

Asp.Net With Web API

UNIT-6

Introduction To REST

REST (Representational State Transfer) is an architectural style for designing networked applications. It was introduced by Roy Fielding in his doctoral dissertation in 2000. RESTful systems typically use HTTP for communication and adhere to a set of principles that define how resources are identified and addressed.

In ASP.NET Web API, you can create RESTful services easily. Here's an introduction to REST principles in the context of ASP.NET Web API:

1. Resource Identification: In REST, everything is considered a resource. Resources are identified by URIs (Uniform Resource Identifiers). For example, `/api/products` could represent a collection of products, and `/api/products/{id}` could represent a specific product identified by its ID.

1. HTTP Verbs: HTTP verbs (GET, POST, PUT, DELETE, etc.) are used to perform actions on resources. Each HTTP verb corresponds to a CRUD (Create, Read, Update, Delete) operation on resources.
2. Stateless Communication: RESTful APIs are stateless, meaning each request from a client must contain all the information necessary to understand and process the request. The server should not store any client state between requests.
3. Representation: Resources can have different representations. Common representations include JSON, XML, HTML, and plain text. Clients can specify their preferred representation using HTTP headers (e.g., `Accept` header).
4. Uniform Interface: RESTful APIs should have a uniform interface to promote simplicity and scalability. This includes using standard HTTP methods, following resource naming conventions, and providing consistent response formats.

INTRODUCTION TO ASP.NET WEB API

ASP.NET Web API is a framework for building HTTP services that can reach a broad range of clients, including browsers and mobile devices. It's part of the ASP.NET platform and provides a simple, yet powerful, way to build APIs that follow RESTful principles.

1. **HTTP-Centric:** ASP.NET Web API is built around HTTP and provides full support for HTTP features, including content negotiation, request and response headers, caching, and more. This makes it ideal for building RESTful APIs.
2. **Routing:** Web API uses routing to map HTTP requests to controller actions. Routing in Web API is similar to ASP.NET MVC routing but optimized for HTTP services. You can define routes using attributes or by configuring routes in a routing table.
3. **Content Negotiation:** Web API supports content negotiation, allowing clients to request data in various formats such as JSON, XML, or even custom media types. This is achieved through the use of HTTP headers like `Accept` and `Content-Type`.

INTRODUCTION TO ASP.NET WEB API

4. Model Binding and Validation: Web API provides powerful model binding capabilities, allowing you to bind data from HTTP requests directly to action method parameters. It also includes built-in support for model validation, making it easy to validate incoming data.
5. ActionResult: In Web API, action methods return instances of `IHttpActionResult`, which represent the result of an HTTP operation. This allows you to return different types of responses, such as `Ok`, `NotFound`, `BadRequest`, etc., based on the outcome of the operation.
6. Filters and Middleware: Web API includes a rich set of filters and middleware that allow you to add cross-cutting concerns such as authentication, authorization, logging, exception handling, and caching to your API.
7. Dependency Injection: Web API supports dependency injection out of the box, allowing you to inject dependencies into your controllers, filters, and other components using constructor injection.
8. Testability: Web API is designed with testability in mind. You can easily write unit tests for your controllers and integration tests for your API using frameworks like NUnit, xUnit, or MSTest.

Code First Approach

- a) First of all, you have to create one model class with name “Product.cs”.
Add Id,Name and price as entities.
- b) Secondly, you can create one controller named as “Productc.cs”.
- c) Now again you have to create one class named as “ApplicationDbContext”, here you will creating dependencies.
- d) Now in appsetting.json, you will have to write connection string.
- e) One important thing, you will have to install nuget packages for ef core, ef tools, and sqlserver(that already we have discussed in Unit4)

WebApplication32

Debug Any CPU http Price

WebApplication32

WebApplication32.Product

namespace WebApplication32

{

 1 7 references

 2

 3 public class Product

 4

 5 3 references

 6 public int Id { get; set; }

 7 3 references

 8 public string? Name { get; set; }

 9

 10 }

Solution Explorer

Search Solution Explorer (Ctrl+)

WebApplication32

- Connected Services
- Dependencies
- Properties
- Controllers
 - C# Product.cs
 - C# WeatherForecastCont
- Migrations
- C# ApplicationDbContext.cs
- appsettings.json
- C# Product.cs
- C# Program.cs
- C# WeatherForecast.cs
- WebApplication32.http

Product.cs

Product.cs

Program.cs

WebAp...view

202404...ate.cs

[Tool Windows]

What's New?

192 % No issues found Ln: 10 Ch: 1 SPC CRLF

Ready Add to Source Control Select Repository

Type here to search

23°C Mostly cloudy ENG 23-04-2024

```
{②} 1  using Microsoft.EntityFrameworkCore;  
2  
3  namespace WebApplication32  
4  {  
5      5 references  
6      public class ApplicationDbContext : DbContext  
7      {  
8          //inject dependency  
9          0 references  
10         public DbSet<Product> Products { get; set; }  
11         0 references  
12         public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) {}  
13     }  
14 }
```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help | Search | WebApplication32 | Sign in

Tabs Schema: https://json.schemastore.org/appsettings.json

[Solution] [NuGet.sln] [WebApp...tion32] [Appli...ext.cs] appse...json* [Product.cs] [Product.cs] [Program.cs] [WebAp...view] [202404...ate.cs] [Tool Windows] What's New?

```
1  "Logging": {  
2      "LogLevel": {  
3          "Default": "Information",  
4          "Microsoft.AspNetCore": "Warning"  
5      }  
6  },  
7  "AllowedHosts": "*"  
8  "ConnectionStrings": {  
9      "DefaultConnection": "Server=DESKTOP-600KGAO\\SQLEXPRESS;Database=PalviSoniDB;"  
10  }  
11  }  
12  }  
13  }
```

174 % No issues found Ln: 13 Ch: 1 SPC CRLF

Type here to search Add to Source Control Select Repository

Ready 23°C Mostly cloudy ENG 23-04-2024

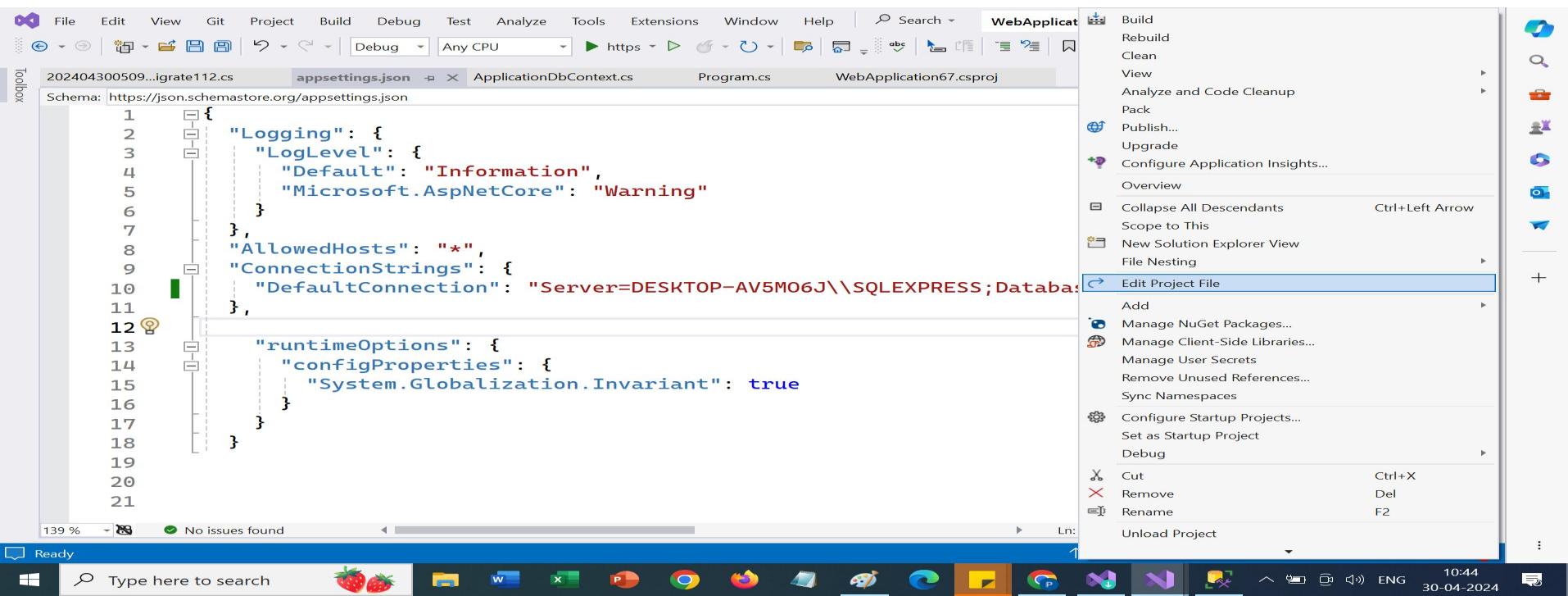
```
"ConnectionStrings": {
```

```
    "DefaultConnection": "Server=DESKTOP-600KGAO\\SQLEXPRESS;Database=PalviSoniDB;Integrated Security=true;TrustServerCertificate=true"
```

```
}
```

If your database is not updating and giving you globalization error

MAKE A RIGHT CLICK ON YOUR PROJECT AND CLICK EDIT PROJECT FILE



The screenshot shows a Windows desktop environment with Visual Studio 2022 open. The main window displays the code editor for an `appsettings.json` file. The code contains configuration for logging, allowed hosts, connection strings, and runtime options, specifically setting `System.Globalization.Invariant` to `true`. The status bar at the bottom indicates "No issues found". On the right side of the screen, the context menu for the project "WebApplication67.csproj" is open, showing various options like Build, Rebuild, Clean, View, and Edit Project File. The "Edit Project File" option is highlighted with a blue selection bar.

```
1  {
2      "Logging": {
3          "LogLevel": {
4              "Default": "Information",
5              "Microsoft.AspNetCore": "Warning"
6          }
7      },
8      "AllowedHosts": "*",
9      "ConnectionStrings": {
10         "DefaultConnection": "Server=DESKTOP-AV5M06J\\SQLEXPRESS;Database=..."
11     },
12     "runtimeOptions": {
13         "configProperties": {
14             "System.Globalization.Invariant": true
15         }
16     }
17 }
18
19
20
21
```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help | Search | WebApplication67.csproj | Build | Rebuild | Clean | View | Analyze and Code Cleanup | Pack | Publish... | Upgrade | Configure Application Insights... | Overview | Collapse All Descendants Ctrl+Left Arrow | Scope to This | New Solution Explorer View | File Nesting | Edit Project File | Add | Manage NuGet Packages... | Manage Client-Side Libraries... | Manage User Secrets | Remove Unused References... | Sync Namespaces | Configure Startup Projects... | Set as Startup Project | Debug | Cut Ctrl+X | Remove Del | Rename F2 | Unload Project

SetInvariantGlobalization false(initially it is true)-even if its not reflecting then also you can paste this in the project file

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <InvariantGlobalization>false</InvariantGlobalization>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Localization" Version="2.2.6" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.4" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.4" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.4" />
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; Assets that are private in this reference; buildtransitive</IncludeAssets>
  </ItemGroup>
</Project>
```

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar says "WebApplication67". The menu bar includes File, Edit, View, Git, Project, Build, Debug, XML, Test, Analyze, Tools, Extensions, Window, Help. The toolbar has icons for file operations like Open, Save, and Run. The ribbon tabs are File, Edit, View, Git, Project, Build, Debug, XML, Test, Analyze, Tools, Extensions, Window, Help. The search bar says "Search". The solution explorer on the right shows the project structure: Connected Services, Dependencies, Properties, Controllers, Migrations, 20240428041919.cs, 20240430050832.cs, 20240430050932.cs, ApplicationDbContext.cs, ApplicationDbConte..., appsettings.json, Product.cs, Program.cs, WeatherForecast.cs, and WebApplication67.htm. The code editor window displays the "WebApplication67.csproj" file with the following content. A red box highlights the `<PropertyGroup>` section, and a red arrow points from the explanatory text above to this section. A tooltip "Assets that are private in this reference" appears over the `<IncludeAssets>` tag.

The screenshot shows the Visual Studio IDE interface with two windows side-by-side. Both windows have the title bar "WebApplication32".

The left window displays the code for `Program.cs`. The code initializes a `WebApplicationBuilder`, adds controllers and a database context, and configures endpoints and Swagger. A red box highlights the database configuration code:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
builder.Services.AddDbContext<ApplicationContext>
(
    options=>
{
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"));
});

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseAuthorization();
```

The right window is the Solution Explorer, showing the project structure for "WebApplication32" with files like `Connected Services`, `Dependencies`, `Properties`, `Controllers` (containing `Product.cs` and `WeatherForecast.cs`), `Migrations` (containing `20240423174205.cs`), `ApplicationDbContext.cs`, `appsettings.json`, `Product.cs`, `Program.cs`, `WeatherForecast.cs`, and `WebApplication32.csproj`.

Any CPU http abc products

```
[Route("api/[controller]")]
[ApiController]
public class Productc : ControllerBase
{
    private static List<Product> products = new List<Product>
    {
        new Product { Id = 11, Name = "Product One", Price = 120000 },
        new Product { Id = 12, Name = "Product two", Price = 420000 },
        new Product { Id = 21, Name = "Product three", Price = 620000 }
    };
    [HttpGet]
    public List<Product> GetProducts()
    {
        return products;
    }
}
```

174% No issues found

Ln: 26 Ch: 1 SPC CRLF

```
PM> add-migration InitialMigrate
```

```
Build started...
```

```
Build succeeded.
```

```
To undo this action, use Remove-Migration.
```

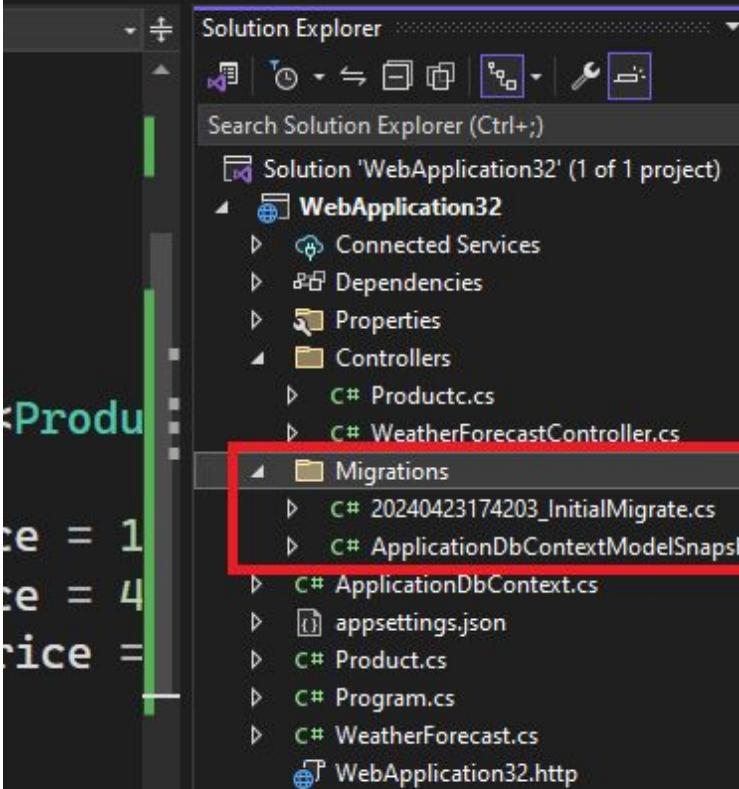
```
PM> update-database
```

```
Build started...
```

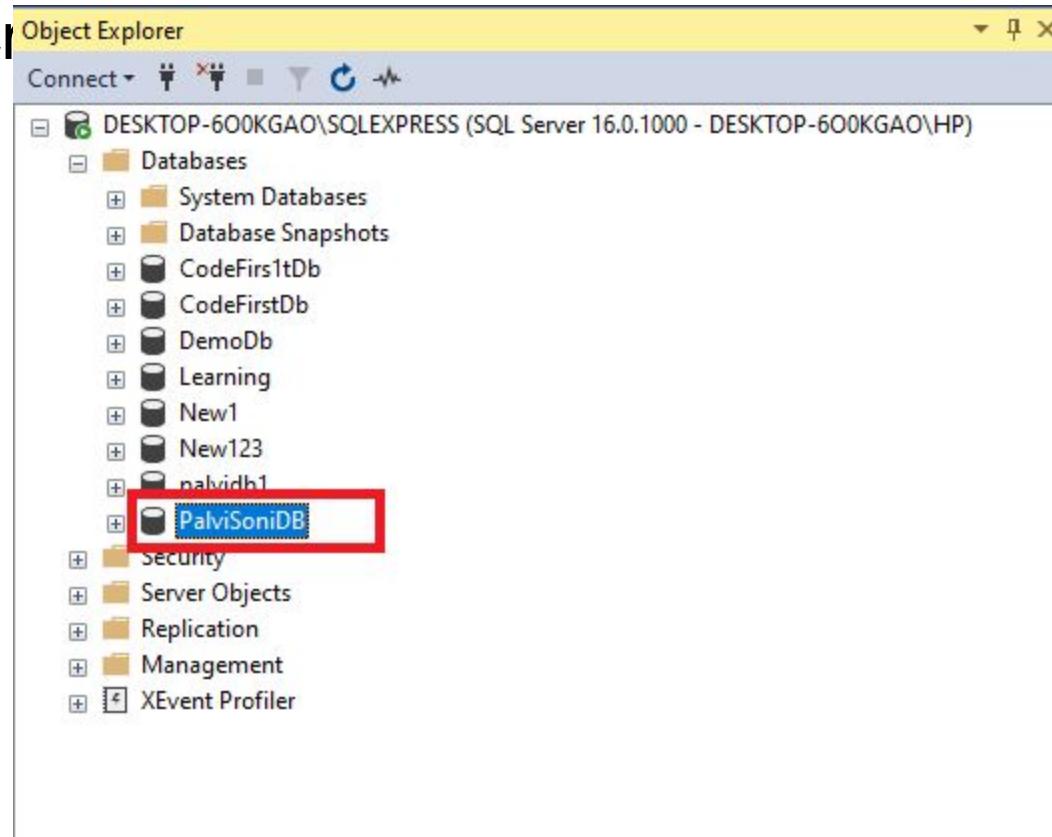
```
Build succeeded.
```

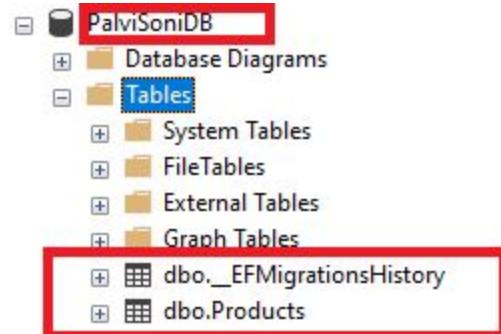
```
Microsoft.EntityFrameworkCore.Database.Command[20101]
```

```
Executed DbCommand (557ms) [Parameters=[], CommandType='Text', CommandText='CREATE DATABASE [PalviSoniDB];']
```



Database creation





CRUD operations with Web API

The screenshot shows the Microsoft Visual Studio interface with the following details:

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Solution Explorer:** Shows the solution structure for "WebApplication35" with items like "Connected Services", "Dependencies", "Properties", and "Controllers".
- Toolbox:** Standard Windows-style icons for file operations.
- Taskbar:** Icons for File Explorer, Task View, Start, Taskbar settings, and a search bar.
- Code Editor:** Displays the file "20240429184209_initialmigrate.cs" containing C# code for Entity Framework Core migrations. The code defines a partial class `initialmigrate` with `Up` and `Down` methods for creating a `products` table with columns `Id`, `Name`, and `Amount`.
- Contextual Menu:** A context menu is open over the code editor, listing options such as Controller..., Razor Component..., New Item..., Existing Item..., New Scaffolded Item..., New Folder, Container Orchestrator Support..., Docker Support..., Application Insights Telemetry..., Client-Side Library..., Machine Learning Model..., New Azure WebJob Project, Existing Project as Azure WebJob, Class..., and New EditorConfig.
- Status Bar:** Shows "This item does not support previewing", "Add to Source Control", "Select Repository", "20°C", "ENG", "00:13", and the date "30-04-2024".

Add New Scaffolded Item

Installed

- Common
 - API
 - MVC
- Controller
- View
- Razor Component
- Razor Pages
- Identity
- Layout

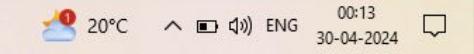
API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
API with read/write endpoints
API with read/write endpoints, using Entity Framework

API Controller - Empty
by Microsoft
v1.0.0.0
An empty API controller.
Id: ApiControllerEmptyScaffolder

Add Cancel

This item does not support previewing

Add to Source Control Select Repository



Any CPU http abc Up(MigrationBuilder migrationBuilder)

```
1  using Microsoft.EntityFrameworkCore.Migrations;
2
3  #nullable disable
4
5  namespace WebApplication35.Migrations
6  {
7      /// <inheritdoc />
8
9
10     public partial class ProductsC : Migration
11     {
12         protected override void Up(MigrationBuilder migrationBuilder)
13         {
14             migrationBuilder.CreateTable(
15                 name: "products",
16                 columns: table =>
17                     table.AddColumn("id", sqlType: "int", constraints: c =>
18                         c.PrimaryKey());
19
20             migrationBuilder.InsertData(
21                 table: "products",
22                 columns: new[] { "id", "name", "category", "price" },
23                 values: new object[] { 1, "Laptop", "Electronics", 1200 });
24
25         }
26
27         protected override void Down(MigrationBuilder migrationBuilder)
28         {
29             migrationBuilder.DropTable(
30                 name: "products");
31
32         }
33     }
34 }
```

Add API Controller with actions, using Entity Framework

Model class: Product (WebApplication35)
DbContext class: ApplicationDbContext (WebApplication35)
Database provider: Configured from the selected DbContext
Controller name: ProductsC

Add Cancel

Add to Source Control Select Repository



Type here to search



20°C



00:14
30-04-2024

After running the project, you will see the following screen

The screenshot shows a web browser window with the title "Swagger UI". The address bar indicates the URL is "localhost:5074/swagger/index.html". The main content area displays the Swagger UI interface for a "WebApplication35" API.

