Crud Operations in Angular 7
Using Web API and Angular Material Theme

Target Audience

This article is intended towards beginners in Angular who has basic knowledge in Web Api and C#.

Overview

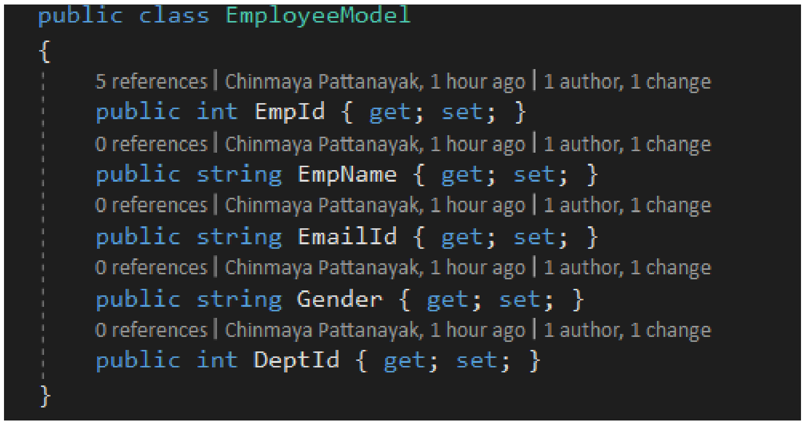
The following areas will be covered in this article.

1. Angular Installation
2. Creation of Angular 7 App
3. Components, Service, Routing etc.
4. UI with Angular Material Theme
5. CRUD Operations in Angular
6. Creation of Api with operations along Json file
7. Deployment of both apps in IIS

The sample application is built using VS 2019, .Net Framework 4.5, Web Api.

Api Project

Create a Web Api project in Visual Studio. Create a folder named **Data**. Add a model "EmployeeModel.cs" with following properties.



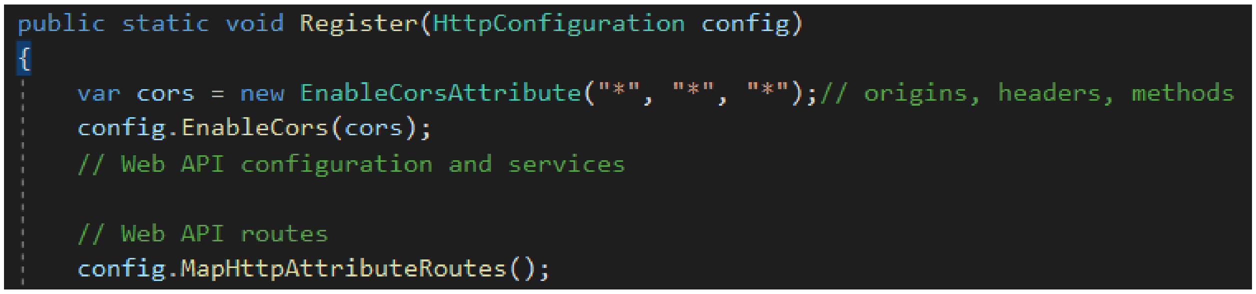
```
public class EmployeeModel
{
    5 references | Chinmaya Pattanayak, 1 hour ago | 1 author, 1 change
    public int EmpId { get; set; }
    0 references | Chinmaya Pattanayak, 1 hour ago | 1 author, 1 change
    public string EmpName { get; set; }
    0 references | Chinmaya Pattanayak, 1 hour ago | 1 author, 1 change
    public string EmailId { get; set; }
    0 references | Chinmaya Pattanayak, 1 hour ago | 1 author, 1 change
    public string Gender { get; set; }
    0 references | Chinmaya Pattanayak, 1 hour ago | 1 author, 1 change
    public int DeptId { get; set; }
}
```

Add a Json file "EmployeeData.json" where we will keep all our Employee related information in Json format and do the CRUD operations.

Add a new Controller as "EmployeeController.cs" and write all the http actions for CRUD functionalities in there.

Now, as we are going to host both angular app and Web Api in different urls, we need to enable CORS (Cross-Origin Resource Sharing) in the api project so that when angular application tries to access resource from api, it won't get any restrictions.

To enable CORS in api project, we need to install CORS package from *nuget* and write the following code in *WebApiConfig.cs*



```
public static void Register(HttpConfiguration config)
{
    var cors = new EnableCorsAttribute("*", "*", "*");// origins, headers, methods
    config.EnableCors(cors);
    // Web API configuration and services

    // Web API routes
    config.MapHttpAttributeRoutes();
```

Now our Api project is ready, run the application and it should run successfully.

<u>Angular Application (UI Project)</u>

Now let's build our angular 7 application which will serve as UI.

Before installing Angular Cli, make sure NodeJS and npm is installed on your computer. If not you can install it from following link.

https://nodejs.org/en/download/

Open command prompt window and run the following command to install latest version of Angular Cli globally.

npm install -g @angular/cli

Once this gets installed successfully, navigate to a directory in command prompt where you want to create the application.

Now run the following command to create angular application.

ng new <YOUR_APP_NAME>

For example – ng new AngularCrudApp. This will create your angular project.

Now navigate to the angular project directory – cd AngularCrudApp and run the following command

ng serve –open

This should compile and run your angular application on default port 4200 (http://localhost:4200). Use Firefox/Chrome to access the angular app. Angular 7 will not work in IE as it renders ES6 components which is not supported by IE.

We will use Angular Material theme to create a rich, interactive and device-oriented UI for our Web app.

