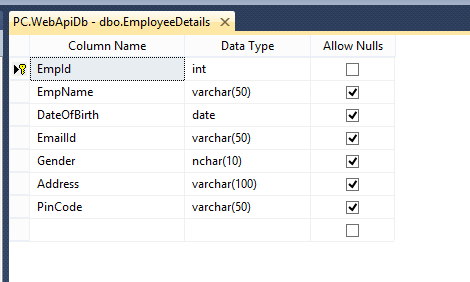
**Step 1. Create a database table**

Create a database. Open SQL Server and create a new database table. As you can see from the following image, I create a database table called EmployeeDetails with 7 columns.

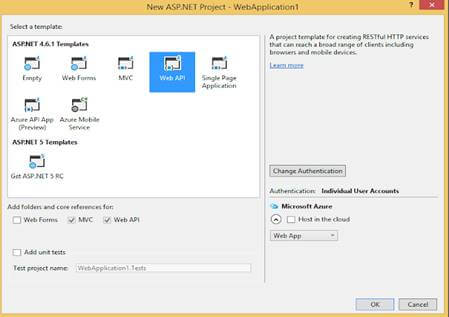


Note: If you already have an existing database and table, you can skip this step.

**Step 2. Create a Web API Project**

Now, we will create a Web API with the functionality of Create, Replace, Update, and Delete (CRUD) operations.

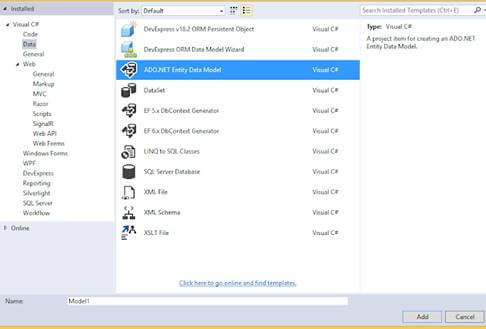
Open Visual Studio >> File >> New >> Project >> Select Web Application. After that click OK and you will see the templates. Select the Web API template.



Click OK.

**Step 3. Add ADO.NET Entity Data Model**

Now, Select Models folder >> Right click >>Add >> New Item >> select Data in left panel >>ADO.NET Entity Data Model,



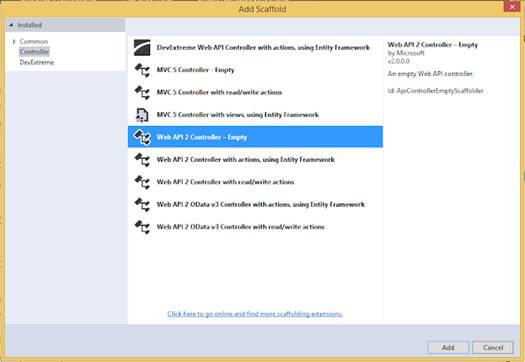
Now click Add button then select EF Designer from database >> Next >> After that give your SQL credential and select the database where your database table and data are.

Click the Add button and select your table and click on the Finish button.

**Step 4. CRUD Operations**

Now, we will write code to perform CRUD operation.

Go to the Controller folder in our API Application and right click >> Add >> Controller >> Select Web API 2 Controller-Empty



Now, we will go to the controller class and set the routing to make it more user friendly by writing the below code.

1. using System;
2. using System.Linq;
3. using System.Web.Http;
4. using CRUDAPI.Models;
6. namespace CRUDAPI.Controllers
7. {
8. [RoutePrefix("Api/Employee")]
9. **public** **class** EmployeeAPIController : ApiController
10. {
11. WebApiDbEntities objEntity = **new** WebApiDbEntities();
13. [HttpGet]
14. [Route("AllEmployeeDetails")]
15. **public** IQueryable<EmployeeDetail> GetEmaployee()
16. {
17. **try**
18. {
19. **return** objEntity.EmployeeDetails;
20. }
21. **catch**(Exception)
22. {
23. **throw**;
24. }
25. }
27. [HttpGet]
28. [Route("GetEmployeeDetailsById/{employeeId}")]
29. **public** IHttpActionResult GetEmaployeeById(string employeeId)
30. {
31. EmployeeDetail objEmp = **new** EmployeeDetail();
32. **int** ID = Convert.ToInt32(employeeId);
33. **try**
34. {
35. objEmp = objEntity.EmployeeDetails.Find(ID);
36. **if** (objEmp == **null**)
37. {
38. **return** NotFound();
39. }
41. }
42. **catch** (Exception)
43. {
44. **throw**;
45. }
47. **return** Ok(objEmp);
48. }
50. [HttpPost]
51. [Route("InsertEmployeeDetails")]
52. **public** IHttpActionResult PostEmaployee(EmployeeDetail data)
53. {
55. **if** (!ModelState.IsValid)
56. {
57. **return** BadRequest(ModelState);
58. }
59. **try**
60. {
61. objEntity.EmployeeDetails.Add(data);
62. objEntity.SaveChanges();
63. }
64. **catch**(Exception)
65. {
66. **throw**;
67. }
69. **return** Ok(data);
70. }
72. [HttpPut]
73. [Route("UpdateEmployeeDetails")]
74. **public** IHttpActionResult PutEmaployeeMaster(EmployeeDetail employee)
75. {
76. **if** (!ModelState.IsValid)
77. {
78. **return** BadRequest(ModelState);
79. }
81. **try**
82. {
83. EmployeeDetail objEmp = **new** EmployeeDetail();
84. objEmp = objEntity.EmployeeDetails.Find(employee.EmpId);
85. **if** (objEmp != **null**)
86. {
87. objEmp.EmpName = employee.EmpName;
88. objEmp.Address = employee.Address;
89. objEmp.EmailId = employee.EmailId;
90. objEmp.DateOfBirth = employee.DateOfBirth;
91. objEmp.Gender = employee.Gender;
92. objEmp.PinCode = employee.PinCode;
94. }
95. **int** i = **this**.objEntity.SaveChanges();
97. }
98. **catch**(Exception)
99. {
100. **throw**;
101. }
102. **return** Ok(employee);
103. }
104. [HttpDelete]
105. [Route("DeleteEmployeeDetails")]
106. **public** IHttpActionResult DeleteEmaployeeDelete(**int** id)
107. {
108. //int empId = Convert.ToInt32(id);
109. EmployeeDetail emaployee = objEntity.EmployeeDetails.Find(id);
110. **if** (emaployee == **null**)
111. {
112. **return** NotFound();
113. }
115. objEntity.EmployeeDetails.Remove(emaployee);
116. objEntity.SaveChanges();
118. **return** Ok(emaployee);
119. }
120. }
121. }

As you may see from the above code, it has functionality to add, replace, update, and delete records to the table.

**Step 5. Build UI Application**

Now, we create the Web application in Angular 7 that will consume Web API.

First we have to make sure that we have Angular CLI installed.

Open command prompt and type below code and press ENTER:

*npm install -g @angular/cli*

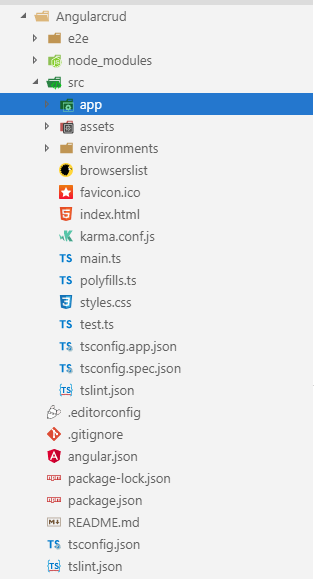
Now, open Visual Studio Code and create a project.

Open TERMINAL in Visual Studio Code and type the following syntax to create a new project. We name it Angularcrud.

*ng new Angularcrud*

After that, hit ENTER. It will take a while to create the project.

Once created, the project should look like this.



Now, we can create some components to provide the UI.

I'm going to create a new component, Employee.

Go to the TERMINAL and go our angular project location using the following command:

*cd projectName*

**CRUD Operation In Angular 7 Using Web API In .NET And Database Of SQL Server**

Now, write the following command that will create a component.