Selected definition: WebApplication35

WebApplication35 1.0 OAS3

<http://localhost:5074/swagger/v1/swagger.json>

ProductsC

- GET** /api/ProductsC
- POST** /api/ProductsC
- GET** /api/ProductsC/{id}
- PUT** /api/ProductsC/{id}
- DELETE** /api/ProductsC/{id}

WeatherForecast

Type here to search 20°C 00:16 30-04-2024 ENG

In beginning, we have added nothing so in get method no values will be displayed. So try executing put

The screenshot shows a browser window with the title "Swagger UI" at the top. The address bar displays "localhost:5074/swagger/index.html". Below the address bar, there are several icons for importing favorites and other services like Gmail and YouTube.

The main content area displays a list of products:

```
[  
  {  
    "id": 0,  
    "name": "string",  
    "amount": 0  
  }  
]
```

Below this, a modal dialog is open for a POST request to "/api/ProductsC". The dialog has a green header bar with "POST /api/ProductsC".

The "Parameters" section is collapsed, showing "No parameters".

The "Request body" section is expanded, showing a dropdown menu set to "application/json".

The JSON request body template is:

```
{  
  "id": 0,  
  "name": "string",  
  "amount": 0  
}
```

The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray displaying the date and time as "30-04-2024".

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

Request body

application/json

```
{  
    "name": "Rice",  
    "amount": 120  
}
```

Execute

Responses

Code	Description	Links
200		No links

Type here to search

00:19
30-04-2024

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

Server response

Code Details

201 *Undocumented*

Response body

```
{  
    "id": 1,  
    "name": "Rice",  
    "amount": 120  
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8  
date: Mon, 29 Apr 2024 18:49:33 GMT  
location: http://localhost:5074/api/ProductsC/1  
server: Kestrel  
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

Media type

text/plain

Controls Accept header.

Example Value | Schema

Type here to search

00:19 20°C Speakers: 100% 2024

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

Request body

application/json

```
{  
    "name": "Wheat",  
    "amount": 100  
}
```

Execute Clear

Responses

Curl

Type here to search

00:20 20°C ENG 30-04-2024

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

```
{  
    "name": "Oats",  
    "amount": 200  
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \  
  'http://localhost:5074/api/ProductsC' \  
  -H 'accept: text/plain' \  
  -H 'Content-Type: application/json'
```

Type here to search

00:21 20°C ENG 30-04-2024

Now, execute get

The screenshot shows a web browser window with the title "Swagger UI". The address bar displays "localhost:5074/swagger/index.html". The main content area shows the Swagger UI interface for a "ProductsC" endpoint.

Endpoint: GET /api/ProductsC

Parameters: No parameters

Responses: (No responses listed)

Curl:

```
curl -X 'GET' \
  'http://localhost:5074/api/ProductsC' \
  -H 'accept: text/plain'
```

Request URL: http://localhost:5074/api/ProductsC

The browser's taskbar at the bottom shows various pinned icons, including File Explorer, Mail, and Edge, along with the system clock showing "00:21 30-04-2024".

Now all items we are getting with get method

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

```
curl -X 'GET' \
'http://localhost:5074/api/Products' \
-H 'accept: text/plain'
```

Request URL

http://localhost:5074/api/Products

Server response

Code Details

200

Response body

```
[{"id": 1, "name": "Rice", "amount": 120}, {"id": 2, "name": "Wheat", "amount": 100}, {"id": 3, "name": "Oats", "amount": 200}]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,29 Apr 2024 18:52:09 GMT
server: Kestrel
transfer-encoding: chunked
```

Type here to search

00:22 20°C ENG 30-04-2024

Now, we will use put method, click on try it out

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

```
    "id": 0,
    "name": "string",
    "amount": 0
}
```

GET /api/ProductsC/{id}

PUT /api/ProductsC/{id}

Parameters

Name Description

id * required
integer(\$int32)
(path)

Request body

application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "amount": 0
}
```

Type here to search

00:23 20°C ENG 30-04-2024

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail YouTube Maps

id * required
integer(\$int32)
(path)

2

Request body

application/json

```
{  
    "id": 2,  
    "name": "tea",  
    "amount": 10  
}
```

Execute Clear

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail YouTube Maps

```
curl -X 'PUT' \
  'http://localhost:5074/api/ProductsC/2' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 2,
    "name": "tea",
    "amount": 10
}'
```

Request URL

http://localhost:5074/api/ProductsC/2

Server response

Code Details

204 Undocumented

Response headers

```
date: Mon, 29 Apr 2024 18:56:59 GMT
server: Kestrel
```

Responses

Code Description Links

200 Success No links

DELETE /api/ProductsC/{id}



Type here to search



00:27
30-04-2024

Again you have to go to get method, for checking whether output is updated or not

Curl

```
curl -X 'GET' \
'http://localhost:5074/api/Products' \
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5074/api/Products
```