Let's install Angular Material theme.

Open command prompt again, navigate to the angular app directory and write the below command:

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

If you want learn more about Angular Material, visit here: link.

Let’s create the service first to access resources through our Web Api.

Run the following command in command prompt.

ng generate service employee *or* ***ng g s employee***

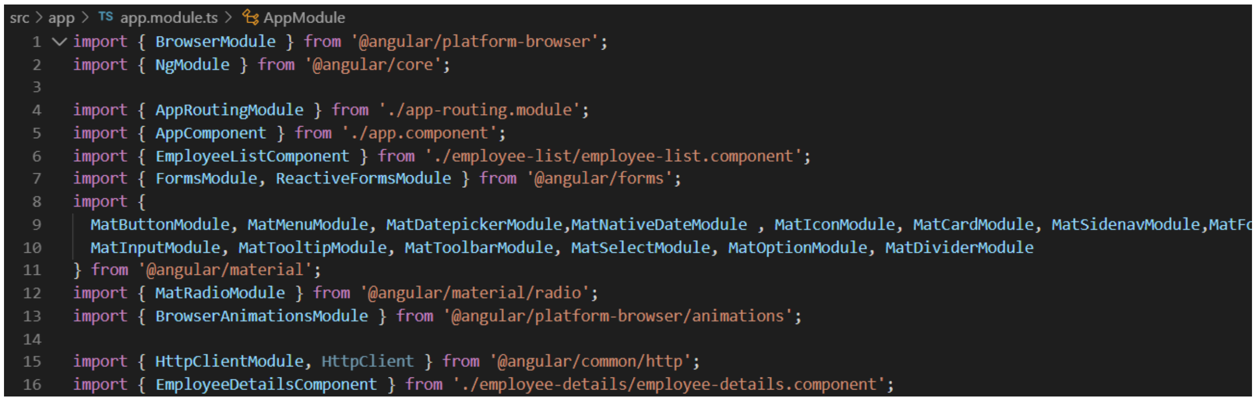
It will create our employee service with two files employee.service.ts and employee.service.spec.ts

The employee.service.ts will contain all the methods which we need during our CRUD operations and consume the exposed api urls from our Api project.

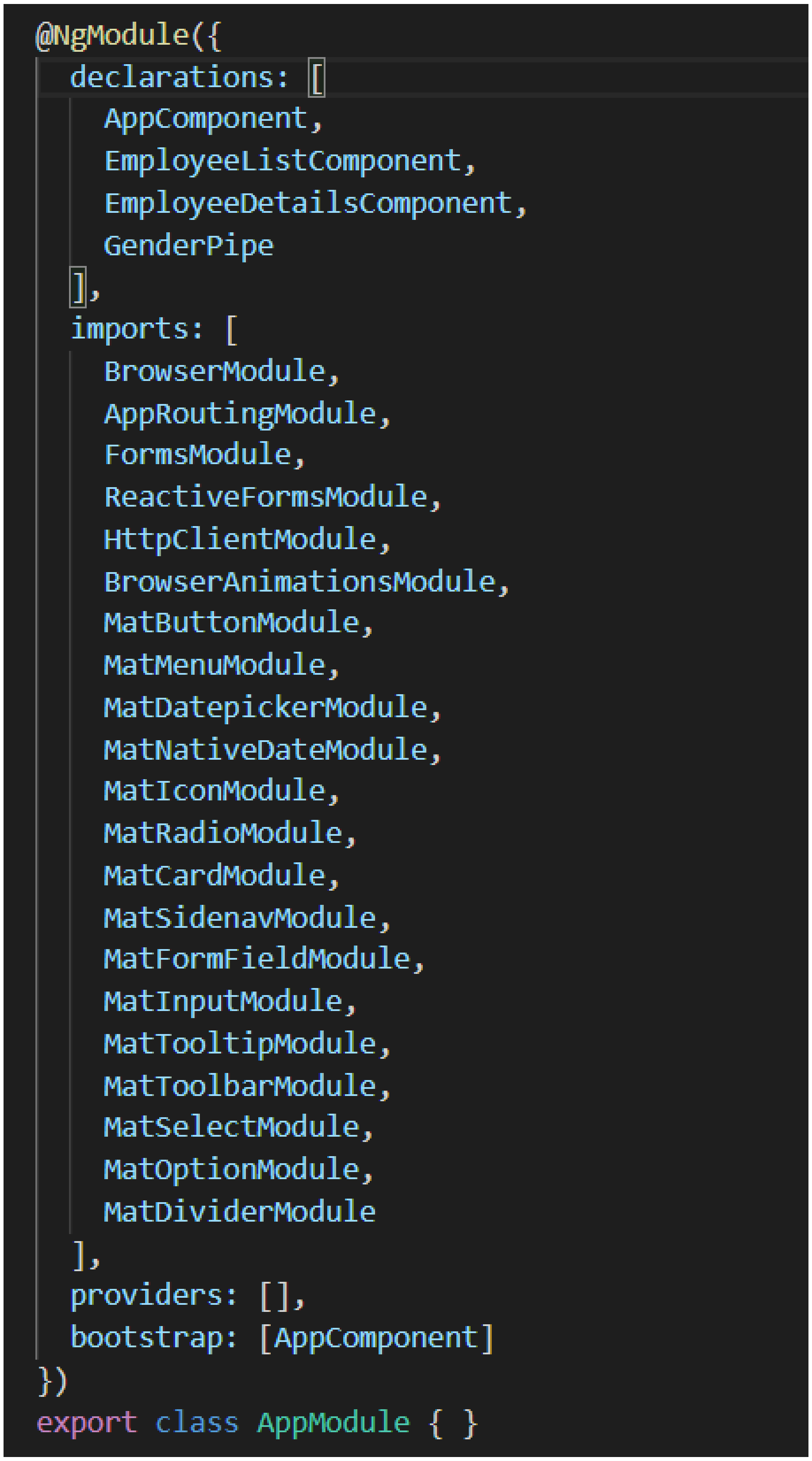
Next we will create a model class similar to our Api Employee Model using following command.

