

Microservices Architecture

Practical 2:

Building ASP.Net core REST API

Software requirement:

1. Download and install

To start building .NET apps you just need to download and install the .NET SDK (Software Development Kit version 3.0 above).

Link:

<https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install>

2. Check everything installed correctly

Once you've installed, open a new command prompt and run the following command:

Command prompt

> dotnet

Create your web API

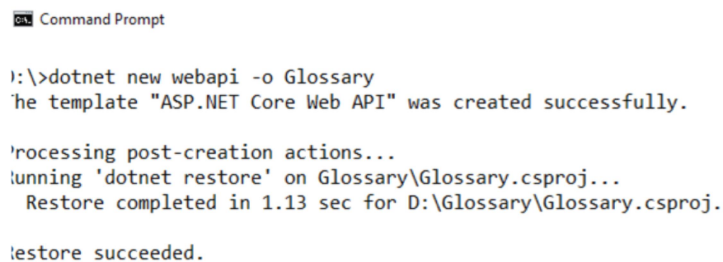
1. Open two command prompts

Command prompt 1:

Command:

dotnet new webapi -o Glossary

output:



```
Command Prompt

I:\>dotnet new webapi -o Glossary
The template "ASP.NET Core Web API" was created successfully.

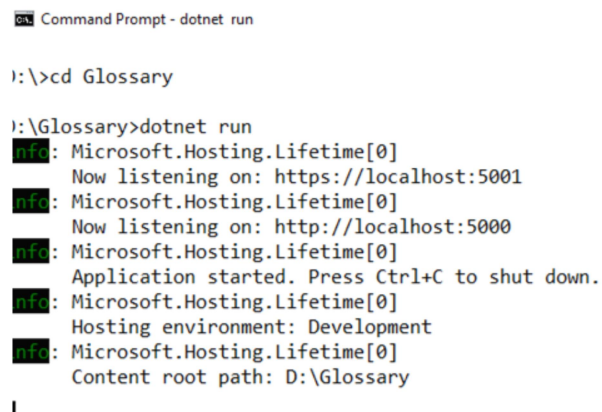
Processing post-creation actions...
Running 'dotnet restore' on Glossary\Glossary.csproj...
Restore completed in 1.13 sec for D:\Glossary\Glossary.csproj.
Restore succeeded.
```

Command:

cd Glossary

dotnet run

Output:



```
Command Prompt - dotnet run

I:\>cd Glossary

I:\Glossary>dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Glossary

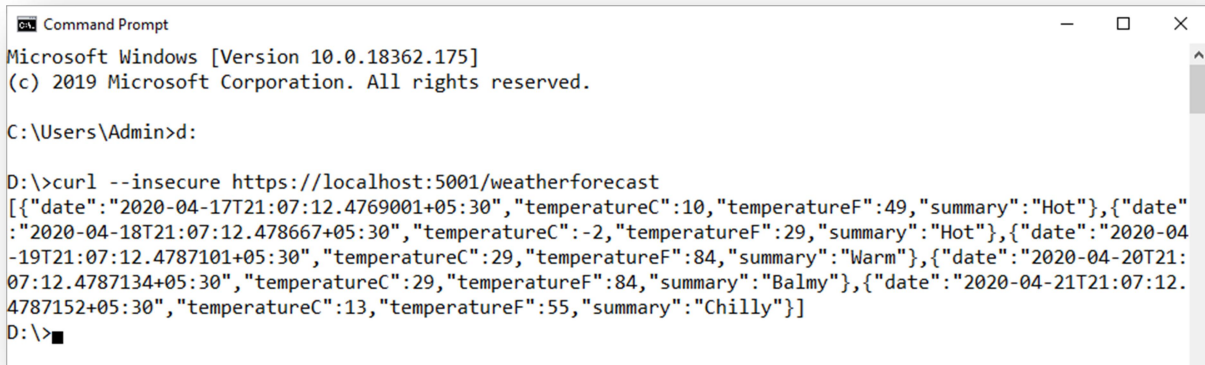
I
```

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

Command:

```
curl --insecure https://localhost:5001/weatherforecast
```

output:



```
Command Prompt
Microsoft Windows [Version 10.0.18362.175]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Admin>d:

D:\>curl --insecure https://localhost:5001/weatherforecast
[{"date":"2020-04-17T21:07:12.4769001+05:30","temperatureC":10,"temperatureF":49,"summary":"Hot"}, {"date":"2020-04-18T21:07:12.478667+05:30","temperatureC":-2,"temperatureF":29,"summary":"Hot"}, {"date":"2020-04-19T21:07:12.4787101+05:30","temperatureC":29,"temperatureF":84,"summary":"Warm"}, {"date":"2020-04-20T21:07:12.4787134+05:30","temperatureC":29,"temperatureF":84,"summary":"Balmy"}, {"date":"2020-04-21T21:07:12.4787152+05:30","temperatureC":13,"temperatureF":55,"summary":"Chilly"}]
D:\>
```

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]

