Create New Project in VS2017, Asp.net Core Web Application >> API, name as “CustService”.

Install following Packages:

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.SqlServer.Design

Create Folder Models and Service.

Execute following command from Package Manager console to generate DBContext and Models.

Scaffold-DbContext "Server=DataSource;Database=CustomerDB;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

this creates CustomerDBContext, Customer,UserLogin Class files in Models Folder.

Comment Method in context file protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder).

Create ICustomerService,IAdminService add required functions for CURD operations.

Create CustomerService,AdminService inherits from ICustomerService, IAdminservice and implements all methods of interface. In this Service Inject CustomerDBContext in constructor.

Add CustomerController,AdminController and create Actions for endpoints. Since there is no much business logic involved, each service has one Http\* action.

**Configuring:**

Add ConnectionStrings(DefaultConnection) in appsettings.json file.

**in Startup.cs file:**

In ConfigureServices function add following lines.

//DI for Services

services.AddScoped<ICustomerService, CustomerService>();

//DI for DBContext

services.AddScoped<CustomerDBContext>();

//Getting connection String from appsettings.json and add to context.

services.AddDbContext<CustomerDBContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

**Swagger Configuration:**

install Packages Swashbuckle.AspNetCore, Swashbuckle.AspNetCore.SwaggerUI to Configure Swaggerand to test APIs.

Add in the Method ConfigureServices :

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Info { Title = "Customer API", Version = "v1" });

});

Add in the Method Configure

// Enable middleware to serve generated Swagger as a JSON endpoint.

app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),

// specifying the Swagger JSON endpoint.

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");

});

**Log Configuration :**

install Package Microsoft.Extensions.Logging.Log4Net.AspNetCore

Add file log4net.config, which has configuration details for logging.

Add to Configure method.

//bind from log4net.config xml file

loggerFactory.AddLog4Net();

<ItemGroup>

<PackageReference Include="Microsoft.AspNetCore.All" Version="2.0.9" />

<PackageReference Include="Microsoft.EntityFrameworkCore" Version="2.1.3" />

<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="2.1.3" />

<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="2.1.3" />

<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer.Design" Version="1.1.6" />

<PackageReference Include="Microsoft.Extensions.Logging.Log4Net.AspNetCore" Version="2.2.10" />

<PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="2.0.4" />

<PackageReference Include="Swashbuckle.AspNetCore" Version="4.0.1" />

<PackageReference Include="Swashbuckle.AspNetCore.SwaggerUI" Version="4.0.1" />

</ItemGroup>

**Implement JWT Security API:**

CustomerController Actions are secured with JWT using [Authorize] attribute. Admincontroller has only one Action authenticate User which is allows anonymous users, without login. [AllowAnonymous] attribute can be used for anonymous access.

**JWT App Settings:**

The AppSettings class contains properties defined in appsettings.json /

Secret can be any GUID or string.

"AppSettings": {

"Secret": "1eff8d6a-360f-48d2-a115-91f1cc2ece5d"

}

Configure appsettings in Startup.cs, in ConfigureServices.

// configure strongly typed settings objects

var appSettingsSection = Configuration.GetSection("AppSettings");

services.Configure<AppSettings>(appSettingsSection);

Mapping of configuration sections is done in the ConfigureServices method of the Startup.cs file. Region JWTConfig.

Run the Application and navigate to url http://localhost:5327/swagger/index.html

**User Authentication:**

AdminService contains method Authenticate which accepts Username, and Password as params to validate user against Table UserLogin.

On successful authentication, generates a JWT (JSON Web Token) using the JwtSecurityTokenHandler classwhich is digitally signed using a secret key stored in appsettings.json. The JWT token is returned to the client application which has to be included in HTTP Authorization header of to access [Authorize] APIs.

Configure JWT for Swagger in Strartup.cs , ConfigureServices method

// Swagger 2.+ support

var security = new Dictionary<string, IEnumerable<string>>

{

{"Bearer", new string[] { }},

};

c.AddSecurityDefinition("Bearer", new ApiKeyScheme

{

Description = "JWT Authorization header using the Bearer scheme. Example: \"Authorization: Bearer {token}\"",