ng g class employee

Now let’s add all required libraries in AppModule.ts

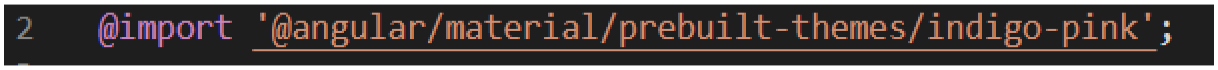


```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import {
  MatButtonModule, MatMenuModule, MatDatepickerModule,MatNativeDateModule , MatIconModule, MatCardModule, MatSidenavModule,MatF
  MatInputModule, MatTooltipModule, MatToolbarModule, MatSelectModule, MatOptionModule, MatDividerModule
} from '@angular/material';
import { MatRadioModule } from '@angular/material/radio';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { HttpClientModule, HttpClient } from '@angular/common/http';
import { EmployeeDetailsComponent } from './employee-details/employee-details.component';
```



```
@NgModule({
  declarations: [
    AppComponent,
    EmployeeListComponent,
    EmployeeDetailsComponent,
    GenderPipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    BrowserAnimationsModule,
    MatButtonModule,
    MatMenuModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatIconModule,
    MatRadioModule,
    MatCardModule,
    MatSidenavModule,
    MatFormFieldModule,
    MatInputModule,
    MatTooltipModule,
    MatToolbarModule,
    MatSelectModule,
    MatOptionModule,
    MatDividerModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

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

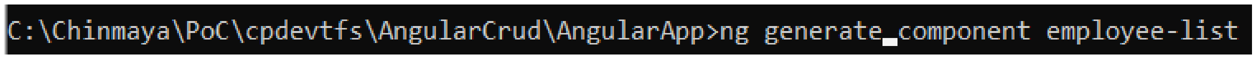


```
@import '@angular/material/prebuilt-themes/indigo-pink';
```

Let's create our first component which will act as input form and display the records which we are going to save through the form.

Run the following command in command prompt to create employee-list component.

ng generate component employee-list



