Microsoft
.NET

Web Development

C# and .Net

Yash Lathiya

Table of Contents

Introduction to Web Development

- ASP .NET provides followed frameworks :
  ASP.NET MVC
  ASP.NET Web API
  ASP.NET Web Pages

ASP .NET Web Pages

It combines the code of html, css, javascript & server code in c#. "Server code" typically refers to the programming code that runs on a server to handle various tasks, processes, and requests from clients. In the context of web development, server code is responsible for processing and responding to requests sent by users' browsers.

It is single page application.

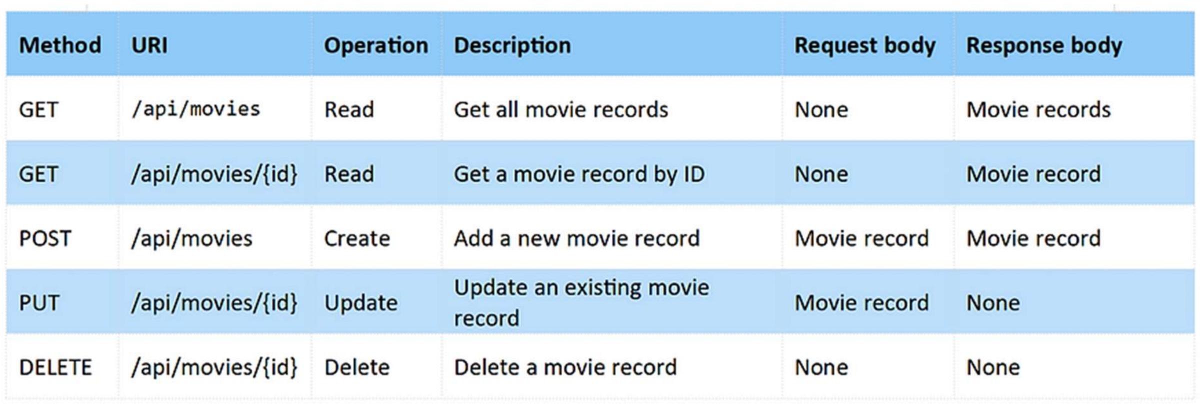
ASP .NET MVC

It is a .NET framework which provides facility of html, css, javascript with mvc architecture, where mvc stands for ..
M – Model (Application’s data and business logic)
V – View (Presenting data to the user & receiving user input)
C – Controller (intermediatory between Model and View)

ASP .NET Web API

ASP .NET Web API is a framework for building HTTP services that can be accessed from any client including browsers and mobile devices.
It develops RESTful (Representational State Transfer) applications.

Building Web API Project with HTTP Action Methods



Method	URI	Operation	Description	Request body	Response body
GET	/api/movies	Read	Get all movie records	None	Movie records
GET	/api/movies/{id}	Read	Get a movie record by ID	None	Movie record
POST	/api/movies	Create	Add a new movie record	Movie record	Movie record
PUT	/api/movies/{id}	Update	Update an existing movie record	Movie record	None
DELETE	/api/movies/{id}	Delete	Delete a movie record	None	None

- HTTP consists following methods which are implemented in demo for Student API..

GET

- Get Method is used for the purpose of viewing / getting / extracting in the database.
- In the project, It is implemented for getting all objects as well as any particular object by specific id (e.g. Aadhar Number, Enrollment ID, etc .)

POST

- Post method is used for the purpose of addition in the database.

PUT

- Put method is used for the updation within the dtabase.
- Any database record can be updated by using put method.
- Here, updation means we have to provide all details of the object, PUT method is not capable to make changes on specific fields, if we want to update only one field in record still we have to provide other all fields too.

DELETE

- Delete method is used for the deletion of the record in database

PATCH

- Patch method is used for the updation within the dtabase.
- Any database record can be updated by using patch method.
- Here, updation means we have to provide only details of the object which we want to update, other details will be as it is as before.

HTTP Action Method Status Codes

HTTP method uses different types of status codes for different business logic which are explained below :

1xx

Informational Response

100 – Continue

2xx

Successful, It means request is successful.

200 – Ok

201 – New resource created.

204 – No Content

3xx

Redirection, when resource pages or files are moved to another location.

301 – Moved permanently resource which is asked

302 – Moved temporarily resource which is asked

4xx

Client errors, denotes error within the request.