*ng g c employee*

Press ENTER.

Note: you can use see the component is created.

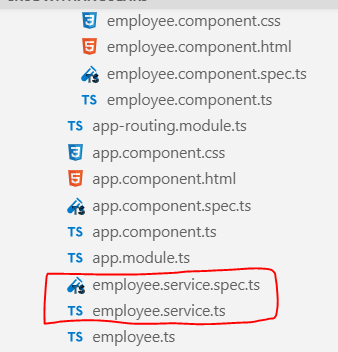
**Step 6. Create a Service**

Now, we will create a service.

Open theTERMINAL and write the below command:

*ng g s employee*

Press ENTER and you will see two service files.



Now, we create a class like model class.

Open TERMINAL and write the below command:

*ng g class employee*

Now, write all properties of the Employee class related to an employee that matches with the database.

1. **export** **class** Employee {
2. EmpId: string;
3. EmpName: string;
4. DateOfBirth: Date;
5. EmailId: string;
6. Gender: string;
7. Address: string;
8. PinCode: string;
9. }

Now, open employee.service.tsand first import necessary class and libraries and then make calls to the WebAPI methods.

1. **import** { Injectable } from '@angular/core';
2. **import** { HttpClient } from '@angular/common/http';
3. **import** { HttpHeaders } from '@angular/common/http';
4. **import** { Observable } from 'rxjs';
5. **import** { Employee } from './employee';
7. After that we write all methods related to consume web **in** employee.service.ts
8. @Injectable({
9. providedIn: 'root'
10. })
12. **export** **class** EmployeeService {
13. url = 'http://localhost:65389/Api/Employee';
14. constructor(**private** http: HttpClient) { }
15. getAllEmployee(): Observable<Employee[]> {
16. **return** **this**.http.get<Employee[]>(**this**.url + '/AllEmployeeDetails');
17. }
18. getEmployeeById(employeeId: string): Observable<Employee> {
19. **return** **this**.http.get<Employee>(**this**.url + '/GetEmployeeDetailsById/' + employeeId);
20. }
21. createEmployee(employee: Employee): Observable<Employee> {
22. **const** httpOptions = { headers: **new** HttpHeaders({ 'Content-Type': 'application/json'}) };
23. **return** **this**.http.post<Employee>(**this**.url + '/InsertEmployeeDetails/',
24. employee, httpOptions);
25. }
26. updateEmployee(employee: Employee): Observable<Employee> {
27. **const** httpOptions = { headers: **new** HttpHeaders({ 'Content-Type': 'application/json'}) };
28. **return** **this**.http.put<Employee>(**this**.url + '/UpdateEmployeeDetails/',
29. employee, httpOptions);
30. }
31. deleteEmployeeById(employeeid: string): Observable<number> {
32. **const** httpOptions = { headers: **new** HttpHeaders({ 'Content-Type': 'application/json'}) };
33. **return** **this**.http.**delete**<number>(**this**.url + '/DeleteEmployeeDetails?id=' +employeeid,
34. httpOptions);
35. }
36. }

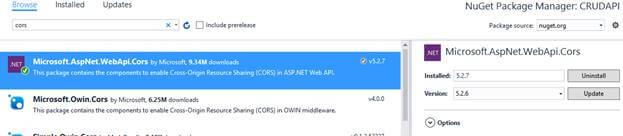
Our service is completed now.

If you consume the Web API, Angular blocks the URL and we called this issue CORS(Cross OriginResource Sharing).

**First, let's resolve this problem.**

Go to the Web API project.

Download a Nuget package for CORS. Go to NuGet Package Manager and download the following file.



After that, go to App\_Start folder in Web API project and open WebApiConfig.cs class. Here, modify the Register method with the below code.

1. Add namespace
2. using System.Web.Http.Cors;
3. **var** cors = **new** EnableCorsAttribute("\*","\*","\*");//origins,headers,methods
4. config.EnableCors(cors);

**Step 7. Install and Configure Angular Material Theme**

As I said earlier, we will use the Angular Material theme to create a rich, interactive, and device-oriented UI for our Web app.

