Create connection string

<add name="DBshopcart" connectionString="Data Source=localhost\SQLEXPRESS;Initial Catalog=DBshopcart;Integrated Security=True" providerName="System.Data.SqlClient" />

And create dbcontext with same name of connection string like DBshopcart

public class DBshopcart : DbContext

{

protected override void OnModelCreating(DbModelBuilder m)

{

Database.SetInitializer<DBshopcart>(null);

base.OnModelCreating(m);

}

public DbSet<CS\_user> CS\_users { get; set; }

static void Main (string[] args)

{

Database.SetInitializer(new MigrateDatabaseToLatestVersion<DBshopcart, Configuration>());

}

}

}

Create table model like this and install entity framework and sqlclient

public class CS\_user

{

[Key]

public int C\_U\_id { get; set; }

public string C\_U\_name { get; set; }

public string C\_U\_email { get; set; }

public string C\_U\_password { get; set; }

public int Num\_id { get; set; }

}

Enable the migration for database update and tables update

Enable-Migrations

Add-Migration anyname

Update-database

Install cors for access thi api from any where

Install-Package Microsoft.AspNet.WebApi.Cors -Version 5.2.7

Add a code on

App\_start open the cs file name with webapiconfig.cs

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Web API configuration and services

// Web API routes

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

config.EnableCors(new EnableCorsAttribute("\*", "\*", "\*")); //this is cors config

}

}

Its done with backend

Create react app

And install cors

 "cors-anywhere": "^0.4.4",

Add this line on package manager on top or below the private

"proxy": "https://localhost:44352",

Home Controller

using Newtonsoft.Json;

using Shopcart.Models;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Runtime.Remoting.Contexts;

using System.Web;

using System.Web.Mvc;

using System.Web.Script.Serialization;

using Newtonsoft;

namespace Shopcart.Controllers

{

public class HomeController : Controller

{

public string con = ConfigurationManager.ConnectionStrings["DBshopcart"].ConnectionString;

or

Contex t name

ApiContext db = new ApiContext();

public dynamic add(CSNumber num)

{

string msg = null;

try

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlCommand cmd = new SqlCommand("insert into CSNumbers values('"+num.Number+"','"+num.Alt\_number+"')", conn);

cmd.ExecuteNonQuery();

msg = "success";

}

catch (Exception)

{

msg = "not";

};

return msg;

}

public dynamic getdata()

{

SqlDataAdapter da = new SqlDataAdapter("select \* from CSNumbers where Num\_id = 3", con);

DataTable dt = new DataTable();

da.Fill(dt);

if(dt.Rows.Count > 0)

{

return JsonConvert.SerializeObject(dt);

}

else

{

return "no data found";

}

}

}

}

Crud Operation with ADO.net

using FSstore.Models;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace FSstore.Controllers

{

public class HomeController : Controller

{

// GET: Home

public string con = ConfigurationManager.ConnectionStrings["FSDB"].ConnectionString;

public dynamic adduser(User\_number u)

{

string msg = null;

try

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlCommand cmd = new SqlCommand("insert into User\_number values('" + u.Number + "','" + u.Alt\_number + "')", conn);

cmd.ExecuteNonQuery();

conn.Close();

msg = "success";

}

catch (Exception)

{

msg = "unsuccess";

}

return msg;

}

public dynamic getdata()

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlDataAdapter da = new SqlDataAdapter("select \* from User\_number ", conn);

DataTable dt = new DataTable();

da.Fill(dt);

if(dt.Rows.Count > 0)

{

return JsonConvert.SerializeObject(dt);

}

else

{

return "no data found";

}

}

public dynamic findbynumber(User\_number u)

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlDataAdapter da = new SqlDataAdapter("select \* from User\_number where Number = "+u.Number+"", conn);

DataTable dt = new DataTable();

da.Fill(dt);

if (dt.Rows.Count > 0)

{

return JsonConvert.SerializeObject(dt);

}

else

{

return "no data found";

}

}

public dynamic updatedata(User\_number u)

{

string msg;

try

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlCommand cmd = new SqlCommand("update User\_number set Alt\_number = "+u.Alt\_number+" where Num\_id = "+u.Num\_id+"", conn);

cmd.ExecuteNonQuery();

conn.Close();

msg = "success";

}

catch (Exception)

{

msg = "unsuccess";

}

return msg;

}

public dynamic deletedata(User\_number u)

{

string msg;

try

{

SqlConnection conn = new SqlConnection(con);

conn.Open();

SqlCommand cmd = new SqlCommand("delete from User\_number where Num\_id="+u.Num\_id+"", conn);

cmd.ExecuteNonQuery();

conn.Close();

msg = "success";

}

catch (Exception)

{

msg = "unsuccess";

}

return msg;

}

}

}

Entityframe work crud

using FSstore.Models;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace FSstore.Controllers

{

public class HomeController : Controller

{

FSDBContext db = new FSDBContext();

public dynamic insertdata(User\_number u)

{

string msg;

try

{

db.user\_Numbers.Add(u);

db.SaveChanges();

msg = "success";

}

catch (Exception)

{

msg = "unsuccess";

}

return msg;

}

public dynamic getdata()

{

var data = db.user\_Numbers.ToList();

var json = JsonConvert.SerializeObject(data);

return json;

}

[HttpPut]

public dynamic Putdata(User\_number u)

