From Nuget Package, Install ***Dapper*** & ***System.Data.SqlClient***

The IConfiguration interface is used to read Settings and Connection Strings from AppSettings.json file.

**Step 1:** Create a Project, After creating click on view > sql server, connect to a database, right click on databases folder and create new database “Dapper”.

Right click on newly created ‘Dapper’ Database and click on new query:

Paste the below script to create a database table:

USE [Dapper]

GO

ALTER TABLE [dbo].[Employees] DROP CONSTRAINT [FK\_Employees\_Companies]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Employees] Script Date: 5/10/2021 10:39:05 AM \*\*\*\*\*\*/

DROP TABLE [dbo].[Employees]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Companies] Script Date: 5/10/2021 10:39:05 AM \*\*\*\*\*\*/

DROP TABLE [dbo].[Companies]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Companies] Script Date: 5/10/2021 10:39:05 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Companies](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Address] [nvarchar](60) NOT NULL,

[Country] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Companies] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Employees] Script Date: 5/10/2021 10:39:05 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Employees](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Age] [int] NOT NULL,

[Position] [nvarchar](50) NOT NULL,

[CompanyId] [int] NOT NULL,

CONSTRAINT [PK\_Employees] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET IDENTITY\_INSERT [dbo].[Companies] ON

INSERT [dbo].[Companies] ([Id], [Name], [Address], [Country]) VALUES (1, N'IT\_Solutions Ltd', N'583 Wall Dr. Gwynn Oak, MD 21207', N'USA')

INSERT [dbo].[Companies] ([Id], [Name], [Address], [Country]) VALUES (2, N'Admin\_Solutions Ltd', N'312 Forest Avenue, BF 923', N'USA')

SET IDENTITY\_INSERT [dbo].[Companies] OFF

SET IDENTITY\_INSERT [dbo].[Employees] ON

INSERT [dbo].[Employees] ([Id], [Name], [Age], [Position], [CompanyId]) VALUES (1, N'Sam Raiden', 26, N'Software developer', 1)

INSERT [dbo].[Employees] ([Id], [Name], [Age], [Position], [CompanyId]) VALUES (2, N'Kane Miller', 35, N'Administrator', 2)

INSERT [dbo].[Employees] ([Id], [Name], [Age], [Position], [CompanyId]) VALUES (3, N'Jana McLeaf', 30, N'Software developer', 1)

SET IDENTITY\_INSERT [dbo].[Employees] OFF

ALTER TABLE [dbo].[Employees] WITH CHECK ADD CONSTRAINT [FK\_Employees\_Companies] FOREIGN KEY([CompanyId])

REFERENCES [dbo].[Companies] ([Id])

GO

ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [FK\_Employees\_Companies]

GO

**Step 2:** Create a repository (Entity) to communicate with the database.

Create a new folder under solution named “Entities”

Create a new class named “Employee”

namespace DapperSample.Entities

{

public class Employee

{

public int Id { get; set; }

public string? Name { get; set; }

public int Age { get; set; }

public string? Position { get; set; }

public int CompanyId { get; set; }

}

}

Create a new class named “Company”

namespace DapperSample.Entities

{

public class Company

{

public int Id { get; set; }

public string? Name { get; set; }

public string? Address { get; set; }

public string? Country { get; set; }

public List<Employee> Employees { get; set; } = new List<Employee>();

}

}

**Step 3:** Add the connection string in appsettings.json file

"ConnectionStrings": {

"SqlConnection": "server=ILPT4055; database=Dapper; Integrated Security=true Encrypt=false"

},

**Step 4:** Now Create a new folder “Context” inside it create a new class as “DapperContext”

using Microsoft.Data.SqlClient;

using System.Data;

namespace DapperSample.Context

{

public class DapperContext

{

private readonly IConfiguration \_configuration;

private readonly string \_connectionString;

public DapperContext(IConfiguration configuration, string connectionString)

{

\_configuration = configuration; //to access our configuration file, using which we can read the properties

\_connectionString = \_configuration.GetConnectionString("Sqlconnection"); //to get the Connection string properties

}

//this method will return a new SQL Connection

public IDbConnection CreateConnection() => new SqlConnection(\_connectionString);

}

}

**Step 5: *Now we have to register this class as a service***

Program.cs

using DapperSample.Context;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddSingleton<DapperContext>();

builder.Services.AddControllers();

var app = builder.Build();

**Step 6:** Create two new folders:

Create a new folder “Contracts” under which create a new class “ICompanyRepository”

*ICompanyRepository.cs:*

using DapperSample.Entities;

namespace DapperSample.Contracts

{

public interface ICompanyRepository

{

public Task<IEnumerable<Company>> GetCompanies();

}

}

Create a new folder “Repository” under which create a new class “CompanyRepository”

*CompanyRepository.cs:*

using Dapper;

using DapperSample.Context;

using DapperSample.Contracts;

using DapperSample.Entities;

namespace DapperSample.Repository

{

public class CompanyRepository : ICompanyRepository

{

private readonly DapperContext \_context;

public CompanyRepository(DapperContext context) => \_context = context;

public async Task<IEnumerable<Company>> GetCompanies()

{

var query = "SELECT \* FROM Companies";

using(var connection = \_context.CreateConnection())

{

var companies = await connection.QueryAsync<Company>(query);

return companies.ToList();

}

}

}

}

**Step 7: *Now we have to register this class as a service in Program.cs***

using DapperSample.Context;

using DapperSample.Contracts;

using DapperSample.Repository;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddSingleton<DapperContext>();

builder.Services.AddScoped<ICompanyRepository, CompanyRepository>();

**Step 8:** Create a controller “CompaniesController” fro Get Implementation

using DapperSample.Contracts;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace DapperSample.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class CompaniesController : ControllerBase

{

private readonly ICompanyRepository \_companyRepo;

public CompaniesController(ICompanyRepository companyRepo) => \_companyRepo = companyRepo;

[HttpGet]

public async Task<IActionResult> GetCompanies()

{

var companies = await \_companyRepo.GetCompanies();

return Ok(companies);

}

}

}

//IF in case the property changes in model i.e company.cs and mismatches with the property in database table then we have to change the query according to the database fields:

var query = "SELECT Id, Name AS CompanyName, Address, Country FROM Companies";

**Step 9:** Create Dto Folder and create a new class inside the Dto folder as *CompanyForCreationDto*

namespace DapperSample.Dto

{

public class CompanyForCreationDto

{

public string? Name { get; set; }

public string? Address { get; set; }

public string? Country { get; set; }

}

}