```
C:\Chinmaya\PoC\cpdevtfs\AngularCrud\AngularApp>ng generate component employee-list
```

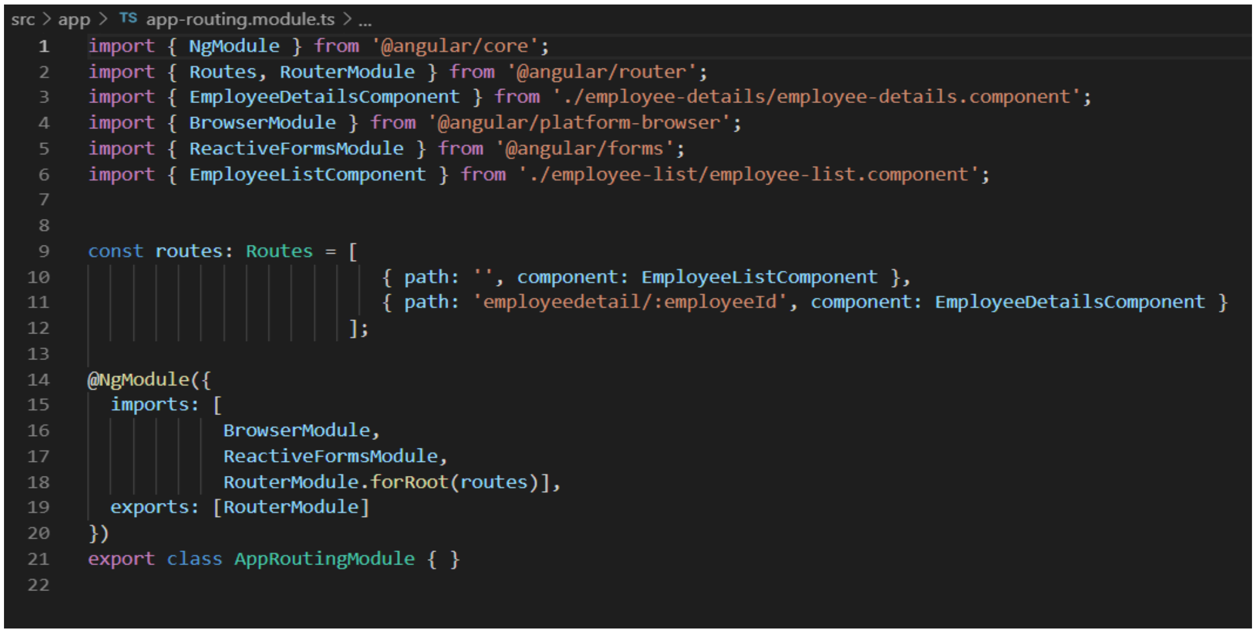
This should generate employee-list component which includes following four files.

- employee-list.component.css
- employee-list.component.html
- employee-list.component.ts
- employee-list.component.spec.ts

We have to throw all our html design into employee-list.component.html and all the events, code behind logic in employee-list.component.ts

Now we are ready with a presentable user interface. We have written the html component, we have written service, and the api of course. But.. how do bring the employee-list html to the first screen because we haven't referenced it in anywhere and the default html which is "AppComponent.html" will be loaded first. So, in order to fix this we have to bring in the most important part of angular which is "Angular Routing".

Let's write the routing to make sure which component/html gets loaded at what time. We'll write all the route details in default file ***app-routing.module.ts***



src > app > TS app-routing.module.ts > ...

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { EmployeeDetailsComponent } from './employee-details/employee-details.component';
import { BrowserModule } from '@angular/platform-browser';
import { ReactiveFormsModule } from '@angular/forms';
import { EmployeeListComponent } from './employee-list/employee-list.component';


const routes: Routes = [
                          { path: '', component: EmployeeListComponent },
                          { path: 'employeedetail/:employeeId', component: EmployeeDetailsComponent }
                       ];

@NgModule({
  imports: [
            BrowserModule,
            ReactiveFormsModule,
            RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

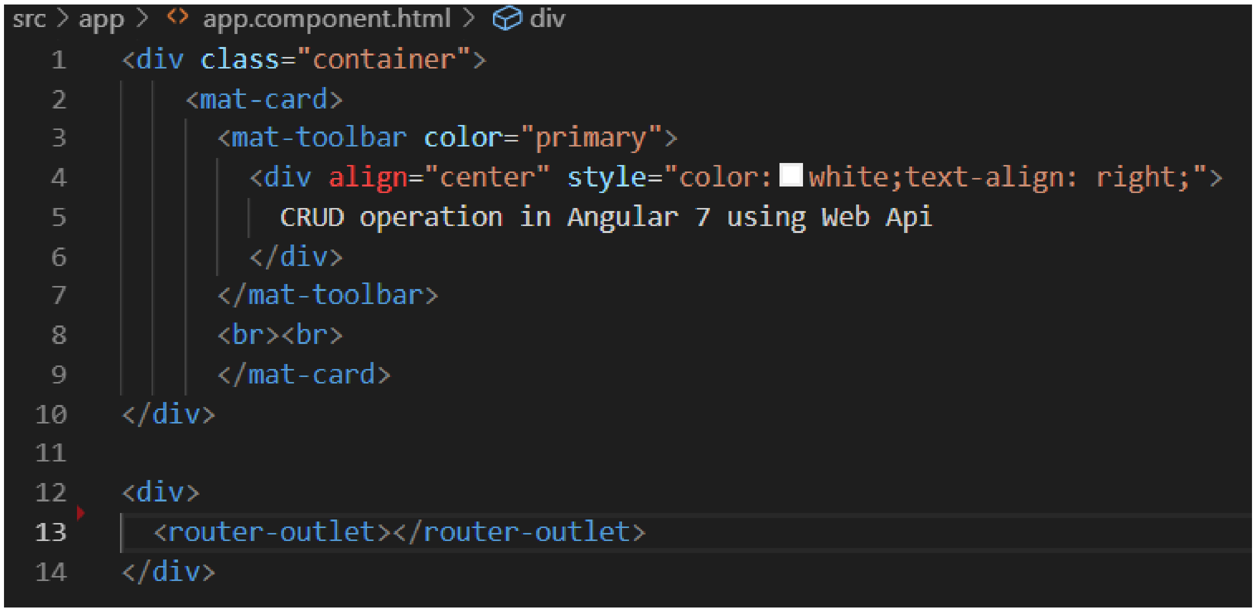
We have to import ReactiveFormsModule as we are using Reactive forms in our employee-list.component.html

To know more about different types of forms click the below link

https://angular.io/guide/forms-overview

After writing the routing details we have to do one more thing, we need to place the **router-outlet** in proper place to render html components when we navigate to a url.

In this case we need to place router-outlet in AppComponent.Html



src > app > app.component.html > div

```html
<div class="container">
    <mat-card>
      <mat-toolbar color="primary">
        <div align="center" style="color: white;text-align: right;">
          CRUD operation in Angular 7 using Web Api
        </div>
      </mat-toolbar>
      <br><br>
      </mat-card>
</div>