{

var \_db = db.user\_Numbers.Find(u.Num\_id);

\_db.Number = u.Number;

db.Entry(\_db).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return "successs";

}

public dynamic findby(User\_number u)

{

User\_number user = db.user\_Numbers.Find(u.Num\_id);

var json = JsonConvert.SerializeObject(user);

return json;

}

public dynamic deletedata(User\_number u)

{

User\_number user = db.user\_Numbers.Find(u.Num\_id);

db.user\_Numbers.Remove(user);

db.SaveChanges();

return "deleted";

}

}

}

Joins in entity framework

Create a class and add the props name which thing you want to in json type

public class Customernumbers

{

public string Name { get; set; }

public string Email { get; set; }

public string Add { get; set; }

public long Number { get; set; }

}

public dynamic cusnum()

{

var result = (from a in db.user\_Numbers

join b in db.customers on a.Num\_id equals b.Num\_id

join c in db.addresses on b.Cus\_id equals c.ADD\_id

select new Customernumbers

{

Name = b.Name,

Email = b.Email,

Number = a.Number,

Add = c.Add

}).ToList();

var data = JsonConvert.SerializeObject(result);

return data;

}

Asp.net core startup

Install these three

Entity frameworkcore

Entity frameworkCore.sqlserver

Entity frameworkCore.Tools this iis yse for migration

public class Appcontext : DbContext

{

public Appcontext(DbContextOptions<Appcontext> options) : base(options)

{

}

public DbSet<Users> users { get; set; }

public DbSet<customer> customers { get; set; }

}

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

string cs = @"Data Source=localhost\SQLEXPRESS;Initial Catalog=beast;Integrated Security=True";

services.AddDbContext<Appcontext>(options => options.UseSqlServer(cs));

}

Asp.net core web api

Install these three

Entityframework core

Entityframework core.sqlserver

Entityframework core.tool

Swasbuckle .aspnetcore.Swagger gen

Swasbuckle .aspnetcore.Swagger ui

Appsettings.json

"ConnectionStrings": {

"dataconnection": "Data Source=localhost\\SQLEXPRESS;Initial Catalog=Mangal;Integrated Security=True"

},

AppContext.cs

public class AppContext: DbContext

{

public AppContext(DbContextOptions<AppContext> options) :base(options)

{

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.EnableSensitiveDataLogging();

}

public DbSet<Employees> employees { get; set; }

Startup.cs

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddDbContext<Model.AppContext>(options =>

{

options.UseSqlServer(Configuration.GetConnectionString("dataconnection"));

});

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v2", new OpenApiInfo { Title = "myapi", Version = "v2" });

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v2/swagger.json", "my api v1");

});

}

}

public dynamic Adduser(User\_number u)

{

User\_number us = new User\_number();

int count = (int)u.U\_number;

int numcount = count.ToString().Count();

string msg;

try

{

if (numcount == 10)

{

us.U\_number = u.U\_number;

us.U\_altnumber = u.U\_altnumber;

us.type = u.type;

db.user\_number.Add(us);

db.SaveChanges();

msg = "success";

}

else

{

msg = "unsuccess";

}

}

catch (Exception)

{

msg = "user already exist";

}

return msg;

}

public dynamic update\_number(User\_number u)

{

var \_db = db.user\_number.Find(u.num\_id);

if(u.U\_number != \_db.U\_number)

{

\_db.U\_number = u.U\_number;

}

else if(u.U\_altnumber != \_db.U\_altnumber)

{

\_db.U\_altnumber = u.U\_altnumber;

}

else

{

return "This number already match with previous";

}

db.Entry(\_db).State = System.Data.EntityState.Modified;

db.SaveChanges();

return "success";

}

AddCors in asp.net core

Default Implementation

Startup.cs

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v2", new OpenApiInfo { Title = "Api", Version = "v2" });

});

services.AddCors(c =>

{

c.AddDefaultPolicy(builder => builder.AllowAnyOrigin());

or

c.AddDefaultPolicy(builder => builder.WithOrigins());

or

c.AddPolicy("mypoliciy", builder =>

builder.WithOrigins("urlname"));

});

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseCors();

OR

app.UseCors(“mypolicy”); = you can add policy name or its also on default

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

OR

endpoints.MapControllers().RequireCors("mypolicy");

});

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v2/swagger.json", "A2pi v");

});

}

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

string cs = @"Data Source=localhost\SQLEXPRESS;Initial Catalog=managaldb;Integrated Security=True";

services.AddDbContext<AppDBContext>(options => options.UseSqlServer(cs));

services.AddCors();

}

app.UseCors(c => c.WithOrigins("http://localhost:3000").AllowAnyHeader().AllowAnyMethod());

Succefully work

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v2", new OpenApiInfo { Title = "Api", Version = "v2" });

});

services.AddCors(c =>

{

c.AddPolicy("policy", builder =>

builder.WithOrigins("http://localhost:3000"));

});

string cs = @"Data Source=localhost\SQLEXPRESS;Initial Catalog=dbtest;Integrated Security=True";

services.AddDbContext<AppDBContext>(options =>

{

options.UseSqlServer(cs);

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseCors("policy");

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v2/swagger.json", "A2pi v");

});

}

Fromsqlraw

var data = db.employees.FromSqlRaw("select \* from employees where id = 2").ToList();