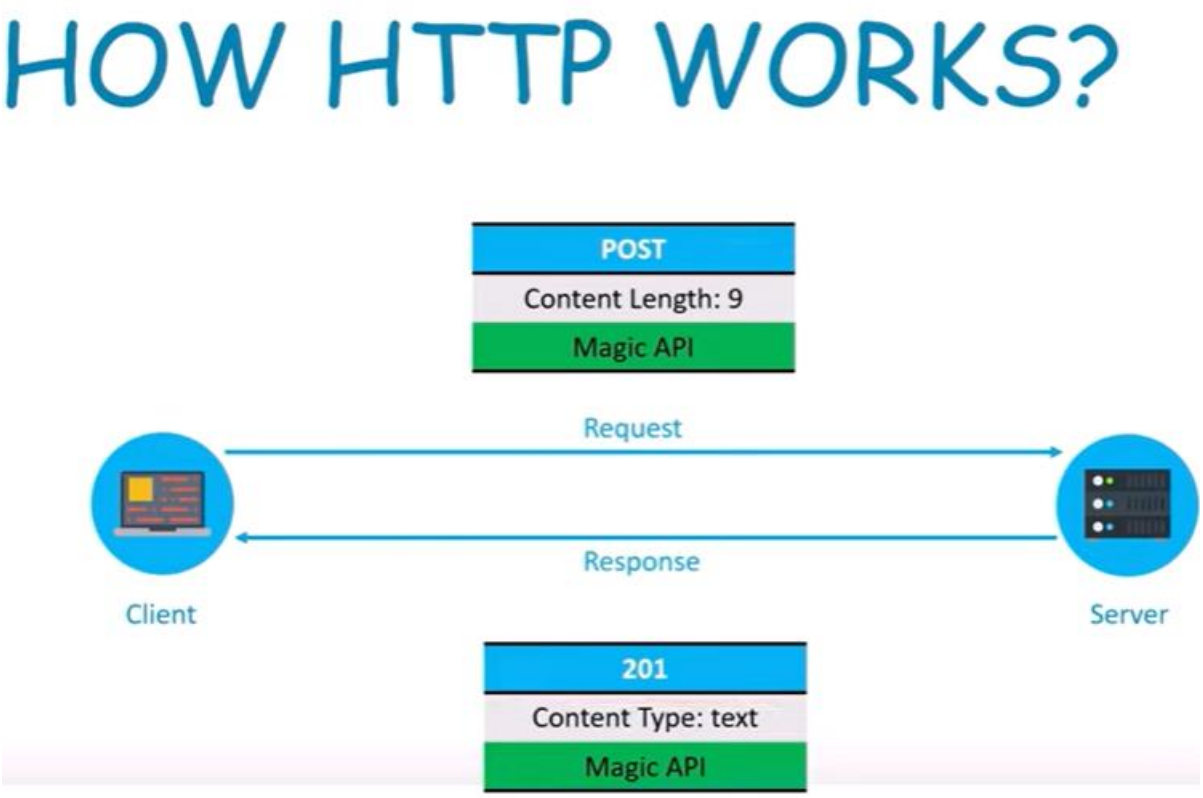
RESTFUL Web API

From - [RESTful API with .NET Core (.NET 7) - Full Course for Beginners (youtube.com)]

Way to transfer data

Way for application to communicate with each other



Stateless – will not remember the request

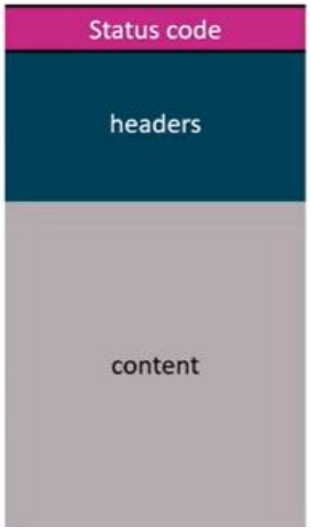
Request's Metadata

- **Content Type** : Content's Format
- **Content Length** : Size of the Content
- **Authorization** : Who is making the request
- **Accept** : What are the accepted type(s)
- More headers...