Server response

Code Details

200 Response body

```
[{"id": 1, "name": "Rice", "amount": 120}, {"id": 2, "name": "tea", "amount": 10}, {"id": 3, "name": "Oats", "amount": 200}]
```

Response headers

Type here to search

20°C ENG 00:27 30-04-2024

Now we will perform delete operation

Just mention id that you want to delete

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

DELETE /api/ProductsC/{id}

Parameters

Name	Description
id * required	integer(\$int32) (path)

Cancel

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:5074/api/ProductsC/2' \
-H 'accept: */*' Copy
```

Request URL

```
http://localhost:5074/api/ProductsC/2 Copy
```

Type here to search 20°C ENG 00:29 30-04-2024

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:5074/api/ProductsC/2' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5074/api/ProductsC/2
```

Server response

Code	Details	Links						
204 <small>Undocumented</small>	Response headers <code>date: Mon,29 Apr 2024 18:58:57 GMT</code> <code>server: Kestrel</code>							
200	Responses <table border="1"><thead><tr><th>Code</th><th>Description</th><th>Links</th></tr></thead><tbody><tr><td>200</td><td>Success</td><td>No links</td></tr></tbody></table>	Code	Description	Links	200	Success	No links	
Code	Description	Links						
200	Success	No links						



Type here to search



00:29
30-04-2024

Again we will check get method, to know whether its deleted or not. Now we will see id 2 is deleted.

Curl

```
curl -X 'GET' \
'http://localhost:5074/api/ProductsC' \
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5074/api/ProductsC
```

Server response

Code Details

200 Response body

```
[  
  {  
    "id": 1,  
    "name": "Rice",  
    "amount": 120  
  },  
  {  
    "id": 3,  
    "name": "Oats",  
    "amount": 200  
  }]
```

Download

Response headers

```
content-type: application/json; charset=utf-8  
date: Mon, 29 Apr 2024 19:00:31 GMT  
server: Kestrel  
transfer-encoding: chunked
```

Type here to search

00:30 20°C ENG 30-04-2024

Testing Web API Using Postman-Used for basically testing API's, you project must be running first for testing whether API's are working or not.

POSTMAN

- APIs are commonly used in development.
- Postman is a most powerful HTTP Client that can be used for API Testing.
- It provides a user-friendly interface for developers to create, test, and document APIs.
- POSTMAN is used to send HTTP requests to web servers and receive responses.
- It supports various HTTP request methods such as GET, POST, PUT, DELETE, and others.
- With POSTMAN, developers can test the functionality and performance of APIs and diagnose any issues with them.

Testing Web API Using Postman

The screenshot shows the Postman application interface. At the top, there is a navigation bar with links for Product, Pricing, Enterprise, Resources and Support, Public API Network, and a search bar. On the right side of the header, there are buttons for Contact Sales, Sign In, and Sign Up for Free.

The main workspace displays a request for "Retrieve a database" from the Notion API. The request method is GET, and the URL is <https://api.notion.com/v1/databases/:id>. The "Params" tab is selected, showing a "Query params" table with a single entry: "id" with value "((DATABASE_ID))". The "Path variables" table also has a single entry: "id" with value "((DATABASE_ID))". The "Body" tab shows a JSON response with one object containing a "Publisher" field with id "N3EN24Ph" and name "Publisher".

On the left side of the interface, there is a sidebar with sections for Collection, Environment, APIs, Mock Servers, Monitors, Flows, and History. A yellow curved highlight is drawn over the sidebar area.

M —
APIs together

Over 30 million developers use Postman. Get started by signing up or downloading the desktop app.

[Sign Up for Free](#)

Download the desktop app for

We use cookies on your device to enhance your navigation experience, analyze usage to improve our site, and customize our marketing efforts. You can learn more about cookies in our [Cookie Notice](#).

[Manage Cookies](#) [Reject All Cookies](#) [Accept All Cookies](#)



Product

Pricing

Enterprise

Resources and Support

Public API Network

Download the app to get started using the Postman API I browser experience, you can try the web v

The Postman app

Download the app to get started with the Postman API Platform.

Windows 64-bit

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#)

Not your OS? Download for Mac (Intel Chip, Apple Chip) or Linux (x64, arm64)

The screenshot shows the Postman application interface. At the top, there's a navigation bar with links for Home, Workspaces, and a search icon. Below that is a sidebar with sections for Collections, Environments, APIs, Mock Servers, and Monitors. The main area displays a "Notion's Public Workspace" with a tree view of collections like "Notion API", "Users", and "Databases". A specific API endpoint under "Databases" is highlighted with a red border, showing a "GET Retrieve" operation. The bottom of the interface has a footer with links for Help, Support, and more.



Join 25 million developers who use Postman

Simplify your API design, development and testing workflows with Postman.

- ⚡ Save and organize your work in collections and workspaces
- ⚡ Share your work with clients and collaborate with teammates
- ⚡ Integrate with tools like VS Code, GitHub and BitBucket



Create a free Postman account

Work email

[Create Free Account](#)

Already have an account? [Log In](#)

Or continue with the [lightweight API client](#).



Type here to search



29°C



16:11
29-04-2024



Home Workspaces Explore

Search Postman

Sign In Create Account

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History New Import

GET Untitled Request

HTTP Untitled Request

Save

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Time to send your first request

All the requests you send will be stored in History. Sign in or create an account to organize them in collections.

Show me how

Response

Enter the URL and click Send to get a response

Console Not connected to a Postman account

Type here to search

Windows Taskbar

16:12 29-04-2024

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

New Import

POST http://localhost:5055/ap

GET http://localhost:5074/api/

+

...

