**Practical No: 5**

**Ninad Karlekar**

**Date: 12/04/2023**

**Aim: Building ASP.Net core REST API**

**Description:**

Give Overview of ASP.Net core REST API

Due to the increasing number of different varieties of clients (mobile apps, browser-based SPAs, desktop apps, IOT apps, etc.), we need better ways for transferring data from servers to clients, independent of technology and server stacks. REST APIs solve this problem. REST stands for representational state transfer. REST APIs are HTTP-based and provide applications the ability to communicate using lightweight JSON format. They run on web servers.

**Code & Output:**

**Create your web API**

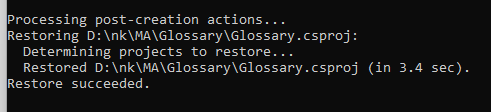
1. Open two command prompts

Command prompt 1:

Command:

dotnet new webapi -o Glossary

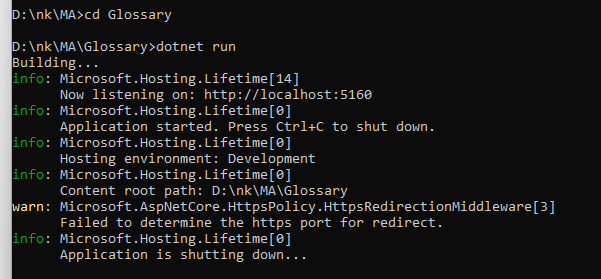
output:



**Command:**

**cd Glossary**

**dotnet run**

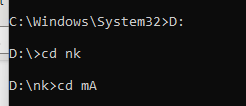


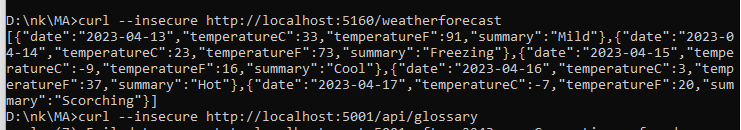
2. Command Prompt 2: (try running ready made weatherforecast class for testing)

Command:

curl --insecure http://localhost:5160/weatherforecast

output:





3. Now Change the content:

To get started, remove the WeatherForecast.cs file from the root of the project and the

WeatherForecastController.cs file from the Controllers folder.

Add Following two files

1) D:\Glossary\GlossaryItem.cs (type it in notepad and save as all files)

//GlossaryItem.cs

namespace Glossary

{

public class GlossaryItem

{

public string Term { get; set; }

public string Definition { get; set; }

}

}

2) D:\Glossary\Controllers\ GlossaryController.cs (type it in notepad and save as all files)

//Controllers/GlossaryController.cs

using System;

using System.Collections.Generic;

using Microsoft.AspNetCore.Mvc;

using System.IO;

namespace Glossary.Controllers

{

[ApiController]

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

public class GlossaryController: ControllerBase

{

private static List<GlossaryItem> Glossary = new List<GlossaryItem> {

new GlossaryItem

{

Term= "HTML",

Definition = "Hypertext Markup Language"

},

new GlossaryItem

{

Term= "MVC",

Definition = "Model View Controller"

},

new GlossaryItem

{

Term= "OpenID",

Definition = "An open standard for authentication"

}

};

[HttpGet]

public ActionResult<List<GlossaryItem>> Get()

{ return Ok(Glossary);

}

[HttpGet]

[Route("{term}")]

public ActionResult<GlossaryItem> Get(string term)

{

var glossaryItem = Glossary.Find(item =>

item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));

if (glossaryItem == null)

{ return NotFound();

} else

{

return Ok(glossaryItem);

}

}

[HttpPost]

public ActionResult Post(GlossaryItem glossaryItem)

{

var existingGlossaryItem = Glossary.Find(item =>

item.Term.Equals(glossaryItem.Term, StringComparison.InvariantCultureIgnoreCase));

if (existingGlossaryItem != null)

{

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

}

else

{

Glossary.Add(glossaryItem);

var resourceUrl = Path.Combine(Request.Path.ToString(), Uri.EscapeUriString(glossaryItem.Term));

return Created(resourceUrl, glossaryItem);

}

}

[HttpPut]

public ActionResult Put(GlossaryItem glossaryItem)

{

var existingGlossaryItem = Glossary.Find(item =>

item.Term.Equals(glossaryItem.Term, StringComparison.InvariantCultureIgnoreCase));

if (existingGlossaryItem == null)

{

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

} else

{

existingGlossaryItem.Definition = glossaryItem.Definition;

return Ok();

}

}

[HttpDelete]

[Route("{term}")]

public ActionResult Delete(string term)

{

var glossaryItem = Glossary.Find(item =>

item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));

if (glossaryItem == null)

{ return NotFound();

}

else

{ Glossary.Remove(glossaryItem);

return NoContent();

}

}

}

}

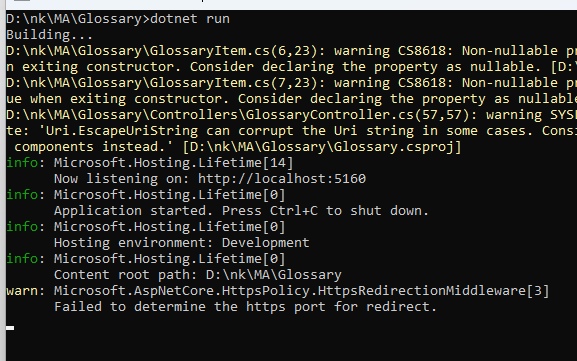
3.Now stop running previous dotnet run on command prompt 1 using Ctrl+C. and Run it again for new code.

On Command prompt1:

Command:

dotnet run

output:



On Command prompt2:

1) Getting a list of items:

Command:

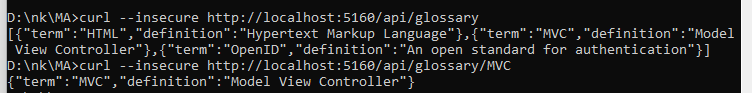
curl --insecure https://localhost:5160/api/glossary

2) Getting a single item

Command:

curl --insecure https://localhost:5001/api/glossary/MVC

Output:



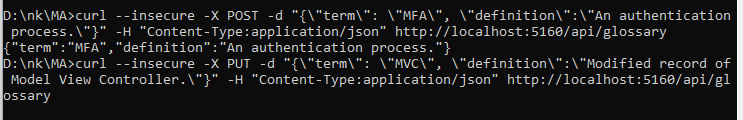
2) Creating an item

Command:

curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\":\"An authentication process.\"}" -H "Content-

Type:application/json" https://localhost:5160/api/glossary

Output:



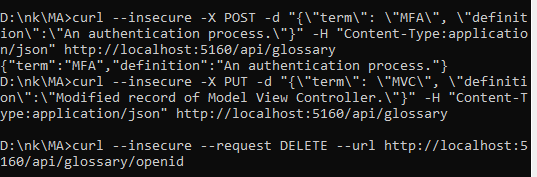
4)Update Item

Command:

curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model View

Controller.\"}" -H "Content-Type:application/json" https://localhost:5001/api/glossary

Output:



5) Delete Item

Command:

curl --insecure --request DELETE --url https://localhost:5001/api/glossary/openid

Output:

