above **export interface WebpartProps**  
SPComponentLoader.loadCss('https://yourspsite.sharepoint.com/sites/dev/SiteAssets/styles.css')  
  
Let's create our form and a table to display data from Azure database.  
21. Replace the code block in **public render ()**with the code block below

|  |  |
| --- | --- |
|  | public render(): void { |
|  | this.domElement.innerHTML = ` |
|  | <div class="container"> |
|  | <form> |
|  | <div class="row"> |
|  | <h2 style="text-align:left" id="statusMode"> |
|  | Add New Record |
|  | </h4> |
|  | <div class="col-25"> |
|  | <label for="fname">First Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="fname" name="firstname" placeholder="First Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="lname">Last Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="lname" name="lastname" placeholder="Last Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="gender">Gender</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="gender" name="gender" placeholder="Gender.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="salary">Salary</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="salary" name="gender" placeholder="Salary.."> |
|  | </div> |
|  | </div> |
|  | <!-- hidden controls --> |
|  | <div style="display: none"> |
|  | <input id="recordId" /> |
|  | </div> |
|  | <div class="row"> |
|  | <input type="submit" value="Submit" id="btnSubmit"> |
|  | </div> |
|  | </form> |
|  | </div> |
|  |  |
|  | //Creates a table to display data from Azure database |
|  | <div class="divTable blueTable"> |
|  | <div class="divTableHeading"> |
|  | <div class="divTableRow"> |
|  | <div class="divTableHead">First Name</div> |
|  | <div class="divTableHead">Last Name</div> |
|  | <div class="divTableHead">Gender</div> |
|  | <div class="divTableHead">Salary</div> |
|  | </div> |
|  | </div> |
|  | <div class="divTableBody" id="fileGrid"> |
|  |  |
|  |  |
|  | </div> |
|  | </div> |
|  | <div class="blueTable outerTableFooter"><div class="tableFootStyle"><div class="links"><a href="#">&laquo;</a> <a class="active" href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">&raquo;</a></div></div></div> |
|  |  |
|  | `; |

[**view raw**](https://gist.github.com/saregbesola/2c46e42cdbeeaa77bcac1c732b974b75/raw/ef806ec9ba0d4a1ff8229be615d281b81d5d5b97/Public%20Render)[**Public Render**](https://gist.github.com/saregbesola/2c46e42cdbeeaa77bcac1c732b974b75#file-public-render) hosted with ❤ by [**GitHub**](https://github.com/)

This code block does two things. It creates a form to carry out Create and Update operations on our table and creates a table to display data from Azure database.  
  
**Implement CRUD Operations in SPFX**  
1. Still in **YourWebpartName.ts,**update the code block inside **Public render()**as follows