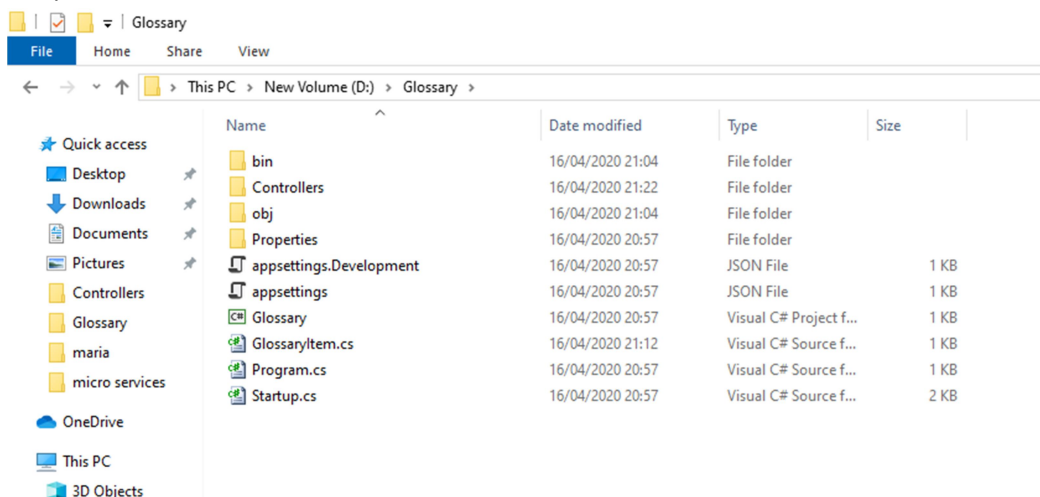
[Route("{term}")]

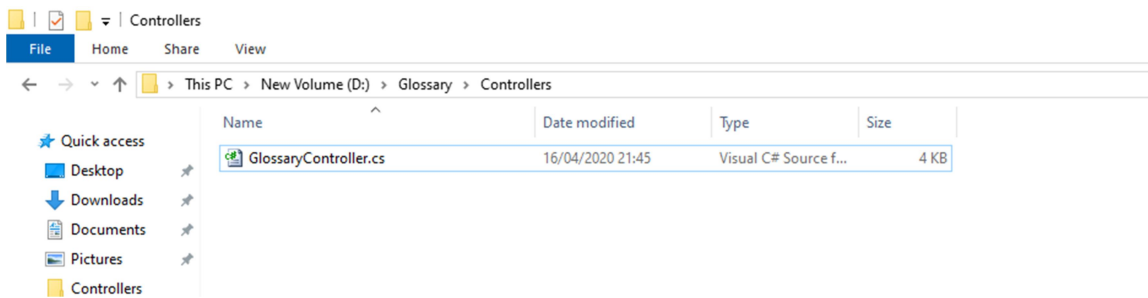
public ActionResult Delete(string term)

```
{
    var glossaryItem = Glossary.Find(item =>
        item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));

    if (glossaryItem == null)
    {
        return NotFound();
    }
    else
    {
        Glossary.Remove(glossaryItem);
        return NoContent();
    }
}
}
```

Output:





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:

```
Command Prompt - dotnet run

D:\Glossary>dotnet run
C:\Program Files\dotnet\sdk\3.1.201\Microsoft.Common.CurrentVersion.targets(4643,5): warning MSB3026: Could not copy "D:\Glossary\obj\Debug\netcoreapp3.1\Glossary.exe" to "bin\Debug\netcoreapp3.1\Glossary.exe". Beginning retry 1 in 1000ms. The process cannot access the file 'D:\Glossary\obj\Debug\netcoreapp3.1\Glossary.exe' because it is being used by another process. [D:\Glossary\Glossary.csproj]
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Glossary
```

On Command prompt2:

1) Getting a list of items:

Command:

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

Output:

```
Command Prompt

D:\>curl --insecure https://localhost:5001/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"}, {"term":"MVC","definition":"Model View Controller"}, {"term":"OpenID","definition":"An open standard for authentication"}]
D:\>
```

2) Getting a single item

Command:

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

Output:

```
Command Prompt

D:\>curl --insecure https://localhost:5001/api/glossary/MVC
{"term":"MVC","definition":"Model View Controller"}
D:\>
```

2) Creating an item

Command:

```
curl --insecure -X POST -d '{"term": "MFA", "definition": "An authentication process."}' -H "Content-Type:application/json" https://localhost:5001/api/glossary
```

Output:



```
Command Prompt
D:\>curl --insecure -X POST -d '{"term": "MFA", "definition": "An authentication process."}' -H "Content-Type:application/json" https://localhost:5001/api/glossary
{"term": "MFA", "definition": "An authentication process."}
D:\>curl --insecure https://localhost:5001/api/glossary
[{"term": "HTML", "definition": "Hypertext Markup Language"}, {"term": "MVC", "definition": "Model View Controller"}, {"term": "OpenID", "definition": "An open standard for authentication"}, {"term": "MFA", "definition": "An authentication process."}]
D:\>
```

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:



```
Command Prompt
D:\>curl --insecure -X PUT -d '{"term": "MVC", "definition": "Modified record of Model View Controller."}' -H "Content-Type:application/json" https://localhost:5001/api/glossary


D:\>curl --insecure https://localhost:5001/api/glossary
[{"term": "HTML", "definition": "Hypertext Markup Language"}, {"term": "MVC", "definition": "Modified record of Model View Controller."}, {"term": "OpenID", "definition": "An open standard for authentication"}, {"term": "MFA", "definition": "An authentication process."}]
D:\>
```

5) Delete Item

Command:

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

Output:



```
Command Prompt
D:\>curl --insecure --request DELETE --url https://localhost:5001/api/glossary/openid

D:\>curl --insecure https://localhost:5001/api/glossary
[{"term": "HTML", "definition": "Hypertext Markup Language"}, {"term": "MVC", "definition": "Modified record of Model View Controller."}, {"term": "MFA", "definition": "An authentication process."}]
D:\>
```