Name = "Authorization",

In = "header",

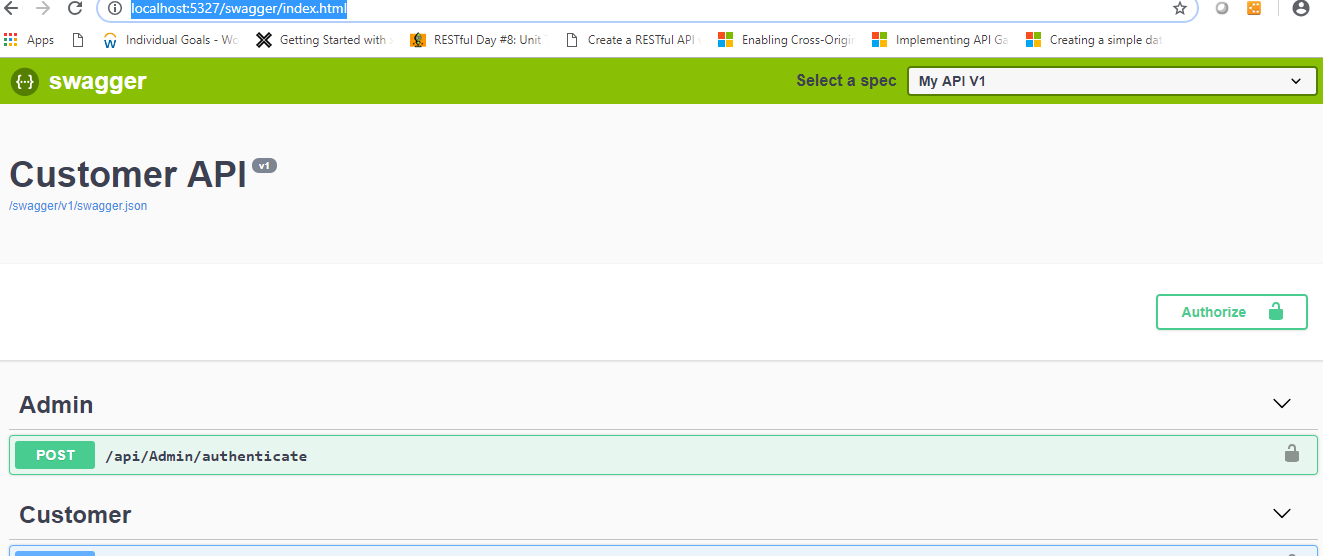
Type = "apiKey"

});

c.AddSecurityRequirement(security);

**Now Let us test our APIs in Swagger.**

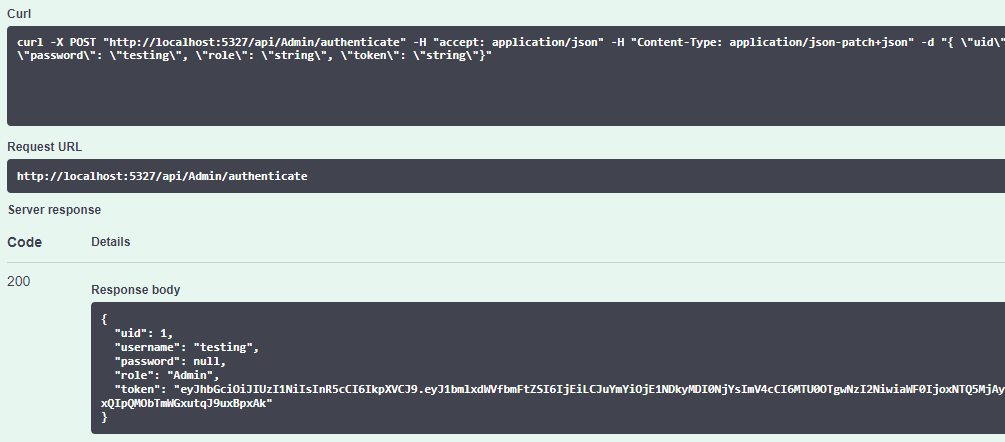
Run the Application and launch url <http://localhost:5327/swagger/index.html>.



Create and sample record in Database table UserLogin, with username=”testing” and password=”testing”.

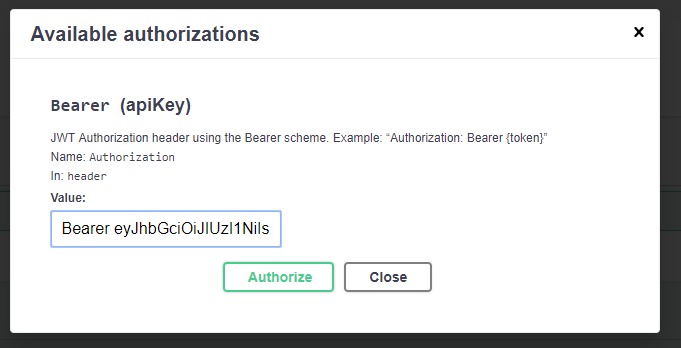
Click on /api/Admin/authenticate, “Try It Out”.

Enter valid username and password in userParam and Execute. Response with status “200” and “token” in Response Body.