|  |  |
| --- | --- |
|  | public render(): void { |
|  | this.domElement.innerHTML = ` |
|  | <div class="container"> |
|  | <form> |
|  | <div class="row"> |
|  | <h2 style="text-align:left" id="statusMode"> |
|  | Add New Record |
|  | </h4> |
|  | <div class="col-25"> |
|  | <label for="fname">First Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="fname" name="firstname" placeholder="First Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="lname">Last Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="lname" name="lastname" placeholder="Last Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="country">Gender</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="gender" name="gender" placeholder="Gender.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="subject">Salary</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="salary" name="gender" placeholder="Salary.."> |
|  | </div> |
|  | </div> |
|  | <!-- hidden controls --> |
|  | <div style="display: none"> |
|  | <input id="recordId" /> |
|  | </div> |
|  | <div class="row"> |
|  | <input type="submit" value="Submit" id="btnSubmit"> |
|  | </div> |
|  | </form> |
|  | </div> |
|  |  |
|  | <div class="divTable blueTable"> |
|  | <div class="divTableHeading"> |
|  | <div class="divTableRow"> |
|  | <div class="divTableHead">First Name</div> |
|  | <div class="divTableHead">Last Name</div> |
|  | <div class="divTableHead">Gender</div> |
|  | <div class="divTableHead">Salary</div> |
|  | </div> |
|  | </div> |
|  | <div class="divTableBody" id="fileGrid"> |
|  |  |
|  |  |
|  | </div> |
|  | </div> |
|  | <div class="blueTable outerTableFooter"><div class="tableFootStyle"><div class="links"><a href="#">&laquo;</a> <a class="active" href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">&raquo;</a></div></div></div> |
|  |  |
|  | `; |
|  | $(document).ready(function(){ |
|  | PopulateData(); |
|  | $('#statusMode').html('Add New Record'); |
|  | $('#btnSubmit').click(function(e){ |
|  | if($('#statusMode').html()=="Add New Record") |
|  | { |
|  | alert('Add record function'); |
|  | AddNewRecord(); |
|  |  |
|  | } |
|  | else{ |
|  | UpdateRecord($('#recordId').val()); |
|  | } |
|  | }); |
|  | }); |
|  | function PopulateData() |
|  | { |
|  | jquery.ajax({ |
|  | url: "https://yourdemoapi.azurewebsites.net/api/employee", |
|  | type:"GET", |
|  | headers: {"Accept": "application/json; odata=verbose"}, |
|  | success: function(data){ |
|  | if(data){ |
|  | var len = data.length; |
|  | var txt = ""; |
|  | if(len > 0){ |
|  | for(var i=0;i<len;i++){ |
|  |  |
|  | txt += '<div class="divTableRow" ><div class="divTableCell">'+data[i].FirstName+'</div><div class="divTableCell">'+data[i].LastName+'</div>' + |
|  | '<div class="divTableCell">'+data[i].Gender+'</div><div class="divTableCell">'+data[i].Salary+'</div><div class="divTableCell">'+"<a id='" + data[i].ID + "' href='#' class='EditFileLink'>Edit</a>"+'</div><div class="divTableCell">'+"<a id='" + data[i].ID + "' href='#' class='DeleteLink'>Delete</a>"+'</div></div>'; |
|  | } |
|  | if(txt != ""){ |
|  | $("#fileGrid").append(txt); |
|  | } |
|  | } |
|  | } |
|  | }, |
|  |  |
|  | error: function(jqXHR, textStatus, errorThrown){ |
|  | alert('error: ' + textStatus + ': ' + errorThrown); |
|  | } |
|  | }); |
|  | } |
|  | $(document).on('click', '.EditFileLink', function (e) { |
|  | e.preventDefault(); |
|  | var id = this.id; |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | $.ajax({ |
|  | url: requestUri, |
|  | method: "GET", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: { "accept": "application/json;odata=verbose" }, |
|  | success: function (data) { |
|  | $('#fname').val(data.FirstName); |
|  | $('#lname').val(data.LastName); |
|  | $('#gender').val(data.Gender); |
|  | $('#salary').val(data.Salary); |
|  | $('#statusMode').html('Edit Record'); |
|  | $('#recordId').val(data.ID); |
|  | } |
|  | }); |
|  | }); |
|  | $(document).on('click', '.DeleteLink', function (e) { |
|  | e.preventDefault(); |
|  | var id = this.id; |
|  | var confirmDelete = confirm('Confirm deletion'); |
|  | if(confirmDelete){ |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | $.ajax({ |
|  | url: requestUri, |
|  | method: "DELETE", |
|  | headers: { "accept": "application/json;odata=verbose" }, |
|  | success: function () { |
|  | alert('Record deleted successfully'); |
|  | } |
|  | }); |
|  | location.reload(true); |
|  | } |
|  | else{ |
|  |  |
|  | } |
|  | }); |
|  |  |
|  | // Add new record |
|  | function AddNewRecord() { |
|  | var firstName = $("#fname").val(); |
|  | var lastName = $("#lname").val(); |
|  | var gender = $("#gender").val(); |
|  | var salary = $("#salary").val(); |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee"; |
|  | var requestHeaders = { |
|  | "accept": "application/json;odata=verbose", |
|  | } |
|  | var requestData = { |
|  | FirstName: firstName, |
|  | LastName: lastName, |
|  | Gender: gender, |
|  | Salary: salary |
|  | }; |
|  | var requestBody = JSON.stringify(requestData); |
|  |  |
|  | jquery.ajax({ |
|  | url: requestUri, |
|  | method: "POST", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: requestHeaders, |
|  | data: requestBody, |
|  | success: function () |
|  | { |
|  | alert('Record successfully updated'); |
|  |  |
|  | }, |
|  | error: function(jqXHR){ |
|  | alert('error: ' + jqXHR.responseText); |
|  | } |
|  | }); |
|  | } |
|  | //update record |
|  | function UpdateRecord(id) { |
|  | var firstName = $("#fname").val(); |
|  | var lastName = $("#lname").val(); |
|  | var gender = $("#gender").val(); |
|  | var salary = $("#salary").val(); |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | var requestHeaders = { |
|  | "accept": "application/json;odata=verbose", |
|  | } |
|  | var requestData = { |
|  | FirstName: firstName, |
|  | LastName: lastName, |
|  | Gender: gender, |
|  | Salary: salary |
|  | }; |
|  | var requestBody = JSON.stringify(requestData); |
|  |  |
|  | jquery.ajax({ |
|  | url: requestUri, |
|  | method: "PUT", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: requestHeaders, |
|  | data: requestBody, |
|  | success: function () |
|  | { |
|  | alert('Record successfully updated'); |
|  |  |
|  | }, |
|  | error: function(jqXHR){ |
|  | alert('error: ' + jqXHR.responseText); |
|  | } |
|  | }); |
|  | } |
|  | } |

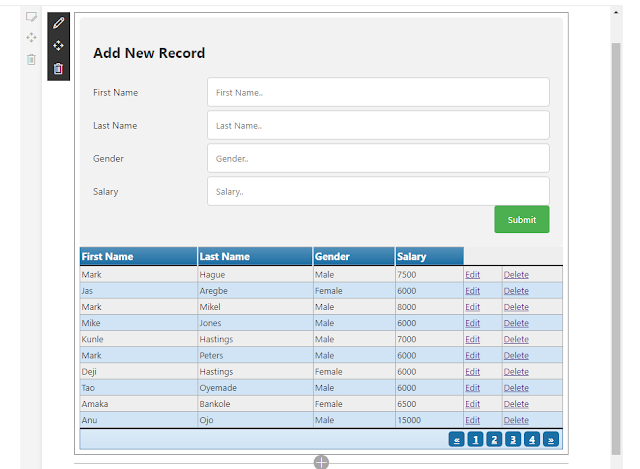
[**view raw**](https://gist.github.com/saregbesola/ecc7fe2f97cd4e694551cf5f4e648f30/raw/2930960f140c09c51c533f96ac7fee12ec652f81/Public%20Render%20updated)[**Public Render updated**](https://gist.github.com/saregbesola/ecc7fe2f97cd4e694551cf5f4e648f30#file-public-render-updated) hosted with ❤ by [**GitHub**](https://github.com/)

This code implements get, create, update and delete operations using PopulateData, AddNewRecord, UpdateRecord and DeleteLink functions respectively, to make calls to the API hosted on Azure.  
  
Your complete code should look like this:

|  |  |
| --- | --- |
|  | import { Version } from '@microsoft/sp-core-library'; |
|  | import { |
|  | BaseClientSideWebPart, |
|  | IPropertyPaneConfiguration, |
|  | PropertyPaneTextField |
|  | } from '@microsoft/sp-webpart-base'; |
|  | import { escape } from '@microsoft/sp-lodash-subset'; |
|  | import \* as jquery from 'jquery'; |
|  | //import styles from './WebApiDemoWebPartWebPart.module.scss'; |
|  | import \* as strings from 'WebApiDemoWebPartWebPartStrings'; |
|  | import { SPComponentLoader } from '@microsoft/sp-loader'; |
|  |  |
|  | SPComponentLoader.loadCss('https://yourspsite.sharepoint.com/sites/dev/SiteAssets/styles.css'); |
|  |  |
|  | export interface IWebApiDemoWebPartWebPartProps { |
|  | description: string; |
|  | } |
|  |  |
|  | export default class WebApiDemoWebPartWebPart extends BaseClientSideWebPart<IWebApiDemoWebPartWebPartProps> { |
|  |  |
|  | public render(): void { |
|  | this.domElement.innerHTML = ` |
|  | <div class="container"> |
|  | <form> |
|  | <div class="row"> |
|  | <h2 style="text-align:left" id="statusMode"> |
|  | Add New Record |
|  | </h4> |
|  | <div class="col-25"> |
|  | <label for="fname">First Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="fname" name="firstname" placeholder="First Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="lname">Last Name</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="lname" name="lastname" placeholder="Last Name.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="country">Gender</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="gender" name="gender" placeholder="Gender.."> |
|  | </div> |
|  | </div> |
|  | <div class="row"> |
|  | <div class="col-25"> |
|  | <label for="subject">Salary</label> |
|  | </div> |
|  | <div class="col-75"> |
|  | <input type="text" id="salary" name="gender" placeholder="Salary.."> |
|  | </div> |
|  | </div> |
|  | <!-- hidden controls --> |
|  | <div style="display: none"> |
|  | <input id="recordId" /> |
|  | </div> |
|  | <div class="row"> |
|  | <input type="submit" value="Submit" id="btnSubmit"> |
|  | </div> |
|  | </form> |
|  | </div> |
|  |  |
|  | <div class="divTable blueTable"> |
|  | <div class="divTableHeading"> |
|  | <div class="divTableRow"> |
|  | <div class="divTableHead">First Name</div> |
|  | <div class="divTableHead">Last Name</div> |
|  | <div class="divTableHead">Gender</div> |
|  | <div class="divTableHead">Salary</div> |
|  | </div> |
|  | </div> |
|  | <div class="divTableBody" id="fileGrid"> |
|  |  |
|  |  |
|  | </div> |
|  | </div> |
|  | <div class="blueTable outerTableFooter"><div class="tableFootStyle"><div class="links"><a href="#">&laquo;</a> <a class="active" href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">&raquo;</a></div></div></div> |
|  |  |
|  | `; |
|  | $(document).ready(function(){ |
|  | PopulateData(); |
|  | $('#statusMode').html('Add New Record'); |
|  | $('#btnSubmit').click(function(e){ |
|  | if($('#statusMode').html()=="Add New Record") |
|  | { |
|  | alert('Add record function'); |
|  | AddNewRecord(); |
|  |  |
|  | } |
|  | else{ |
|  | UpdateRecord($('#recordId').val()); |
|  | } |
|  | }); |
|  | }); |
|  |  |
|  | //bind data to the table |
|  | function PopulateData() |
|  | { |
|  | jquery.ajax({ |
|  | url: "https://yourdemoapi.azurewebsites.net/api/employee", |
|  | type:"GET", |
|  | headers: {"Accept": "application/json; odata=verbose"}, |
|  | success: function(data){ |
|  | if(data){ |
|  | var len = data.length; |
|  | var txt = ""; |
|  | if(len > 0){ |
|  | for(var i=0;i<len;i++){ |
|  |  |
|  | txt += '<div class="divTableRow" ><div class="divTableCell">'+data[i].FirstName+'</div><div class="divTableCell">'+data[i].LastName+'</div>' + |
|  | '<div class="divTableCell">'+data[i].Gender+'</div><div class="divTableCell">'+data[i].Salary+'</div><div class="divTableCell">'+"<a id='" + data[i].ID + "' href='#' class='EditFileLink'>Edit</a>"+'</div><div class="divTableCell">'+"<a id='" + data[i].ID + "' href='#' class='DeleteLink'>Delete</a>"+'</div></div>'; |
|  | } |
|  | if(txt != ""){ |
|  | $("#fileGrid").append(txt); |
|  | } |
|  | } |
|  | } |
|  | }, |
|  |  |
|  | error: function(jqXHR, textStatus, errorThrown){ |
|  | alert('error: ' + textStatus + ': ' + errorThrown); |
|  | } |
|  | }); |
|  | } |
|  |  |
|  | //load and bind data to form in edit mode |
|  | $(document).on('click', '.EditFileLink', function (e) { |
|  | e.preventDefault(); |
|  | var id = this.id; |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | $.ajax({ |
|  | url: requestUri, |
|  | method: "GET", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: { "accept": "application/json;odata=verbose" }, |
|  | success: function (data) { |