Let's install Install Angular Material theme.

Open TERMINAL again and write the below command:

*npm install --save @angular/material @angular/cdk @angular/animations*

If you want to learn more about Angular Material, visit here: [link](https://material.angular.io/guide/getting-started).

After installed successfully, we can check in package.json file.



Now, let's all required libraries in app.module.ts. We also import a date picker because we'll use the date picker for date of birth field.

Now, open app.module.ts class and write the below code.

1. **import** { BrowserModule } from '@angular/platform-browser';
2. **import** { NgModule } from '@angular/core';
3. **import** { EmployeeService } from './employee.service';
4. **import** { FormsModule, ReactiveFormsModule } from '@angular/forms';
5. **import** { HttpClientModule, HttpClient } from '@angular/common/http';
6. **import** {
7. MatButtonModule, MatMenuModule, MatDatepickerModule,MatNativeDateModule , MatIconModule, MatCardModule, MatSidenavModule,MatFormFieldModule,
8. MatInputModule, MatTooltipModule, MatToolbarModule
9. } from '@angular/material';
10. **import** { MatRadioModule } from '@angular/material/radio';
11. **import** { BrowserAnimationsModule } from '@angular/platform-browser/animations';
13. **import** { AppRoutingModule } from './app-routing.module';
14. **import** { AppComponent } from './app.component';
15. **import** { EmployeeComponent } from './employee/employee.component';
17. @NgModule({
18. declarations: [
19. AppComponent,
20. EmployeeComponent
21. ],
22. imports: [
23. BrowserModule,
24. FormsModule,
25. ReactiveFormsModule,
26. HttpClientModule,
27. BrowserAnimationsModule,
28. MatButtonModule,
29. MatMenuModule,
30. MatDatepickerModule,
31. MatNativeDateModule,
32. MatIconModule,
33. MatRadioModule,
34. MatCardModule,
35. MatSidenavModule,
36. MatFormFieldModule,
37. MatInputModule,
38. MatTooltipModule,
39. MatToolbarModule,
40. AppRoutingModule
41. ],
42. providers: [HttpClientModule, EmployeeService,MatDatepickerModule],
43. bootstrap: [AppComponent]
44. })
45. **export** **class** AppModule { }

Now, we have to import library in styles.css file.

1. @**import** '@angular/material/prebuilt-themes/indigo-pink.css';

**Step 8. Design HTML**

Let's design our HTML page now.

Open employee.component.html and write the below code.