<div>
  <router-outlet></router-outlet>
</div>
```

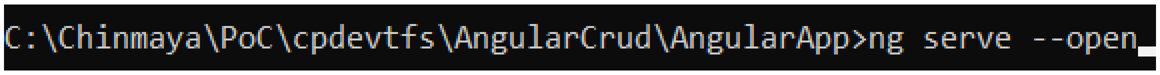
To know more about router-outlet click the below link

https://angular.io/guide/router#router-outlet

Now let run our application and see how it works. Before running the angular application make sure the api application is running.

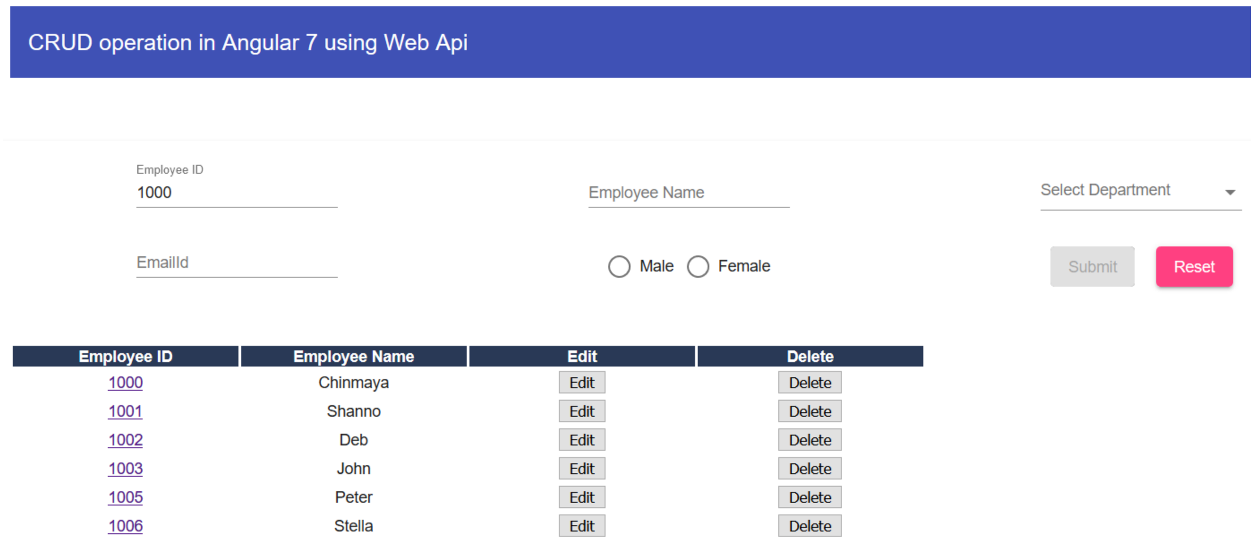
To run the angular application, run the following command in command prompt.

ng serve –open



```
C:\Chinmaya\PoC\cpdevtfs\AngularCrud\AngularApp>ng serve --open
```

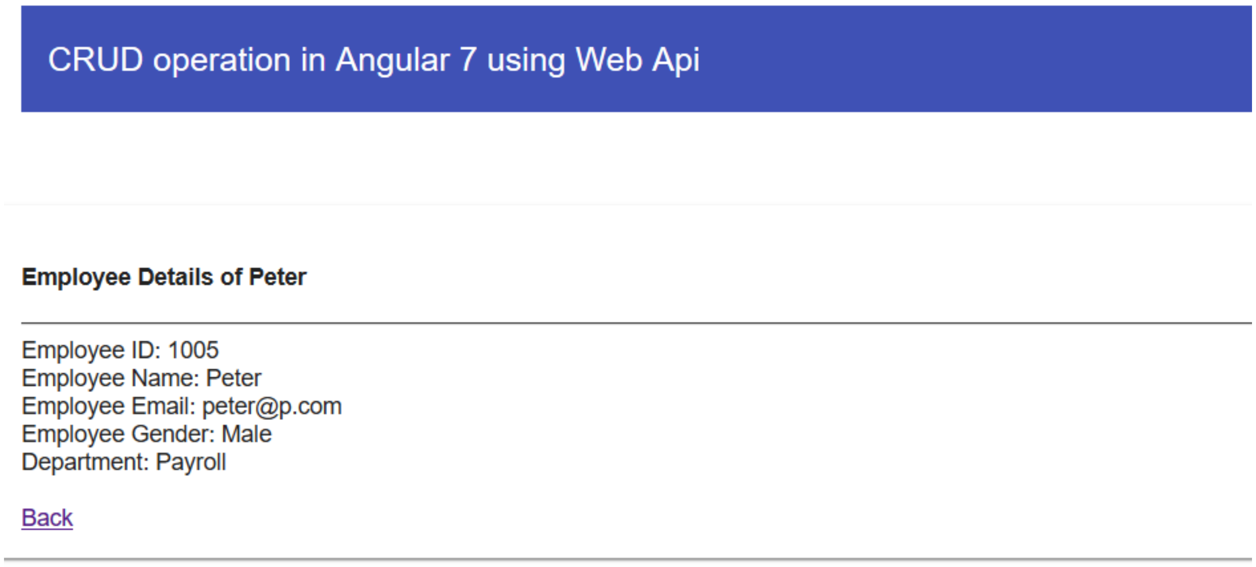
The application looks like something like this.


CRUD operation in Angular 7 using Web Api

Employee ID: 1000	Employee Name	Select Department
