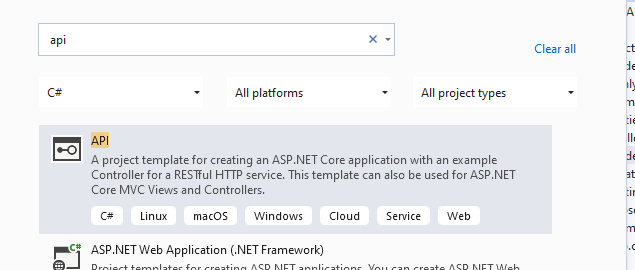
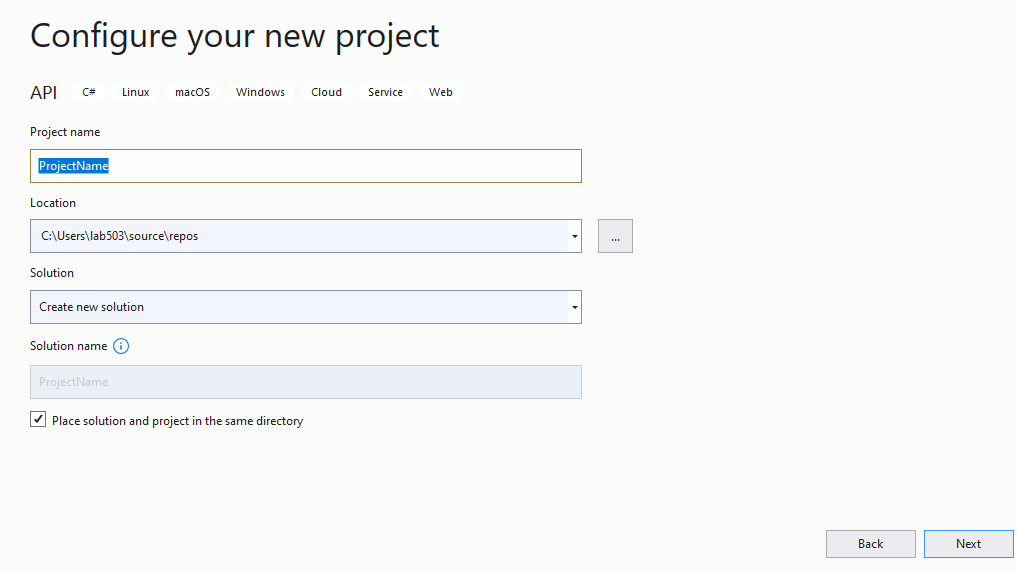
Building ASP.Net core REST API

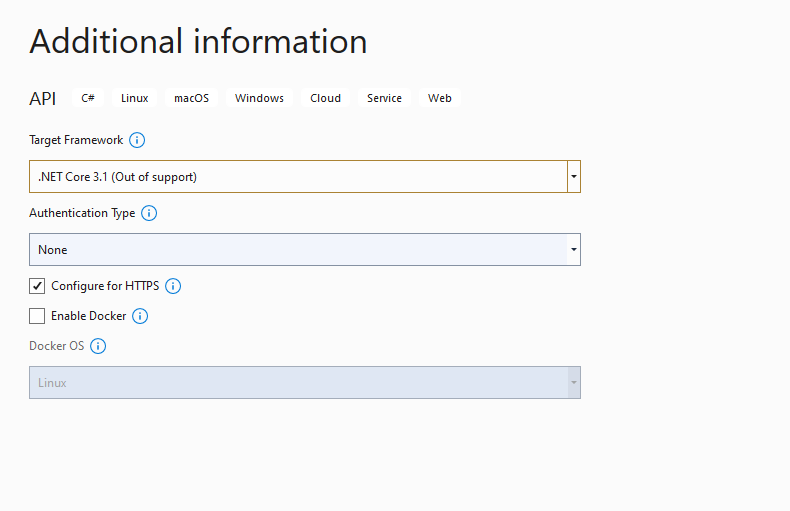
Step 1:- create new project in visual studio 2019

Step 2:- Select the Template API as mentioned below



Step 3:- Create the project with name and configures





Step 4:- You will Get all the pre-configured Code under the \*Program.cs\* and \*Startup.cs\*

**Below are the Code of Startup.cs**

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.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjectName

{

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();

}

// 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();

});

}

}

}

**Below are the Code of Program.cs**

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjectName

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

**Step 5: Create Student pojo class to store the student info**

**Below Student.cs pojo class**

namespace RestAPI.Controllers

{

public class Students

{

public string Name { get; set; }

public string Rollno { get; set; }

}

}

**Below Student Controller class file**

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.IO;

namespace RestAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class Student : ControllerBase

{

//this is local database

private static readonly List<Students> Studentlist = new List<Students>

{

new Students

{

Name = "Abdulsalam",

Rollno = "195"

},

new Students

{

Name = "Maria",

Rollno = "179"

},

new Students

{

Name = "imran",

Rollno = "201"

}

};

//You will get all student from here post method

[HttpGet]

public ActionResult<List<Students>> Get()

{

return Ok(Studentlist);

}

//You will get perticular student from here post method

[HttpGet]

[Route("{Name}")]

public ActionResult<List<Students>> Get(string Name)

{

var student = Studentlist.Find(student =>

student.Name.Equals(Name, StringComparison.InvariantCultureIgnoreCase));

if (student == null)

{

return NotFound();

}

else

{

return Ok(student);

}

}

//You will post perticular student from here post method

// post

[HttpPost]

public ActionResult Post(Students sentstudents)

{

var sendstudent = Studentlist.Find(item =>

item.Name.Equals(sentstudents.Name,

StringComparison.InvariantCultureIgnoreCase));

if (sendstudent != null)