1. <div **class**="container">
3. <mat-card>
4. <mat-toolbar color="accent">
5. <div align="center" style="color:white;text-align: right;">
6. CRUD operation **in** Angular 7 using Web api and Sql Database
7. </div>
8. </mat-toolbar>
9. <br><br>
10. <mat-card-content>
11. <form [formGroup]="employeeForm"(ngSubmit)="onFormSubmit(employeeForm.value)">
12. <table>
13. <tr>
14. <td **class**="tbl1">
15. <mat-form-field **class**="demo-full-width">
16. <input formControlName="EmpName" matTooltip="Enter Employee Name" matInput placeholder="Employee Name">
17. </mat-form-field>
18. <mat-error>
19. <span \*ngIf="!employeeForm.get('EmpName').value && employeeForm.get('EmpName').touched"></span>
20. </mat-error>
21. </td>
22. <td **class**="tbl1">
23. <mat-form-field **class**="demo-full-width">
24. <input matInput [matDatepicker]="picker"matTooltip="Enter Date Of Birth" formControlName="DateOfBirth"placeholder="Choose Date Of Birth">
25. <mat-datepicker-toggle matSuffix [**for**]="picker"></mat-datepicker-toggle>
26. <mat-datepicker #picker></mat-datepicker>
27. </mat-form-field>
28. <mat-error>
29. <span \*ngIf="!employeeForm.get('DateOfBirth').value && employeeForm.get('DateOfBirth').touched"></span>
30. </mat-error>
31. </td>
32. <td **class**="tbl1">
33. <mat-form-field **class**="demo-full-width">
34. <input formControlName="EmailId" matTooltip="Enter EmailId" matInput placeholder="EmailId">
35. </mat-form-field>
36. <mat-error>
37. <span \*ngIf="!employeeForm.get('EmailId').value && employeeForm.get('EmailId').touched"></span>
38. </mat-error>
39. </td>
40. </tr>
41. <tr>
42. <td **class**="tbl1">
43. <span>Gender</span>
44. <br><br>
45. <mat-radio-group matTooltip="Enter Gender"formControlName="Gender">
46. <mat-radio-button value="0">Male</mat-radio-button>
47. <mat-radio-button value="1">Female</mat-radio-button>
48. </mat-radio-group>
49. <mat-error>
50. <span \*ngIf="!employeeForm.get('Gender').value && employeeForm.get('Gender').touched"></span>
51. </mat-error>
52. </td>
53. <td **class**="tbl1">
54. <mat-form-field **class**="demo-full-width">
55. <input matTooltip="Enter Address"formControlName="Address" matInput placeholder="Address">
56. </mat-form-field>
57. <mat-error>
58. <span \*ngIf="!employeeForm.get('Address').value && employeeForm.get('Address').touched"></span>
59. </mat-error>
60. </td>
61. <td **class**="tbl1">
62. <mat-form-field **class**="demo-full-width">
63. <input formControlName="PinCode" matTooltip="Enter Pine Code" matInput placeholder="PinCode">
64. </mat-form-field>
65. <mat-error>
66. <span \*ngIf="!employeeForm.get('PinCode').value && employeeForm.get('PinCode').touched"></span>
67. </mat-error>
68. </td>
69. </tr>
70. <tr>
71. <td></td>
72. <td  **class**="content-center">
73. <button type="submit" mat-raised-button color="accent"matTooltip="Click Submit Button"[disabled]="!employeeForm.valid">Submit</button>
74. <button type="reset" mat-raised-button color="accent"matTooltip="Click Reset Button" (click)="resetForm()">Reset</button>
75. </td>
76. <td>
77. <p \*ngIf="dataSaved" style="color:rgb(0, 128, 0);font-size:20px;font-weight:bold" Class="success" align="left">
78. {{massage}}
79. </p>
80. </td>
81. </tr>
82. </table>
83. <br><br>
84. <table **class**="table" >
85. <tr ngclass="btn-primary">
86. <th **class**="tbl2">Employee Name</th>
87. <th **class**="tbl2">Date Of Birth</th>
88. <th **class**="tbl2">Email Id</th>
89. <th **class**="tbl2">Gender</th>
90. <th **class**="tbl2">Address</th>
91. <th **class**="tbl2">Pine Code</th>
92. <th **class**="tbl2">Edit</th>
93. <th **class**="tbl2">Delete</th>
94. </tr>
95. <tr \*ngFor="let employee of allEmployees | async">
96. <td **class**="tbl2">{{employee.EmpName}}</td>
97. <td **class**="tbl2">{{employee.DateOfBirth | date }}</td>
98. <td **class**="tbl2">{{employee.EmailId}}</td>
99. <td **class**="tbl2">{{employee.Gender ==0? 'Male' : 'Female'}}</td>
100. <td **class**="tbl2">{{employee.Address}}</td>
101. <td **class**="tbl2">{{employee.PinCode}}</td>
102. <td **class**="tbl2">
103. <button type="button" **class**="btn btn-info"matTooltip="Click Edit Button"(click)="loadEmployeeToEdit(employee.EmpId)">Edit</button>
104. </td>
105. <td **class**="tbl2">
106. <button type="button" **class**="btn btn-danger"matTooltip="Click Delete Button"(click)="deleteEmployee(employee.EmpId)">Delete</button>
107. </td>
108. </tr>
110. </table>
111. </form>
112. </mat-card-content>
113. </mat-card>
114. </div>

**Step 9**

Open app.component.html and write the below code.

1. <p>
2. <app-employee></app-employee>
3. </p>

**Step 10**

Open employee.component.ts file and write the below code.