400 – Bad Request

401 – Unauthorized (Invalid authentication)

403 – Authentication is success but authorization has error.

404 – Requested resource is not available or found.

405 – Not Supported

5xx

Error from server side.

500 – Internal server error

503 – Service Unavailable

504 – Gateway Timeout

Security

CORS

“Cross Origin Resource Sharing” – CORS is a HTTP protocol which allows web applications to access their rosources from the different origin, here origin means same host & same port as well as same protocol, if only path differs then it is same origin.

CORS is required due to Same Origin Policy of Web, https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Example of urls from different origin.

- http://www.stackhawk.com
- https://www.stackhawk.com
- https://stackhawk.com/why-stackhawk/

Enable CORS

- First Parameter   : Allows requests from any origin
- Second Parameter : Allows all HTTP Methods (GET, POST, PUT , DELETE, etc.)
- Third Parameter   : ALlowed Headers

```
var cors = new EnableCorsAttribute("*", "*", "*");
config.EnableCors(cors);
```

Authentication

Authentication is all about knowing identity of the user. Suppose Sachin Tendulkar is trying to access the web api.

- Using Host
- Using HTTP Message Handlers

Authorization

Authorization is deciding whether user is allowed to perform action or not. Suppose Sachin Tendulkar is trying to add student in api, but it's not allowed as he is student and that right is given only to HOD.

Demo : StudentAPI consists the demonstration of

- HTTP Action Methods
- Status Codes
- CORS
- Authentication
- Authorization

Exception

Mostly all unsolved exception in ASP.NET shows 500 – Internal Sever Error, still if we want to add Exception Handling in ASP .NET web API can be done by several following methods :

HttpResponseException

This approach allows to return specified status code with specified message too.

```
var resp = new HttpResponseMessage(HttpStatusCode.NotFound)
{
       Content = new StringContent(string.Format("No product with ID = {0}", id)),
      ReasonPhrase = "Product ID Not Found"
 };
 throw new HttpResponseException(resp);
```

Filter Exception

Exception filter is a approach of customized exception in ASP .NET we API. Suppose we want specific exception on specific condition, this exception is used.

e.g. Method implementation is pending then we can add our own NotImplementedException when user tries to access that method.

That is demonstrated in WebAPI > StudentAPI demo > NotImplementedException

We have to create ExceptionNameExceptionFilterAttribute class which inherits the class ExceptionFilterAttribute & override the method GetException() by adding details such as statusCode, responseMessage, Content, ReasonPhrase etc.

We can add that exception in controller at

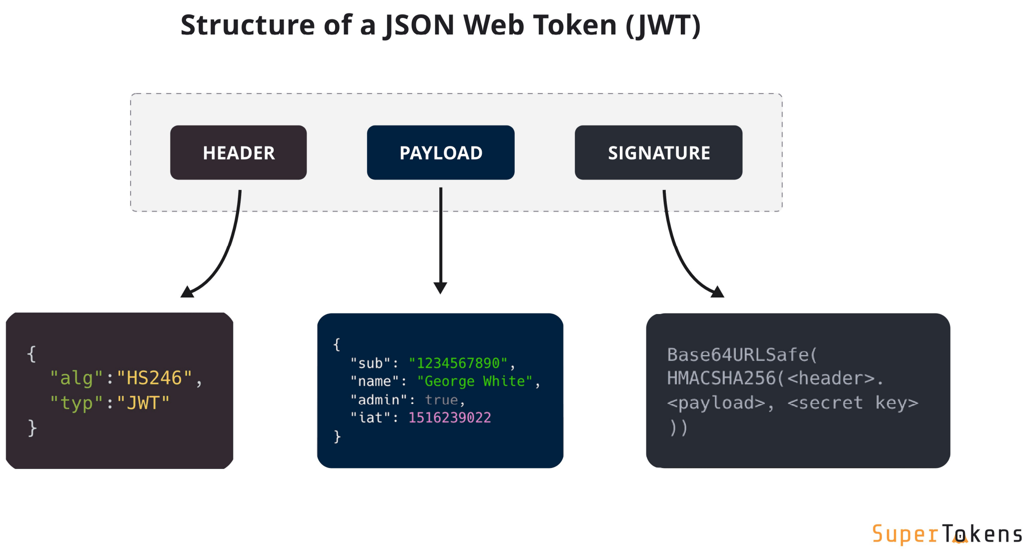
- Specific Method
- Top of the Controller (For all methods within the controller)
- Global.config (For all controllers)

HttpError

HttpError provides Error to the specified logic or method.

```
var message = String.Format($"There is no student in the database with the enrollment : {enrollment}");
return (IHttpActionResult)Request.CreateErrorResponse(HttpStatusCode.NotFound, message);
```

JWT Token


Structure of a JSON Web Token (JWT)
HEADER
PAYLOAD
SIGNATURE
{ "alg":"HS246", "typ":"JWT" }
{ "sub": "1234567890", "name": "George White", "admin": true, "iat": 1516239022 }
Base64URLSafe(HMACSHA256(<header>. <payload>, <secret key>))
SuperTokens

JWT token is used to improve the time complexity of the client – server transactions. When user is authenticated once, JWT Token is generated & provided to client then after client sends the request with the token and if token is valid then server does not go for userId and password again .

```
Install-Package Microsoft.Owin.Security.OAuth
Install-Package Microsoft.Owin.Cors
Install-Package System.IdentityModel.Tokens.Jwt
```

Caching

Caching is a technique of storing frequently used data or information in a local memory, for a certain time period. So, next time, when the client requests the same information, instead of retrieving the information from the database, it will give the information from the local memory. The main advantage of caching is that it improves the performance by reducing the processing burden.

It is implemented by using CacheFilter, it responses cache in cache control response header as “ Yes ” for the given time limit.

That is implemented in project WebAPI > StudentAPI > Caching

Apart from cacheFilter we can use caching by other ways too.which are entitled below :

Memory Cache

MemoryCache is suitable for scenarios where you want to store data in memory for quick access within a single application. It's commonly used for caching data that is expensive to compute or retrieve from a data source.

Distributed Chaining

Distributed caching is useful in scenarios where you have multiple instances of your application or multiple servers and need to share cached data among them. Redis, in this example, is a popular distributed caching solution.

HttpContext.Cache

HttpContext.Cache is specific to ASP.NET applications and is useful for caching data that is tied to a particular request or needs to be shared among different parts of the application during the request lifecycle.

ObjectCache

ObjectCache provides a common interface for different caching implementations. It's beneficial when you want to abstract your code from a specific caching provider and work with a common set of caching features.

Output Caching

Output caching is used to cache the entire output of a page or a specific action method in an ASP.NET application. This is beneficial when the content of a page or action is relatively static and can be reused for multiple requests.

Remember that the choice of caching mechanism depends on factors like data size, expiration requirements, distribution needs, and the specific characteristics of your application. Each caching approach has its strengths and is suitable for different scenarios.

Versioning

Versioning is a technique in which users can access different versions of web api.

Suppose there are multiple users who are accessing api, and some of them want updated version & some wants previously stable version, in such case versioning is used.

It can be implemented by several methods explained below :

- URI

  In this method different url & controller is assigned in web api. Also that is set in web.config file too .

```
// API Version 1
config.Routes.MapHttpRoute(
    name: "ApiVersion1",
    routeTemplate: "api/v1/student1",
    defaults : new { controller = "STStudentV1"}
);

// API Version 2
```

```
            config.Routes.MapHttpRoute(
                name: "ApiVersion2",
                routeTemplate: "api/v1/student2",
                defaults: new { controller = "STStudentV2" }
            );
```

- QueryString parameter

Version is passed in query string of url -> Specifying with request

It can be implemented by CustomController which selects version as per the requirement. Also we have to add followed code in the webConfig.cs file and also remove all routings from the controller.

url/api/ststudent?v=1

url/api/ststudent?v=2

```
// Replace the customController with the default httpControllerselector of the web api
            config.Services.Replace(typeof(IHttpControllerSelector),
                                    new CustomControllerSelector(config));

            // Routing path of the customControllerSelector
            config.Routes.MapHttpRoute(
                name: "DefaultRoute",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
```

- Custom Header parameter

Custom Headers are used for providing

- additional information,
- troubleshooting and
- implementing server-side logic, etc.

In this method we specifies the version number of API, on basis of that controllers are selected & methods of specified version is selected.