Request's Content

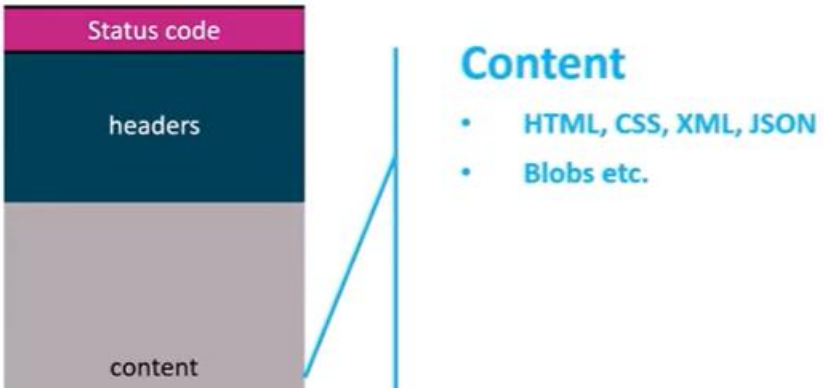
- **HTML, CSS, XML, JSON**
- **Information for the request.**
- **Blobs etc.**

THE RESPONSE OBJECT



Status Codes for Operation Result

- **100-199:** Informational
- **200-299:** Success
  - 200 – OK
  - 201 – Created
  - 204 – No Content
- **300-399:** Redirection
- **400-499:** Client Errors
  - 400 – Bad Request
  - 404 – Not Found
  - 409 – Conflict
- **500-599:** Server Errors
  - 500 – Internal Server Error



Folder Structure

Dependencies

Properties

Appsettings.json

Program.cs

Controller

A call with controller name and Derived from controllerBase

And has a attribute [ApiController] above the class

Models

Class file in Model folder

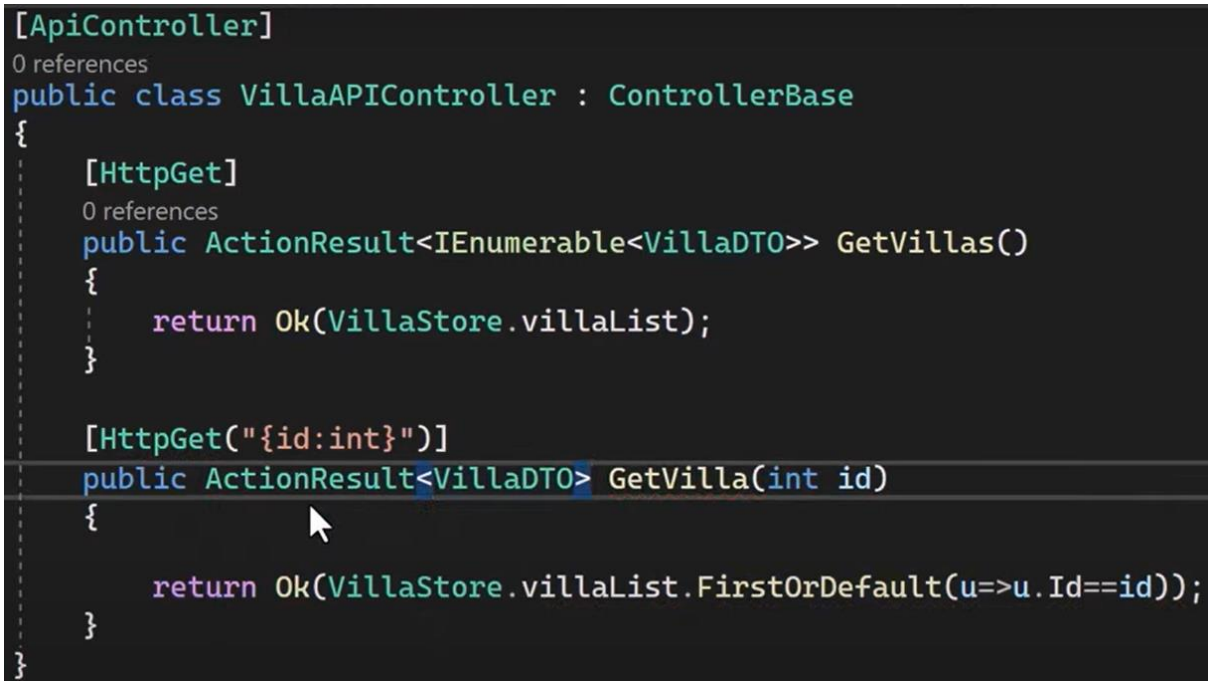
Has properties

Route

Above controller

[Route{“api/APIName”]

Or [Route{“api/[controller]”]