EmailId	○ Male ○ Female	Submit / Reset

Employee ID	Employee Name	Edit	Delete
1000	Chinmaya	Edit	Delete
1001	Shanno	Edit	Delete
1002	Deb	Edit	Delete
1003	John	Edit	Delete
1005	Peter	Edit	Delete
1006	Stella	Edit	Delete

Now when you click on the Employee ID hyperlink, it will take us to another screen to show the employee details.

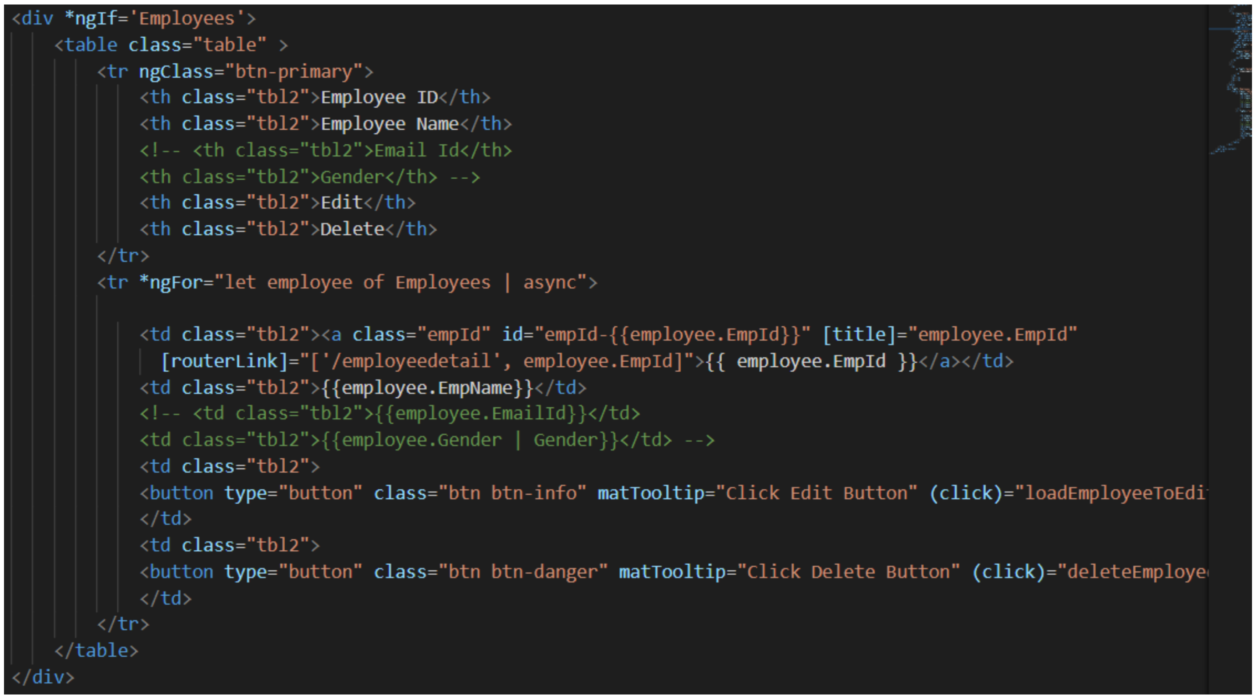

CRUD operation in Angular 7 using Web Api

Employee Details of Peter

Employee ID: 1005
Employee Name: Peter
Employee Email: peter@p.com
Employee Gender: Male
Department: Payroll

Back

To create this employee details screen, let's add one more component and before that put a routerLink in employee-list.component.html.