Copy the Token, which is used to test [Authorized] Apis.

Click on “Authorize” button and enter “Bearer “ past Token. It looks like following. Click on Authorize button. For subsequent API calls from Swagger, this Bearer token will be added in header for Authorization.



Create customer from Swagger with required information, not to Enter CustID which is Identity Column.

Try testing “GetAllCustomer” API, to very customers are created successfully.

Development of sample Web API has been completed.

**Now We can start developing UI in Angular6.**

Install Angular CLI and verify version "ng --v"

Angular CLI: 6.1.4

Node: 8.11.4

..and other packages installed.

Create Customer UI command: "ng new CustomerUI"

it may take some time to install all packages.

Run following Command for routing Module: for Navigation from List to Create Customer.

ng generate module app-routing --flat --module=app

//Display all Customers

ng g component CustomerList

//Display Customer Details and to add New Customer.

ng g component CustomerDetail

//Login page

Ng g component login

//generate customer class

This application has 2 Models Customer and UserLogin. Create 2 classes and 2 services to communicate with API.

//generate classes

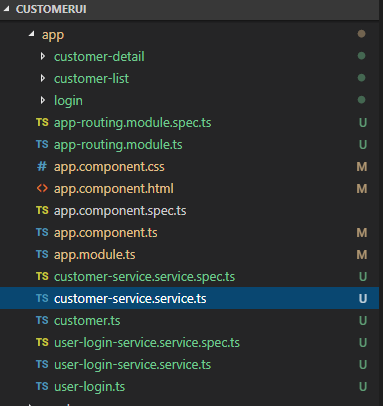
ng g class customer

Ng g class userlogin

Ng g service customerservice

Ng g service userlogin

After creation of required components and Class and service Studio code look as follows.

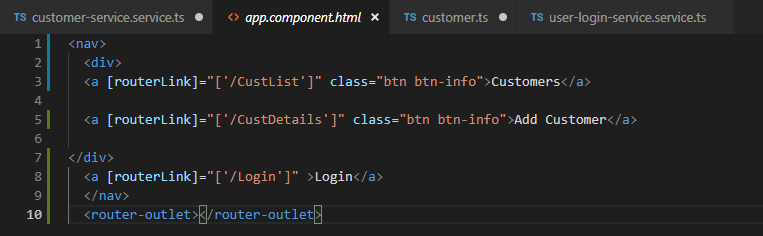


Here I would like to give some Tips for new Angular 4/6 developers who are from Dotnet, Angularjs background. **This may not be so accurate for helpful for beginners.**

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Angular 4/6** | **Dotnet** |
| view | .html | aspx |
| code file | .ts | .aspx..cs |
| adding namespace | import { FormBuilder, FormGroup, Validators } from '@angular/forms'; | using Syste.\* |
| coding language | typescript | C# |
| new page | ng generate componet.  Adds name\*.component.html, component.ts, component.css | add page : .aspx and aspx.cs files |
| Masterpage | app.component.html | acts as Master page |
| Masterpage cs file | app.component.ts | code behind file for master page |
| Reusable forms | Templates | usercontrols |
| start up | main.ts >> app.module.ts | startup.cs, program.cs |
| Add model | ng g class Modelname | add cs file |
| Add Service | ng g service servicename | add cs file |
| Common resources static files storage | Assets | wwwroot or styles or scripts folder. |
| Navigation / Menu | Routing/ nav.  ng generate module app-routing --flat --module=app | Menu / html |
| Navigation Container | routes: Routes and use container <router-outlet></router-outlet> in master / landing page | not available |
| Page Navigation: | TS file : this.router.navigate([this.returnUrl]) html : <a routerLink="/Nav1"> Navigation1 </a> | cs: Response.Redirect, Server.Trasfer html: <a href='' /> |
| Page Lifecycle events | ngOnchanges, ngOnInit, ngDoCheck, ngAfterContentInit, ngAfterContentCheck, ngAfterViewInit, ngAfterViewcheck, ngDestroy | Init, load, validate, eventhandling, prerender, render |
| **Coding:** |  |  |
| Current Instance | this (mandatory to use) | this (optional) |
| debugging | debugger; and this.showSpinner=false; commonly used. Same as jquery. | break point |
| Declaration | custId :number | int CustId |
| Commonly used Data types | number, string, Date, boolean | int, double, float, string, DateTime, boolean. |
| constructor | constructor(protected http: HttpClient) | Same as class name |
| function declaration | method name(params):returntype **example:** authenticateUser(userLogin:UserLogin):Observable<UserLogin> {} onSelect(selCust:Customer) {   } | return type Methodname(params)  **Example:** UserLogin Authenticate(string Username, string Password)  {} |
| condition | if(condition) in html : \*ngIf | if(condition) |
| loop | in ts file: for(let a in array){} html file directive: \*ngFor="let curCust of custList" | foreach(var a in list){} |
| Binding from code to html | {{curCust.custId}} | Server control: controlID.Text |
| Two way binding | [(ngModel)]="variable" | not available |
| accessing DOM html in code file | get f() { return this.FormGroupname.controls; } this.variable=this.f.controlname.value; | runat=server, can access in code behind. |
| Submit form | form Submit (ngSubmit)="onSubmit()" | form postback |
| Event binding | <button (click)="onClickMe()">Click me!</button> <input (keyup)="onKey($event)"> | object.event + = new delegate(method) |