Code implementation in the project WebAPI > VersioningWebAPI > Custom > CustomControlSelectorCustomHeader

In Request Header

Student-API-Version : 2

- Accept Header parameter

  Accepts Headers requests the server about the file format of the data required by the browser. This data is expressed as MIME Types which stands for “Multipurpose Internet Mail Exchange”. The MIME type is generally case-insensitive.

Code implementation in the project WebAPI > VersioningWebAPI > Custom > CustomControllerSelectorAcceptHeader

In Request Header

Accept : application/json;version=2

Use of Swagger

Swagger is a set of open-source tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs with help of major Swagger tools.

API Specification,

Documentation:

Swagger allows developers to generate interactive and user-friendly API documentation automatically from the OAS file. This documentation includes details such as endpoints, request and response formats, authentication methods, and example API calls.

The documentation is often presented in a web-based format, allowing users to test API endpoints directly from the documentation.

API Design and Development:

(Major use of swagger)

During the API design phase, Swagger can be used to define the structure of the API, including the available endpoints, request and response models, and supported authentication mechanisms.

Code Generation:

Swagger provides tools to automatically generate client libraries, server stubs, and API documentation for various programming languages. This accelerates development by reducing the manual effort required to create and maintain code that interacts with the API.

Testing:

Swagger simplifies the testing process by providing an interactive API documentation page where developers can make test requests and observe the responses.

Tools like Swagger UI enable users to explore and test API endpoints directly from their web browsers.

Versioning and Evolution:

Swagger supports versioning of API specifications, making it easier to manage changes and updates to an API over time.

Developers can use Swagger to document version-specific details, helping clients adapt to changes in a controlled manner.

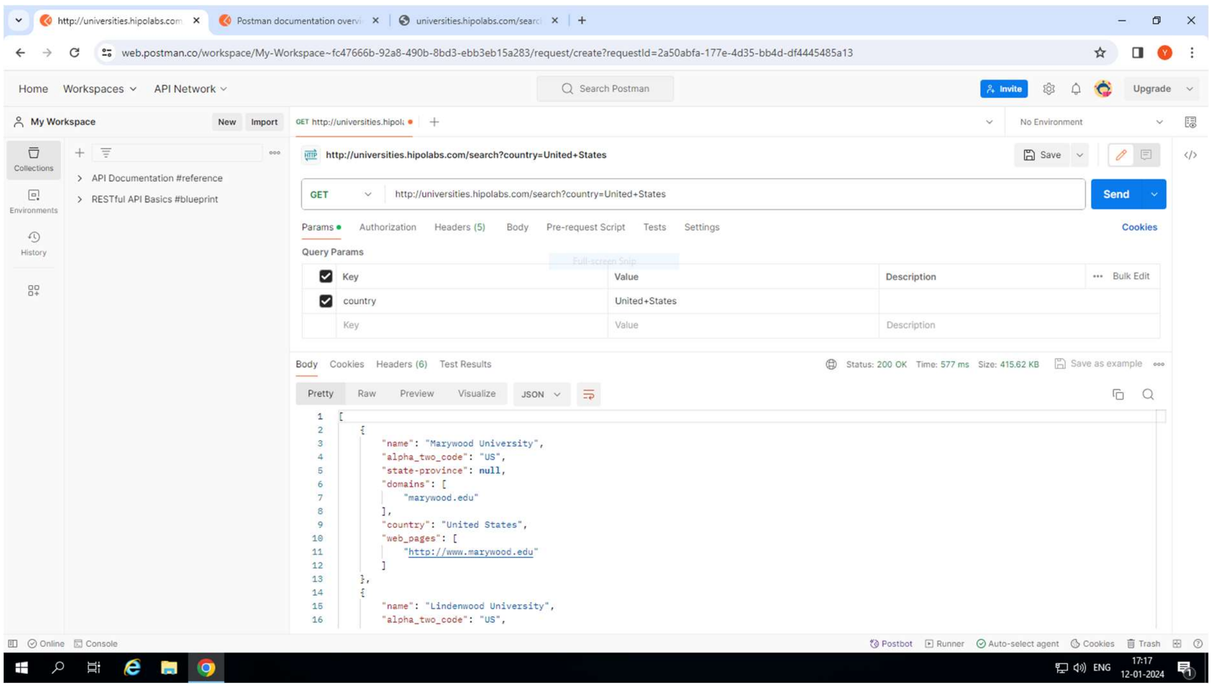
Use of Postman

Developers can use desktop as well as online web version of postman for the purpose of api testing, If we are testing web api from the localhost – it requires Desktop agent installed in system.

(Thunder Client Extension in VS Code can also be used for purpose of web API testing.)

Postman provides facility of followed :

- Send Requests
  To send dynamic requests to web api & get response of that



web.postman.co/workspace/My-Workspace~fc47666b-92a8-490b-8bd3-ebb3eb15a283/request/create?requestId=2a50abfa-177e-4d35-bb4d-df4445485a13

GET http://universities.hipolabs.com/search?country=United+States

Query Params

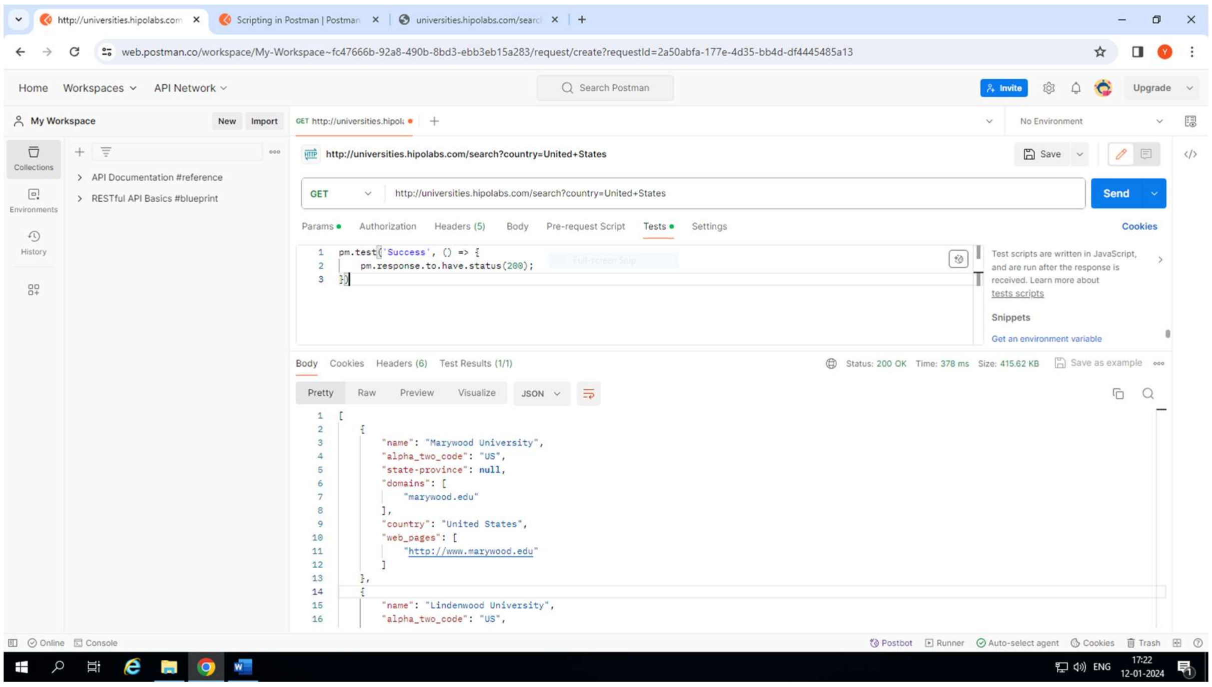
	Key	Value	Description
✓	country	United+States	

Status: 200 OK Time: 577 ms Size: 415.62 KB

```
[
    {
        "name": "Marywood University",
        "alpha_two_code": "US",
        "state-province": null,
        "domains": [
            "marywood.edu"
        ],
        "country": "United States",
        "web_pages": [
            "http://www.marywood.edu"
        ]
    },
    {
        "name": "Lindenwood University",
        "alpha_two_code": "US",
```

- Write Scripts

  Postman’s runtime is developed in node.js it allows dynamic request, we can do it by writing scripts for web api.