```
<div *ngIf='Employees'>
    <table class="table" >
        <tr ngClass="btn-primary">
            <th class="tbl2">Employee ID</th>
            <th class="tbl2">Employee Name</th>
            <!-- <th class="tbl2">Email Id</th>
            <th class="tbl2">Gender</th> -->
            <th class="tbl2">Edit</th>
            <th class="tbl2">Delete</th>
        </tr>
        <tr *ngFor="let employee of Employees | async">

            <td class="tbl2"><a class="empId" id="empId-{{employee.EmpId}}" [title]="employee.EmpId"
              [routerLink]="['/employeedetail', employee.EmpId]">{{ employee.EmpId }}</a></td>
            <td class="tbl2">{{employee.EmpName}}</td>
            <!-- <td class="tbl2">{{employee.EmailId}}</td>
            <td class="tbl2">{{employee.Gender | Gender}}</td> -->
            <td class="tbl2">
            <button type="button" class="btn btn-info" matTooltip="Click Edit Button" (click)="loadEmployeeToEdi
            </td>
            <td class="tbl2">
            <button type="button" class="btn btn-danger" matTooltip="Click Delete Button" (click)="deleteEmploye
            </td>
        </tr>
    </table>
</div>
```

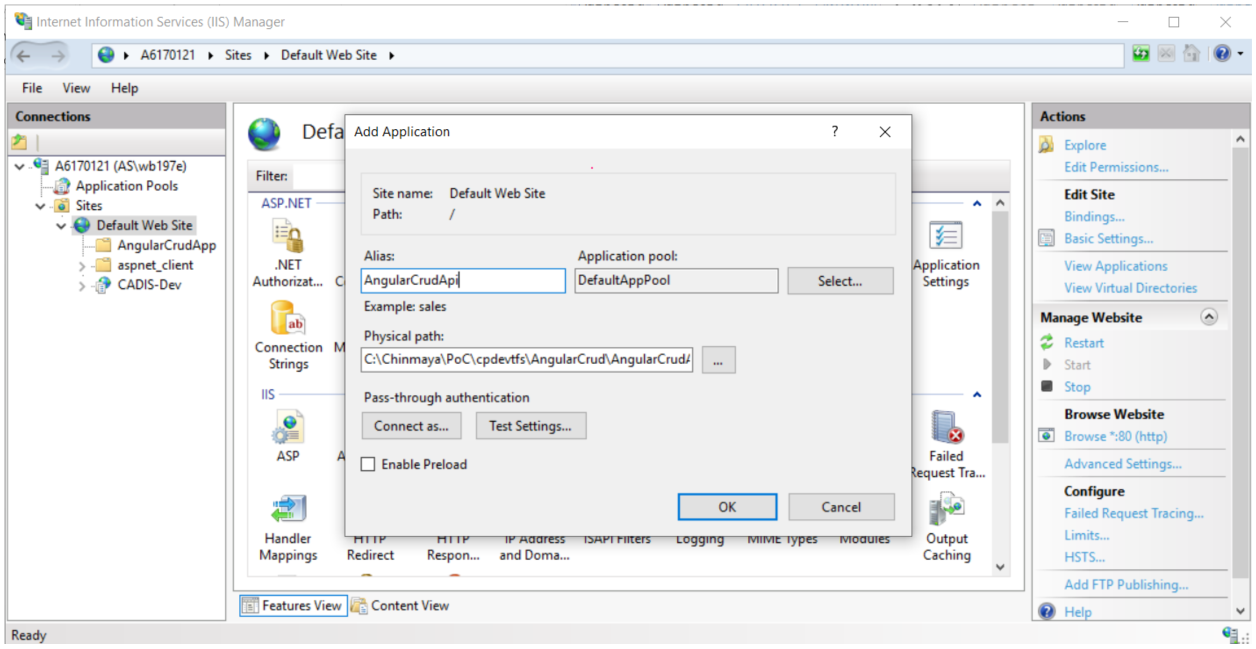
Now generate the below component.

ng generate component employee-details

Write all your html in employee-details.cmponents.html

<u>Deployment</u>

Go to IIS Manager, Right click on Default Web Site and Add Application. Now give a meaningful name and point your Api project physical location.



Add Application

Site name: Default Web Site
Path: /

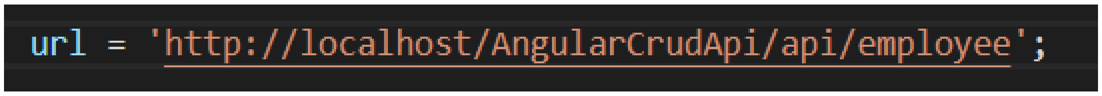
Alias: AngularCrudApi
Application pool: DefaultAppPool
Example: sales
Physical path: C:\Chinmaya\PoC\cpdevtfs\AngularCrud\AngularCrud/

Pass-through authentication

☐ Enable Preload

Click OK and your api will be hosted on IIS now. Right click on the newly added application, go to Manage Application and the Browse. Verify if your api is running successfully or not.

Copy the new api url and replace the old api url with this in angular app employee.service.ts

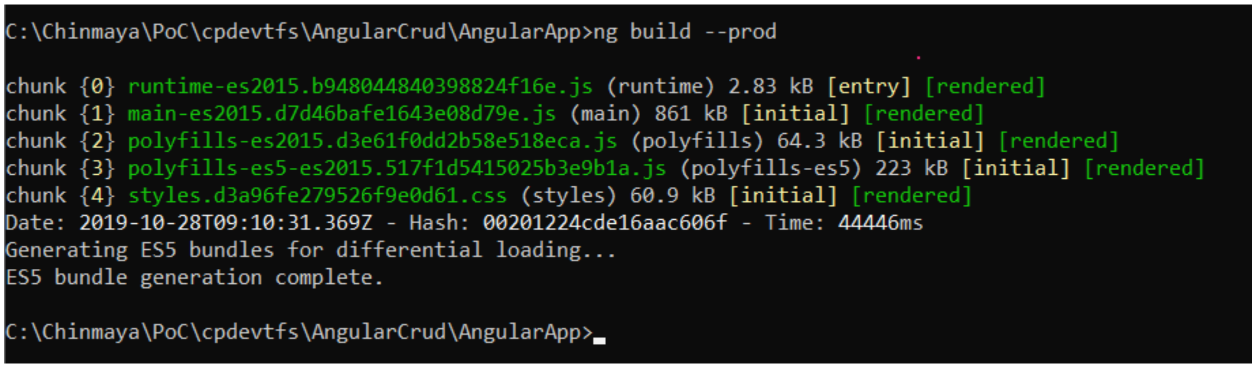


```
url = 'http://localhost/AngularCrudApi/api/employee';
```

Then to deploy angular application, first we need to build it in prod mode with the help of following command.

ng build –prod

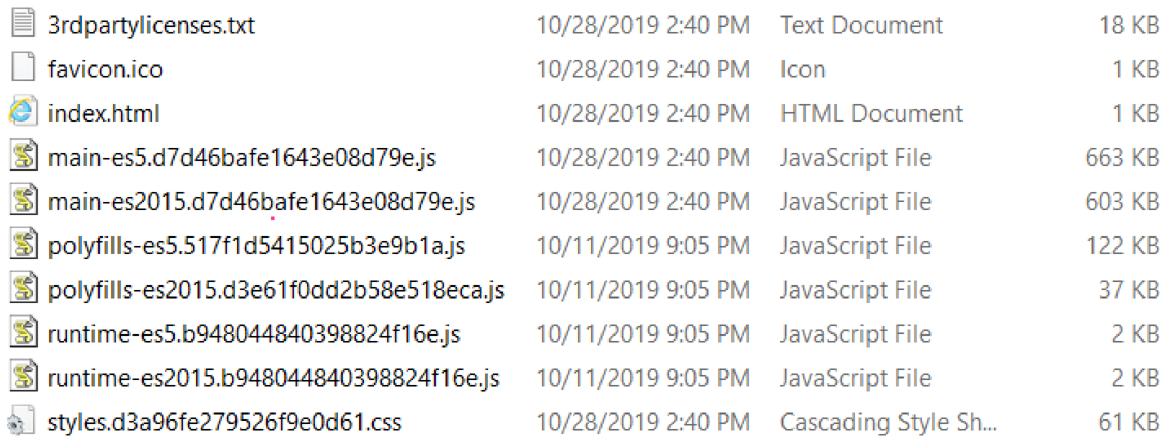
Once build is successful, you will see the following messages in command prompt.



