**LAB 7**

**OBJECTIVE:** To setup the connection for API and test the api from postman.

**THEORY:**

REST is acronym for REpresentational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous [dissertation](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm).

**GROUP DISCUSSION / PROCEDURE:**

**Navigate to “appsettings.json” file and add “ConnectionStrings” section:**

{

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=ApexRestaurantDb;Trusted\_Connection=True;"

},

"Logging": {

"IncludeScopes": false,

"LogLevel": {

"Default": "Warning"

}

}

}

**Navigate to “Startup.cs” and add entries for RepositoryModule and** ServiceModule in the

method ConfigureServices as below.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using ApexRestaurant.Repository;

using ApexRestaurant.Services;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

using Microsoft.Extensions.Options;

namespace ApexRestaurant.Api

{

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)

{

RepositoryModule.Register(services,

Configuration.GetConnectionString("DefaultConnection"),

GetType().Assembly.FullName);

ServicesModule.Register(services);

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

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

HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseMvc();

}

}

}

**Under “Controllers” folder, add “CustomerController.cs”**

using ApexRestaurant.Repository.Domain;

using ApexRestaurant.Services.SCustomer;

using Microsoft.AspNetCore.Mvc;

namespace ApexRestaurant.Api.Controller

{

[Route("api/customer")]

public class CustomerController : ControllerBase

{

private readonly ICustomerService \_customerService;

public CustomerController(ICustomerService customerService)

{

\_customerService = customerService;

}

[HttpGet]

[Route("{id}")]

public IActionResult Get([FromRoute] int id)

{

var customer = \_customerService.GetById(id);

if (customer == null)

return NotFound();

return Ok(customer);

}

[HttpGet]

[Route("")]

public IActionResult GetAll()

{

var customers = \_customerService.GetAll();

return Ok(customers);

}

[HttpPost]

[Route("")]

public IActionResult Post([FromBody] Customer model)

{

\_customerService.Insert(model);

return Ok();

}

[HttpPut]

[Route("")]

public IActionResult Put([FromBody] Customer model)

{

\_customerService.Update(model);

return Ok();

}

[HttpDelete]

[Route("")]

public IActionResult Delete([FromBody] Customer model)

{

\_customerService.Delete(model);

return Ok();

}

}

}

**Add necessary dependencies to the project.**

cd ApexRestaurant.Api ¶

dotnet add package Microsoft.EntityFrameworkCore ¶

dotnet add package Microsoft.EntityFrameworkCore.Abstractions¶

dotnet add package Microsoft.EntityFrameworkCore.Analyzers¶

dotnet add package Microsoft.EntityFrameworkCore.Relational¶

dotnet add package Microsoft.EntityFrameworkCore.SqlServer¶

dotnet add package Microsoft.Extensions.Caching.Abstractions¶

dotnet add package Microsoft.Extensions.Caching.Memory ¶

dotnet add package Microsoft.Extensions.Configuration ¶

dotnet add package Microsoft.Extensions.Configuration.Abstractions ¶

dotnet add package Microsoft.Extensions.Configuration.Binder ¶

dotnet add package Microsoft.Extensions.DependencyInjection ¶

dotnet add package Microsoft.Extensions.DependencyInjection.Abstractions ¶

dotnet add package Microsoft.Extensions.Logging ¶

dotnet add package Microsoft.Extensions.Logging.Abstractions ¶

dotnet add package Microsoft.Extensions.Options ¶