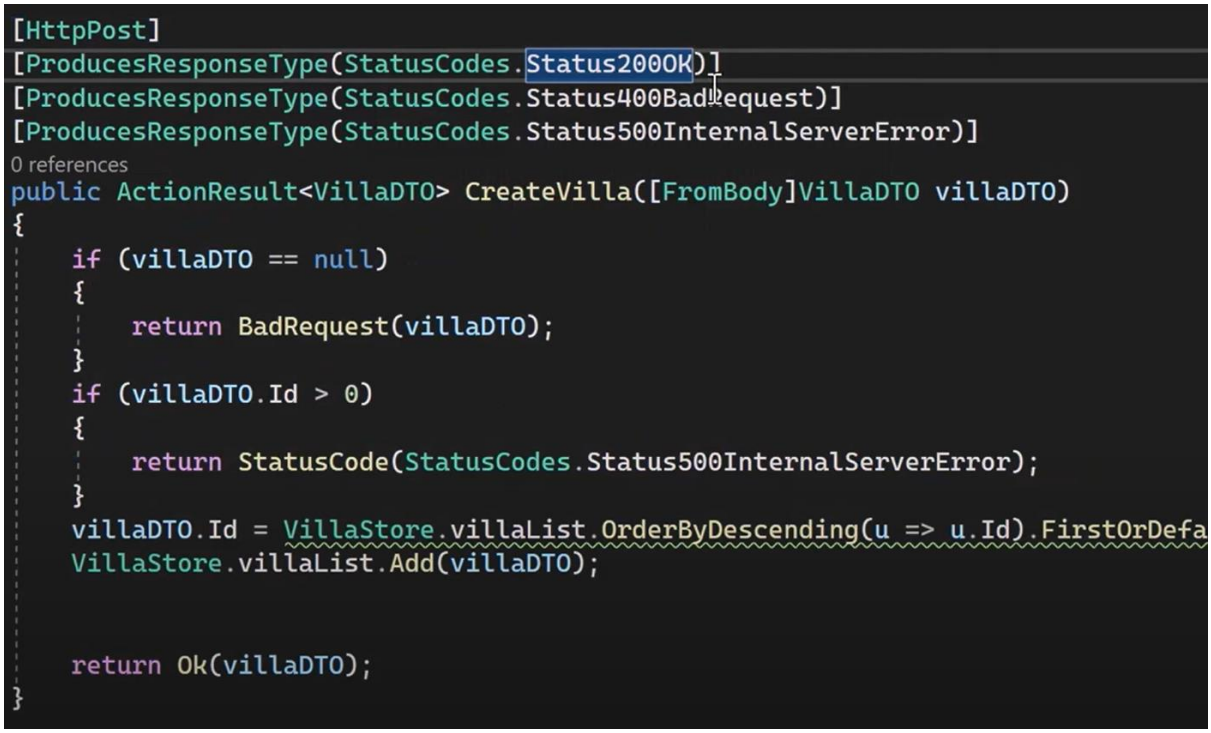
DTO – Data transfer Objects – what we need to expose to client. Like model has created date but DTO won’t

To **document** possible outcome of a request-

About action method

[ProducesResposeType(200, Type=typeof(Model))]

HttpPost



```
[HttpPost]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
public ActionResult<VillaDTO> CreateVilla([FromBody]VillaDTO villaDTO)
{
    if (villaDTO == null)
    {
        return BadRequest(villaDTO);
    }
    if (villaDTO.Id > 0)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }
    villaDTO.Id = VillaStore.villaList.OrderByDescending(u => u.Id).FirstOrDefa
    VillaStore.villaList.Add(villaDTO);


    return Ok(villaDTO);
}
```

CreatedAtRoute method will give 201

Data Annotation to validate the records or ModelState.Valid

Custom Validation error – ModelState.AddModelError(Name, Message)

Return badReq(ModelRequest)

HttpDelete

```
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[HttpDelete("{id:int}", Name = "DeleteVilla")]
0 references
public IActionResult DeleteVilla(int id)
{
    if (id == 0)
    {
        return BadRequest();
    }
    var villa = VillaStore.villaList.FirstOrDefault(u => u.Id == id);
    if (villa == null)
    {
        return NotFound();
    }
    VillaStore.villaList.Remove(villa);
    return NoContent();
```