Today

- POST http://localhost:5055/api/CustomersC
- POST http://localhost:5055/api/CustomersC
- POST http://localhost:5055/api/CustomersC
- POST http://localhost:5055/api/CustomersC
- GET http://localhost:5055/api/CustomersC

HTTP http://localhost:5074/api/ProductsC

Save

GET

http://localhost:5074/api/ProductsC

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Response



20°C



ENG

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

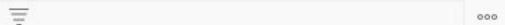
History

New Import

POST http://localhost:5055/ap

GET http://localhost:5074/api/

+ ...



Today

GET http://localhost:5074/api/ProductsC

Yesterday

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

GET http://localhost:5055/api/CustomersC

http://localhost:5074/api/ProductsC

Save

</>

GET

http://localhost:5074/api/ProductsC

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key

Value

Bulk Edit

Body

Cookies

Headers (4)

Test Results

Status: 200 OK Time: 93 ms Size: 221 B

Save Response

Pretty

Raw

Preview Visualize

JSON

Bulk

Edit

```
1 [
2   {
3     "id": 1,
4     "name": "Rice",
5     "amount": 120
6   },
7   {
8     "id": 3,
9     "name": "Oats",
10    "amount": 200
11  }
12 ]
```



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman**.

History

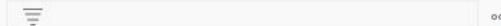
New Import

POST http://localhost:5055/ap

GET http://localhost:5074/api/

POST http://localhost:5074/ap

+ ...



http://localhost:5074/api/ProductsC

Save

</>

POST http://localhost:5074/api/ProductsC

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

<input checked="" type="checkbox"/> Connection	keep-alive
<input checked="" type="checkbox"/> Content-Type	application/json
Key	Value

Response



Click Send to get a response



20°C

00:59
30-04-2024

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

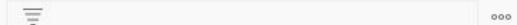
New Import

POST http://localhost:5055/ap

GET http://localhost:5074/api/

POST http://localhost:5074/ap

+ ...



Today

GET http://localhost:5074/api/ProductsC

Yesterday

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

POST http://localhost:5055/api/CustomersC

GET http://localhost:5055/api/CustomersC

HTTP http://localhost:5074/api/ProductsC

Save

</>

POST

http://localhost:5074/api/ProductsC

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

JSON

Beautify

```
1
2
3     "name": "Coffee",
4     "amount": 1200
5
```

Response



Now again go to the get request for checking whether product has added or not

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'Explore', 'Search Postman', 'Sign In', and 'Create Account'. Below the header, a message states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.' The left sidebar, titled 'History', shows requests from 'Today' and 'Yesterday'. The main workspace displays a 'GET' request to 'http://localhost:5074/api/ProductsC'. The 'Body' tab shows the response body as JSON:

```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
{
  "id": 3,
  "name": "Oats",
  "amount": 200
},
{
  "id": 4,
  "name": "String",
  "amount": 0
},
{
  "id": 5,
  "name": "Coffee",
  "amount": 1200
}
```

The status bar at the bottom indicates 'Status: 200 OK'.

Put

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, Explore, a search bar, and account options. The main area displays a history of API calls and a current request configuration.

Request Details:

- Method: PUT
- URL: <http://localhost:5074/api/ProductsC/4>
- Body tab selected, showing JSON content:

```
1 {  
2   "id": 5,  
3   "name": "Coffee",  
4   "amount": 1200  
5 }
```

Response:

- Status: 400 Bad Request
- Body (Pretty):

```
1 {  
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.1",  
3   "title": "Bad Request",  
4   "status": 400,  
5   "traceId": "00-460780ebe4df21cf3cf7d0b87ac17466-3ed7343af8e89615-00"  
6 }
```

At the bottom, there are icons for Console, Not connected to a Postman account, and various system status indicators.

Now at id 5, I am adding coffee11, you can check using get

The screenshot shows the Postman application interface. On the left, there's a sidebar titled "History" with sections for "Today" and "Yesterday", each listing several API requests (POST, GET, PUT) made to localhost:5074/api/ProductsC. The main workspace is focused on a "PUT" request to `http://localhost:5074/api/ProductsC/5`. The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   ... "id": 5,  
3   ... "name": "Coffee11",  
4   ... "amount": 1200  
5 }
```

Below the body, the "Status" field shows "204 No Content". At the bottom, there are tabs for "Pretty", "Raw", "Preview", "Visualize", and "Text". A "Save Response" button is also visible.

Swagger UI

localhost:5074/swagger/index.html

Import favorites | Gmail | YouTube | Maps

<http://localhost:5074/api/Products>

Server response

Code Details

200 Response body

```
[{"id": 1, "name": "Rice", "amount": 120}, {"id": 2, "name": "Oats", "amount": 200}, {"id": 3, "name": "string", "amount": 0}, {"id": 4, "name": "Coffee11", "amount": 1200}, {"id": 5, "name": "string", "amount": 0}]
```

Download

Response headers

content-type: application/json; charset=utf-8

Type here to search

20°C 01:10
30-04-2024 ENG

Conventional Routing

CONVENTION ROUTING

- It is used for defining routes based on conventions and defaults,
- rather than explicitly defining each route.
- Developers define a set of conventions that are used to generate routes for different types of controllers and actions.
- It provides a **simple and flexible** way to define routes in an ASP.NET Core application.

- The URL pattern is generated based on the names of the controller and action methods.

Program.cs X

coreMvcFundamentals