|  | $('#fname').val(data.FirstName); |
|  | $('#lname').val(data.LastName); |
|  | $('#gender').val(data.Gender); |
|  | $('#salary').val(data.Salary); |
|  | $('#statusMode').html('Edit Record'); |
|  | $('#recordId').val(data.ID); |
|  | } |
|  | }); |
|  | }); |
|  |  |
|  | //delete record |
|  | $(document).on('click', '.DeleteLink', function (e) { |
|  | e.preventDefault(); |
|  | var id = this.id; |
|  | var confirmDelete = confirm('Confirm deletion'); |
|  | if(confirmDelete){ |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | $.ajax({ |
|  | url: requestUri, |
|  | method: "DELETE", |
|  | headers: { "accept": "application/json;odata=verbose" }, |
|  | success: function () { |
|  | alert('Record deleted successfully'); |
|  | } |
|  | }); |
|  | location.reload(true); |
|  | } |
|  | else{ |
|  |  |
|  | } |
|  | }); |
|  |  |
|  | // Add new record |
|  | function AddNewRecord() { |
|  | var firstName = $("#fname").val(); |
|  | var lastName = $("#lname").val(); |
|  | var gender = $("#gender").val(); |
|  | var salary = $("#salary").val(); |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee"; |
|  | var requestHeaders = { |
|  | "accept": "application/json;odata=verbose", |
|  | } |
|  | var requestData = { |
|  | FirstName: firstName, |
|  | LastName: lastName, |
|  | Gender: gender, |
|  | Salary: salary |
|  | }; |
|  | var requestBody = JSON.stringify(requestData); |
|  |  |
|  | jquery.ajax({ |
|  | url: requestUri, |
|  | method: "POST", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: requestHeaders, |
|  | data: requestBody, |
|  | success: function () |
|  | { |
|  | alert('Record successfully updated'); |
|  |  |
|  | }, |
|  | error: function(jqXHR){ |
|  | alert('error: ' + jqXHR.responseText); |
|  | } |
|  | }); |
|  | } |
|  | //update record |
|  | function UpdateRecord(id) { |
|  | var firstName = $("#fname").val(); |
|  | var lastName = $("#lname").val(); |
|  | var gender = $("#gender").val(); |
|  | var salary = $("#salary").val(); |
|  | var requestUri = "https://yourdemoapi.azurewebsites.net/api/employee/"+id+""; |
|  | var requestHeaders = { |
|  | "accept": "application/json;odata=verbose", |
|  | } |
|  | var requestData = { |
|  | FirstName: firstName, |
|  | LastName: lastName, |
|  | Gender: gender, |
|  | Salary: salary |
|  | }; |
|  | var requestBody = JSON.stringify(requestData); |
|  |  |
|  | jquery.ajax({ |
|  | url: requestUri, |
|  | method: "PUT", |
|  | contentType: "application/json;odata=verbose", |
|  | headers: requestHeaders, |
|  | data: requestBody, |
|  | success: function () |
|  | { |
|  | alert('Record successfully updated'); |
|  |  |
|  | }, |
|  | error: function(jqXHR){ |
|  | alert('error: ' + jqXHR.responseText); |
|  | } |
|  | }); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | protected get dataVersion(): Version { |
|  | return Version.parse('1.0'); |
|  | } |
|  |  |
|  | protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration { |
|  | return { |
|  | pages: [ |
|  | { |
|  | header: { |
|  | description: strings.PropertyPaneDescription |
|  | }, |
|  | groups: [ |
|  | { |
|  | groupName: strings.BasicGroupName, |
|  | groupFields: [ |
|  | PropertyPaneTextField('description', { |
|  | label: strings.DescriptionFieldLabel |
|  | }) |
|  | ] |
|  | } |
|  | ] |
|  | } |
|  | ] |
|  | }; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/saregbesola/16e61aab941082eae736ce6a6c991f4e/raw/af432b4e987f7128d641e8525c6fc36ae24cc64d/Complete%20YourWebpartName.ts)[**Complete YourWebpartName.ts**](https://gist.github.com/saregbesola/16e61aab941082eae736ce6a6c991f4e#file-complete-yourwebpartname-ts) hosted with ❤ by [**GitHub**](https://github.com/)

Ensure you replace the urls in the code with the url for your API  
  
Now it is time to test our web part:  
  
in command prompt, go to the directory for your web part and type one after the other  
***gulp build***  
***gulp serve***  
**gulp serve**launches your browser and opens  local workbench. Click the plus sign and add your web part. If everything works fine, the web part will display your data.  
Now go to your dev SharePointOnline, append ***\_layouts/15/workbench.aspx***to your url. Something like***https://yourspsite.sharepoint.com/sites/dev/\_layouts/15/workbench.aspx.***  
Click the plus sign and look for your web part. If everything works fine, your web part should display data from your azure database. Your web part should look like the image below if everything is fine:

[](https://1.bp.blogspot.com/-Qpu2RQx_ZAw/WlR9v-DQVeI/AAAAAAAACBY/dyHoAxEz7AYuq5fhK9IcSpf4IT1GyhWBQCLcBGAs/s1600/Capture.PNG)

Congratulations!! you have succeeded in displaying your existing data on SQL server in SharePointOnline. 