GET http://universities.hipolabs.com/search?country=United+States

Status: 200 OK Time: 378 ms Size: 415.62 KB

```
pm.test('Success', () => {
    pm.response.to.have.status(200);
})
```

```
[
    {
        "name": "Marywood University",
        "alpha_two_code": "US",
        "state-province": null,
        "domains": [
            "marywood.edu"
        ],
        "country": "United States",
        "web_pages": [
            "http://www.marywood.edu"
        ]
    },
    {
        "name": "Lindenwood University",
        "alpha_two_code": "US",
```

- Use Collection

  Postman collection is set of requests accessed by the user, that can managed by potman collection.

- Use Postman Flow

  Postman Flows is a visual tool for creating API workflows. You can use Flows to chain requests, handle data, and create real-world workflows in your Postman workspace.

- Use Postman CLI

  The Postman CLI is a secure command-line companion for Postman. You can use the Postman CLI to run a collection, send run results to Postman

- Document our API

  Documentation plays vital role in any we api, that can be created and accessed with the help of Postman.

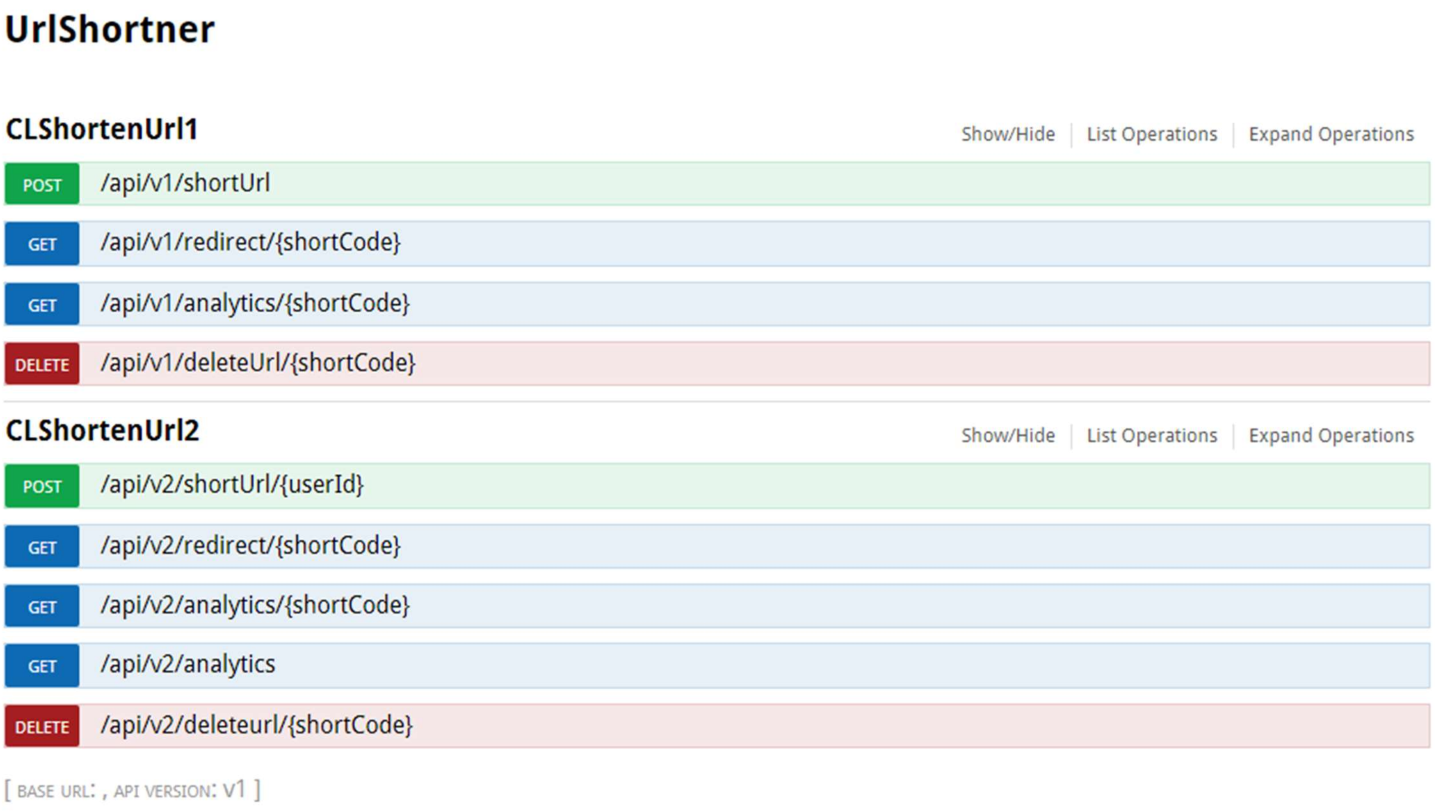
Deployment

Web API can be published by following basic steps for publishing in Visual Studio.