```
10     {
11         app.UseExceptionHandler("/Home/Error");
12     }
13     app.UseStaticFiles();
14
15     app.UseRouting();
16
17     app.UseAuthorization();
18
19     app.MapControllerRoute(
20         name: "default",
21         pattern: "{controller=Customer}/{action=Index}/{id?}");
22
23     app.Run();
24
```



class System.String

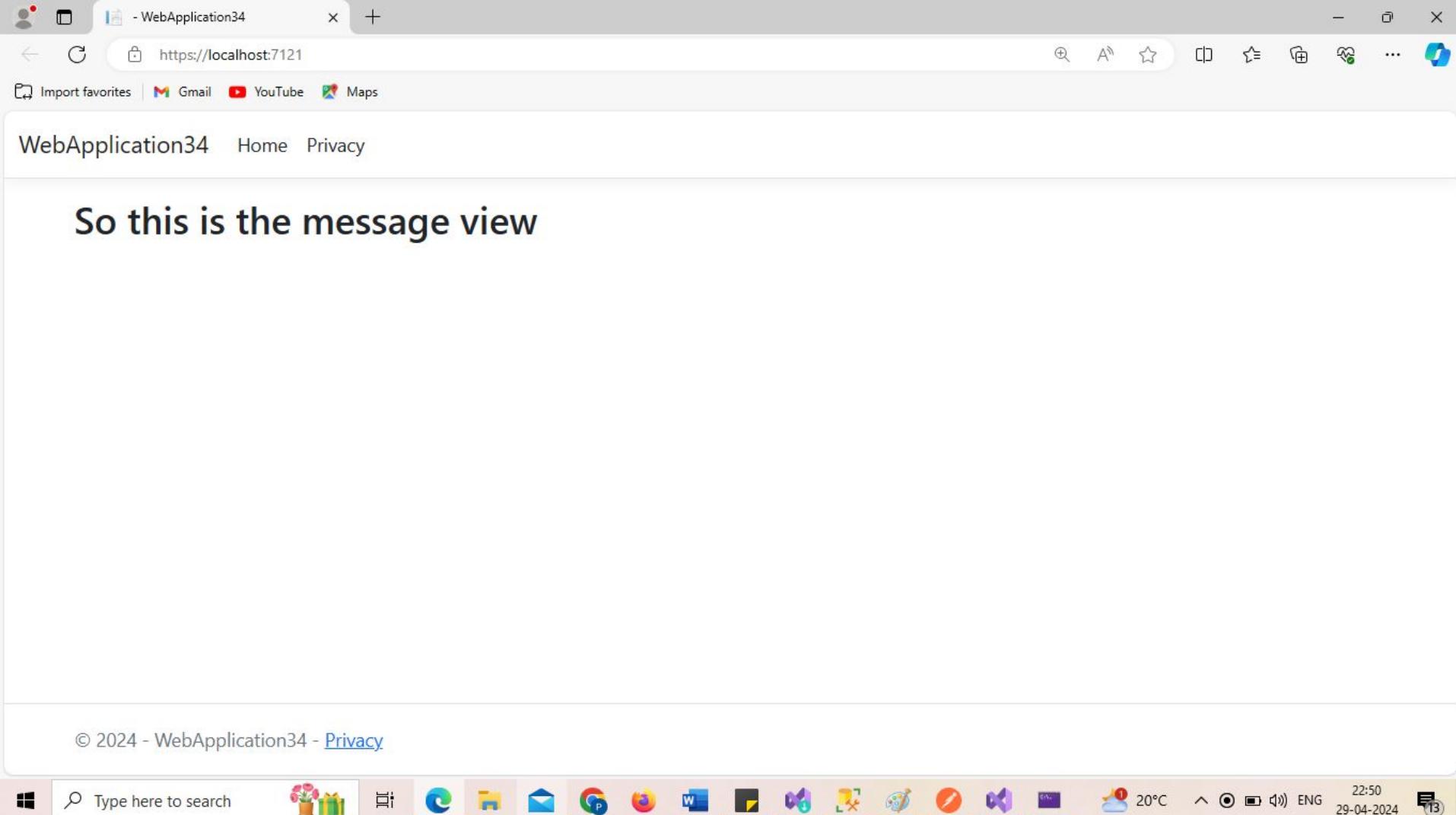
Represents text as a sequence of UTF-16 code units.

Attribute Routing

ATTRIBUTE ROUTING

- It is a technique used for defining routes directly in the controller code using attributes.
- Developers can specify the URL pattern for a particular action method using attributes,
- instead of relying on conventions or explicit route definitions.
- In attribute routing, [Route] attribute is used to specify the URL pattern for an action method.
- By specifying the route directly in the controller code, developers can have fine-grained control over the URL patterns for each action method.

```
ebApplication34          - WebApplication34.Controllers.Palvi
1   using Microsoft.AspNetCore.Mvc;
2
3   namespace WebApplication34.Controllers
4   {
5       public class Palvi : Controller
6       {
7           public IActionResult Index()
8           {
9               return View();
10
11              [Route("")]
12              [Route("/Sample/message")]
13
14              public IActionResult Message()
15              {
16                  return View();
17
18              }
19      }
```



Type here to search



20°C
29-04-2024

22:50
13

If I want that by default message method will open, then use ["/Route"], and if I want some specific URL will open that path, then add [Route("Jai/Message")]

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "WebApplication69". The code editor displays the file "Basic.cs" with the following content:

```
public class Basic : Controller
{
    [Route("Jai/Message")]
    public IActionResult Index()
    {
        return View();
    }
    [Route("/")]
    public IActionResult Message()
    {
    }
}
```

The code editor includes syntax highlighting and a vertical ruler on the left side. The Solution Explorer is visible on the right, showing the project structure.