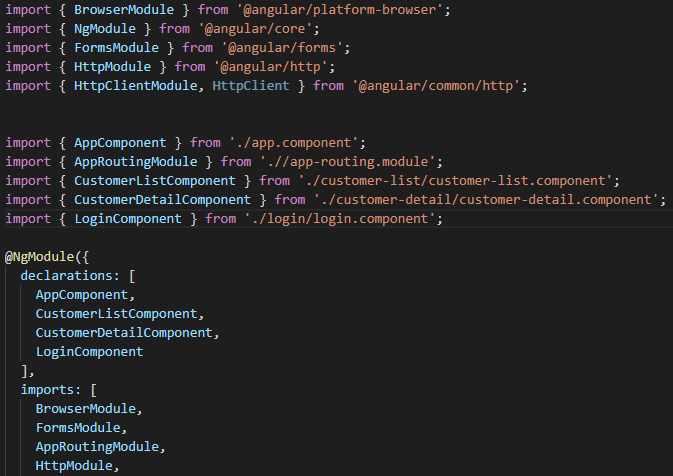
Html DOM functions are valid like document.getElementById importing FormsModule

Open app.component.html and delete content and add Navigation contect.



App.module.cs:

Import Required Modules. Here FormsModule(forms controls and F, HttpModule(http Requests), HttpClient (api calls and service implementation).

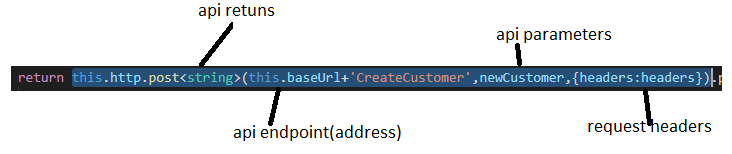


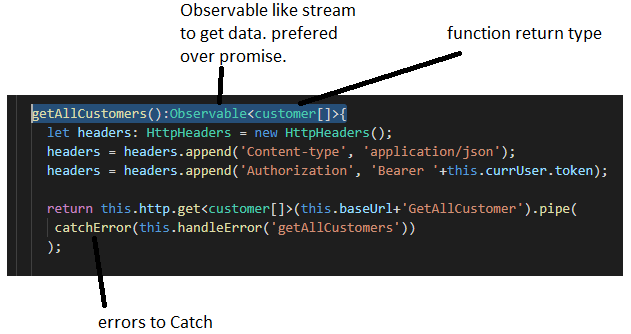
**Classes:**

userLogin, customer are two classes with required properties.

**Services:**

These are responsible for making api calls using http protocol. This Import HttpClient, HttpHeaders, HttpParams, RequestContentType, HttpResponse modules, which provides required classes. This coding same as XMLHttpRequest.,Rxjs.





**Components:**

**Login**

Formgroup, controls validation and error display are same as html.

OnForm Submit, get username, password from form controls and call authenticateUser api, if credentials are valid returns currentUser with token, which has stored in localStorage (html 5 features can be accessed across components. Saves as key value pair).

**CustomerList**:

This has html table to display Customer List. \*ngFor Structure directive will loop through customerlist and to create <tr>. (click) to select tr / currCustomer.

ngOnInt event fires while initializing page, in this event makes a call to getAllcustomer api to get Customers.

**CustomerDetail Component:**

This is simple FormsControl page to create Customer.

**Common Errors:**

Access to XMLHttpRequest at 'http://localhost:5327/api/Customer//GetAllCustomer' from origin 'http://localhost:4200' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Cause: CORS is not enabled.

Solution: CORS has to be enabled in startup.cs file. Can be done other ways also.

401 Error: Unauthorized:

Cause: Authorize Bearer Token is the issue. Syntax not correct, Token not valid or Token Validation in API is not correct.

Solution: debug and find api address, and Toekn syntax. for Dotnet core Version 2.0 add app.UseAuthentication() in startup.cs/Configure function