Right click on Project > Publish.

It generates one package and that can be accessed by localhost url.

URL Shortner



UrlShortner

CLShortenUrl1 — Show/Hide | List Operations | Expand Operations

POST	/api/v1/shortUrl
GET	/api/v1/redirect/{shortCode}
GET	/api/v1/analytics/{shortCode}
DELETE	/api/v1/deleteUrl/{shortCode}

CLShortenUrl2 — Show/Hide | List Operations | Expand Operations

POST	/api/v2/shortUrl/{userId}
GET	/api/v2/redirect/{shortCode}
GET	/api/v2/analytics/{shortCode}
GET	/api/v2/analytics
DELETE	/api/v2/deleteurl/{shortCode}

[BASE URL: , API VERSION: V1]

It consists 2 controller which implements different version of api. Version1 & Version2.

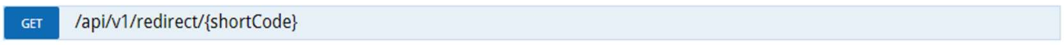
Version 1

POST /api/v1/shortUrl

It generates short URL with the short code of 6 characters,

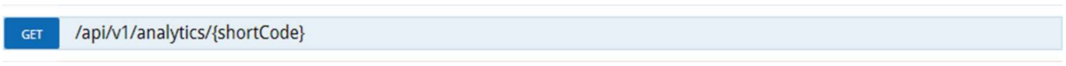
It requires original URL link in request body.

It implements role based authorization for user.


GET /api/v1/redirect/{shortCode}

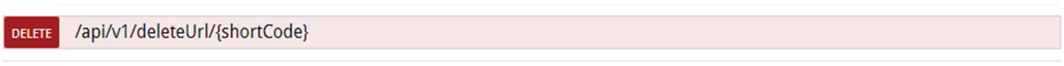
It redirects the short URL to original URL,

It is publicly accessible.


GET /api/v1/analytics/{shortCode}

It provides analytics of short url,

It implements role based authorization for user.

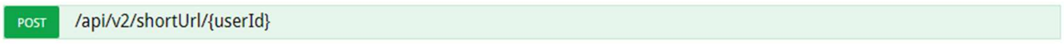

DELETE /api/v1/deleteUrl/{shortCode}

It deletes short url within the database,

It implements role based authorization for admin.

Version 2

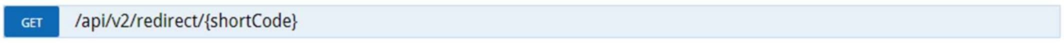
Version 2 has additional features of User Id as well as it generates the short url with short code of 8 characters. Also added a new role super admin which can view analytics of all url.


POST /api/v2/shortUrl/{userId}

It generates short URL with the short code of 8 characters,

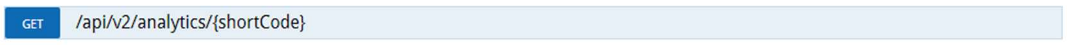
It requires original URL link in request body & user Id as url parameter.

It implements role based authorization for user.


GET /api/v2/redirect/{shortCode}

It redirects the short URL to original URL,

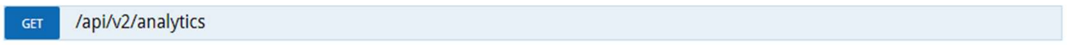
It is publicly accessible.



GET /api/v2/analytics/{shortCode}

It provides analytics of short url,

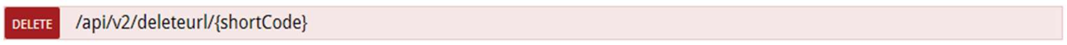
It implements role based authorization for user.



GET /api/v2/analytics

It provides analytics of all short url,

It implements role based authorization for super admin.



DELETE /api/v2/deleteurl/{shortCode}

It deletes short url within the database,

It implements role based authorization for admin.