```
C:\Chinmaya\PoC\cpdevtfs\AngularCrud\AngularApp>ng build --prod

chunk {0} runtime-es2015.b948044840398824f16e.js (runtime) 2.83 kB [entry] [rendered]
chunk {1} main-es2015.d7d46bafe1643e08d79e.js (main) 861 kB [initial] [rendered]
chunk {2} polyfills-es2015.d3e61f0dd2b58e518eca.js (polyfills) 64.3 kB [initial] [rendered]
chunk {3} polyfills-es5-es2015.517f1d5415025b3e9b1a.js (polyfills-es5) 223 kB [initial] [rendered]
chunk {4} styles.d3a96fe279526f9e0d61.css (styles) 60.9 kB [initial] [rendered]
Date: 2019-10-28T09:10:31.369Z - Hash: 00201224cde16aac606f - Time: 44446ms
Generating ES5 bundles for differential loading...
ES5 bundle generation complete.

C:\Chinmaya\PoC\cpdevtfs\AngularCrud\AngularApp>
```

Now go to the angular application physical location and there will be folder named as **dist**. Open it.

You will find a directory with your Project name and when you go through it, you will see all the compiled files.



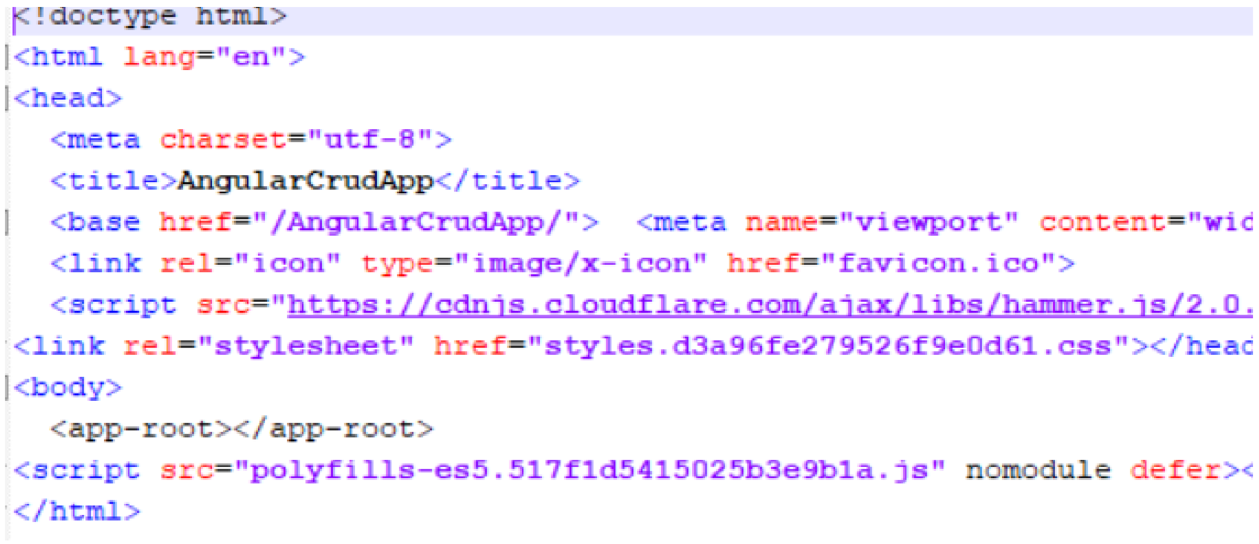
3rdpartylicenses.txt	10/28/2019 2:40 PM	Text Document	18 KB
favicon.ico	10/28/2019 2:40 PM	Icon	1 KB
index.html	10/28/2019 2:40 PM	HTML Document	1 KB
main-es5.d7d46bafe1643e08d79e.js	10/28/2019 2:40 PM	JavaScript File	663 KB
main-es2015.d7d46bafe1643e08d79e.js	10/28/2019 2:40 PM	JavaScript File	603 KB
polyfills-es5.517f1d5415025b3e9b1a.js	10/11/2019 9:05 PM	JavaScript File	122 KB
polyfills-es2015.d3e61f0dd2b58e518eca.js	10/11/2019 9:05 PM	JavaScript File	37 KB
runtime-es5.b948044840398824f16e.js	10/11/2019 9:05 PM	JavaScript File	2 KB
runtime-es2015.b948044840398824f16e.js	10/11/2019 9:05 PM	JavaScript File	2 KB
styles.d3a96fe279526f9e0d61.css	10/28/2019 2:40 PM	Cascading Style Sh...	61 KB

Copy all the files you see above and go to the IIS Root directory **C:\inetpub\wwwroot** and paste all the files in it. Now open the browser and access http://localhost/index.html and you should be able to access your application.

If you want to deploy your app in a sub folder, then you need to specify the directory name during ng build.

ng build --base-href "/AngularCrudApp/" –prod

OR alternatively you can do the normal build (***ng build –prod***) and modify the **<base href/>** attribute in **index.html** with your subfolder name.



```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AngularCrudApp</title>
  <base href="/AngularCrudApp/">  <meta name="viewport" content="wid
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/hammer.js/2.0.
<link rel="stylesheet" href="styles.d3a96fe279526f9e0d61.css"></head
<body>
  <app-root></app-root>
<script src="polyfills-es5.517f1d5415025b3e9b1a.js" nomodule defer><
</html>
```

Copy all the compiled files into the specified sub folder inside www root folder and you app works !!