1. **import** { Component, OnInit } from '@angular/core';
2. **import** { FormBuilder, Validators } from '@angular/forms';
3. **import** { Observable } from 'rxjs';
4. **import** { EmployeeService } from '../employee.service';
5. **import** { Employee } from '../employee';
7. @Component({
8. selector: 'app-employee',
9. templateUrl: './employee.component.html',
10. styleUrls: ['./employee.component.css']
11. })
12. **export** **class** EmployeeComponent **implements** OnInit {
13. dataSaved = **false**;
14. employeeForm: any;
15. allEmployees: Observable<Employee[]>;
16. employeeIdUpdate = **null**;
17. massage = **null**;
19. constructor(**private** formbulider: FormBuilder, **private** employeeService:EmployeeService) { }
21. ngOnInit() {
22. **this**.employeeForm = **this**.formbulider.group({
23. EmpName: ['', [Validators.required]],
24. DateOfBirth: ['', [Validators.required]],
25. EmailId: ['', [Validators.required]],
26. Gender: ['', [Validators.required]],
27. Address: ['', [Validators.required]],
28. PinCode: ['', [Validators.required]],
29. });
30. **this**.loadAllEmployees();
31. }
32. loadAllEmployees() {
33. **this**.allEmployees = **this**.employeeService.getAllEmployee();
34. }
35. onFormSubmit() {
36. **this**.dataSaved = **false**;
37. **const** employee = **this**.employeeForm.value;
38. **this**.CreateEmployee(employee);
39. **this**.employeeForm.reset();
40. }
41. loadEmployeeToEdit(employeeId: string) {
42. **this**.employeeService.getEmployeeById(employeeId).subscribe(employee=> {
43. **this**.massage = **null**;
44. **this**.dataSaved = **false**;
45. **this**.employeeIdUpdate = employee.EmpId;
46. **this**.employeeForm.controls['EmpName'].setValue(employee.EmpName);
47. **this**.employeeForm.controls['DateOfBirth'].setValue(employee.DateOfBirth);
48. **this**.employeeForm.controls['EmailId'].setValue(employee.EmailId);
49. **this**.employeeForm.controls['Gender'].setValue(employee.Gender);
50. **this**.employeeForm.controls['Address'].setValue(employee.Address);
51. **this**.employeeForm.controls['PinCode'].setValue(employee.PinCode);
52. });
54. }
55. CreateEmployee(employee: Employee) {
56. **if** (**this**.employeeIdUpdate == **null**) {
57. **this**.employeeService.createEmployee(employee).subscribe(
58. () => {
59. **this**.dataSaved = **true**;
60. **this**.massage = 'Record saved Successfully';
61. **this**.loadAllEmployees();
62. **this**.employeeIdUpdate = **null**;
63. **this**.employeeForm.reset();
64. }
65. );
66. } **else** {
67. employee.EmpId = **this**.employeeIdUpdate;
68. **this**.employeeService.updateEmployee(employee).subscribe(() => {
69. **this**.dataSaved = **true**;
70. **this**.massage = 'Record Updated Successfully';
71. **this**.loadAllEmployees();
72. **this**.employeeIdUpdate = **null**;
73. **this**.employeeForm.reset();
74. });
75. }
76. }
77. deleteEmployee(employeeId: string) {
78. **if** (confirm("Are you sure you want to delete this ?")) {
79. **this**.employeeService.deleteEmployeeById(employeeId).subscribe(() => {
80. **this**.dataSaved = **true**;
81. **this**.massage = 'Record Deleted Succefully';
82. **this**.loadAllEmployees();
83. **this**.employeeIdUpdate = **null**;
84. **this**.employeeForm.reset();
86. });
87. }
88. }
89. resetForm() {
90. **this**.employeeForm.reset();
91. **this**.massage = **null**;
92. **this**.dataSaved = **false**;
93. }
94. }

**Step 11. Run**

We have completed all needed code functionality for our CRUD operations. Before running the application, first, make sure to save your work.

Now, let's run the app and see how it works.

Open TERMINAL and write the following command to run the program.

*ng serve -o*

The output looks like the following image. It's a stunning UI created with CRUD operations.