Chapter 3: Web Application

Lesson2: Basic CRUD using Asp.net Core and Entity Framework

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.

Step 1 : Contacts API Overview

The Contacts API is very simple, basic Web API which does CRUD operations. I have focused on writing web API rather than integrating it with databases. This table summaries Contacts API which we'll create

Step 2: Create ASP.NET Core Web API project

Install ASP.NET Core RC2 tools is must

Open Visual Studio 2015 Update 2, create "New Project" with name "**ContactsApi**" targeting .NET Framework 4.6. We can target .NET 4.5.1 or .NET 4.5.2

Step 3: Packages included for ASP.NET Core MVC 6 Web API

The packages included are "MVC", "EnvironmentalVariables", "JSON", "Logging".

Step 4: Creating Contacts model

Contacts class is centre of this MVC 6 Web API. Its POCO class containing some properties which are self explanatory.

Right click "**ContactsApi**" solution, create folder "**Models**"; under this "**Models**" folder create C# class "**Contacts.cs**" and copy this code

Step 5: Create and Register repository class for *Contacts*

The use of repository classes is really optional, but I have added it so that we can connect to any databases later.

Create "**Repository**" folder under "*ContactsApi*" solution, we will add one C# interface file and C# class file implementing this interface.

Step 6: Add Contacts API Controller

Its time to add the controller API which acts as MVC 6 Web API. Create "**Controllers**" folder under "*ContactsApi*" project solution and add C# class file "**ContactsController.cs**"; copy below code

Step 7: Enable CamelCasePropertyNamesContractResolver

Any Web Api (REST based) should return JSON response in form of Camel Case so that we can sure consume the API in any client. We need to enable CamelCasePropertyNamesContractResolver in Configure Services.

Step 8: Testing Contacts MVC 6 WEB API using POSTMAN

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using ContactsApi.Repository;
using Newtonsoft.Json.Serialization;

namespace ContactsApi
{
```

```
public class Startup
{
    public Startup(IHostingEnvironment env)
    {
        var builder = new ConfigurationBuilder()
            .SetBasePath(env.ContentRootPath)
            .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
            .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
            .AddEnvironmentVariables();
        Configuration = builder.Build();
    }

    public IConfigurationRoot Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        // Add framework services.
        services.AddMvc()
                .AddJsonOptions(a => a.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver()); ;

        //using Dependency Injection
        services.AddSingleton<IContactsRepository, ContactsRepository>();
    }
```

```
        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            app.UseMvc();
        }
    }
}
```