{

return Conflict("Cannot create the student because it already exists.");

}

else

{

Studentlist.Add(sentstudents);

var resourceUrl = Path.Combine(Request.Path.ToString(),

Uri.EscapeUriString(sentstudents.Name));

return Created(resourceUrl, sentstudents);

}

}

//post

//You will add perticular student from here put method

//put

//

[HttpPut]

public ActionResult Put(Students sentstudents)

{

var sendstudent = Studentlist.Find(item =>

item.Name.Equals(sentstudents.Name,

StringComparison.InvariantCultureIgnoreCase));

if (sendstudent == null)

{

return BadRequest("Cannot update a nont existing term.");

}

else

{

sendstudent.Rollno = sentstudents.Rollno;

return Ok();

}

}

//

//delete

// //You will Delete perticular student from here Delete method

[HttpDelete]

[Route("{Name}")]

public ActionResult Delete(string Name)

{

var Studentnames = Studentlist.Find(item =>

item.Name.Equals(Name,

StringComparison.InvariantCultureIgnoreCase));

if (Studentnames == null)

{

return NotFound();

}

else

{

Studentlist.Remove(Studentnames);

return NoContent();

}

}

}

//

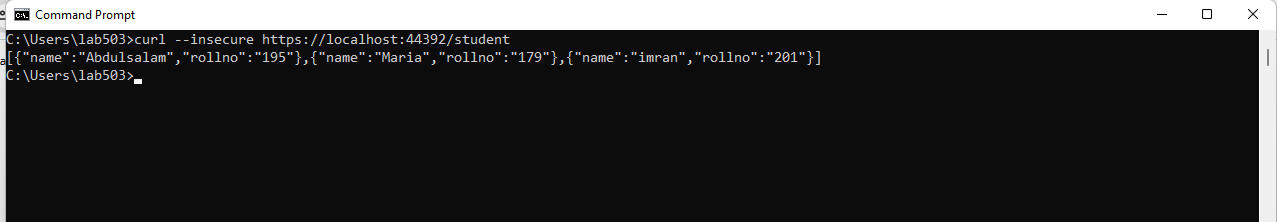
//

}

**All other configurations and file will remain same**

**Output**

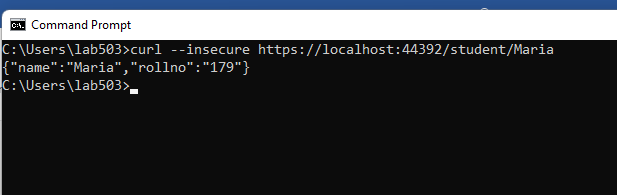
**Get method**

****

**Get method**

**Particular Student**

**In this case student name \*Maria\***

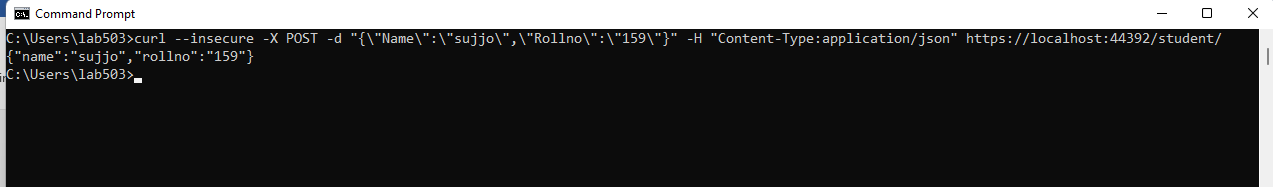
****

**Post method**

**Particular Student**

**In this case student name \*Sujjo\***

**curl --insecure -X POST -d "{\"Name\":\"sujjo\",\"Rollno\":\"159\"}" -H "Content-Type:application/json"** [**https://localhost:44392/student/**](https://localhost:44392/student/)

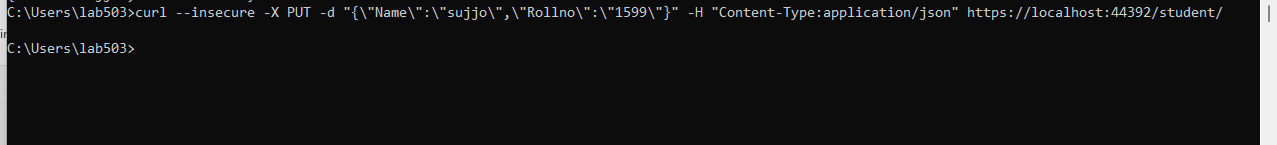
****

**Put method**

**Particular Student**

**In this case student name \*Sujjo\***

**curl --insecure -X PUT -d "{\"Name\":\"sujjo\",\"Rollno\":\"1599\"}" -H "Content-Type:application/json"** [**https://localhost:44392/student/**](https://localhost:44392/student/)

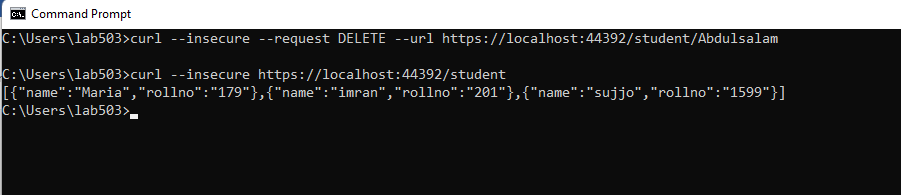
****

**Delete**

**Particular Student**

**In this case student name \*Abdulsalam\***

**curl --insecure --request DELETE --url** [**https://localhost:44392/student/Abdulsalam**](https://localhost:44392/student/Abdulsalam)

****