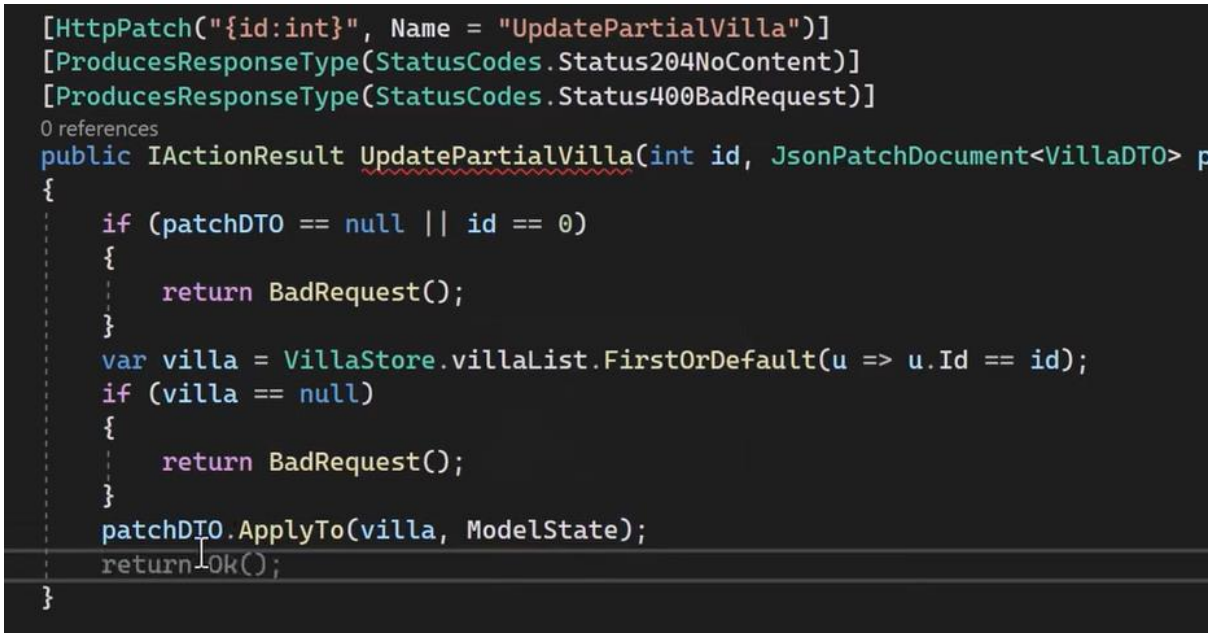
HttpPut

```
[HttpPut("{id:int}", Name = "UpdateVilla")]
0 references
public IActionResult UpdateVilla(int id, [FromBody]VillaDTO villaDTO)
{
    if (villaDTO == null || id != villaDTO.Id)
    {
        return BadRequest();
    }
    var villa = VillaStore.villaList.FirstOrDefault(u => u.Id == id);
    villa.Name = villaDTO.Name;
    villa.Sqft = villaDTO.Sqft;
    villa.Occupancy = villaDTO.Occupancy;

    return NoContent();
```

HttpPatch

Package – JSONPatch, NewtonsoftJSON



```
[HttpPatch("{id:int}", Name = "UpdatePartialVilla")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
0 references
public IActionResult UpdatePartialVilla(int id, JsonPatchDocument<VillaDTO> p
{
    if (patchDTO == null || id == 0)
    {
        return BadRequest();
    }
    var villa = VillaStore.villaList.FirstOrDefault(u => u.Id == id);
    if (villa == null)
    {
        return BadRequest();
    }
    patchDTO.ApplyTo(villa, ModelState);
    return Ok();
}
```

Dependency Injection

Constructor injection

Create a class and use it in constructor

And in program.cs

Builder.services.addScoped<Ilogging, logging>();

addSingleton – longest lifetime. created when app start and will be used everytime application request implementation

addScoped -for every request

add transient – every time object is accessed. Within a single request accessed multiple times

Entity Framework Core – Code First approach

LINQ will be changed to SQL queries

Create a model class

```
[Key]
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
0 references
public int Id { get; set; }
0 references
public string Name { get; set; }
0 references
public string Details { get; set; }
```

Add EFCore, EFcore.Tools sqlserver packages

Create data folder and create a DBContext class which inherits from DBContext and create DBsets

```
namespace MagicVilla_VillaAPI.Data
{
    3 references
    public class ApplicationDbContext : DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        0 references
        public DbSet<Villa> Villas { get; set; }
    }
}
```

Add connection string in appsettings.JSON

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultSQLConnection": "Server=.;Database=Magic_VillaAPI;Trusted_Connection=True;MultipleActiveRe
  }
}
```

Register DB Context class and provide connection string

```
builder.Services.AddDbContext<ApplicationDbContext>(option => {
    option.UseSqlServer(builder.Configuration.GetConnectionString("DefaultSQLConnection"));
});
```

In PM Console –

add-migration name

Update database

Seeding data in DBContext

```
0 references
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Villa>().HasData(
        new Villa()
        {
            Id=1,
            Name="Royal Villa",
            Details= "Fusce 11 tincidunt maximus leo, sed scelerisque massa auctor sit amet. Donec e
            ImageUrl= "https://dotnetmasteryimages.blob.core.windows.net/bluevillaimages/villa3.jpg"
            Occupancy=5,
            Rate=200,
            Sqft=550,
            Amenity=""
        }
        )
}
```

To use DB Context in a class

Create aprivate readonly variable _var

In Ctor DI _db=db;

_db.DBSet.Remove(), .Update(), FirstOrDefault()

_ db.Savechanges();

Not to track a record –

_db.DbSet.AsNoTracking()

Rate Limiting – from .net 7

```
builder.Services.AddRateLimiter(rateLimiterOptions =>
{
    rateLimiterOptions.AddFixedWindowLimiter("fixed", options =>
    {
        options.PermitLimit = 1;
        options.Window = TimeSpan.FromSeconds(5);
        options.QueueLimit = 0;
    });
    rateLimiterOptions.RejectionStatusCode = StatusCodes.Status429TooManyRequests;
});
```

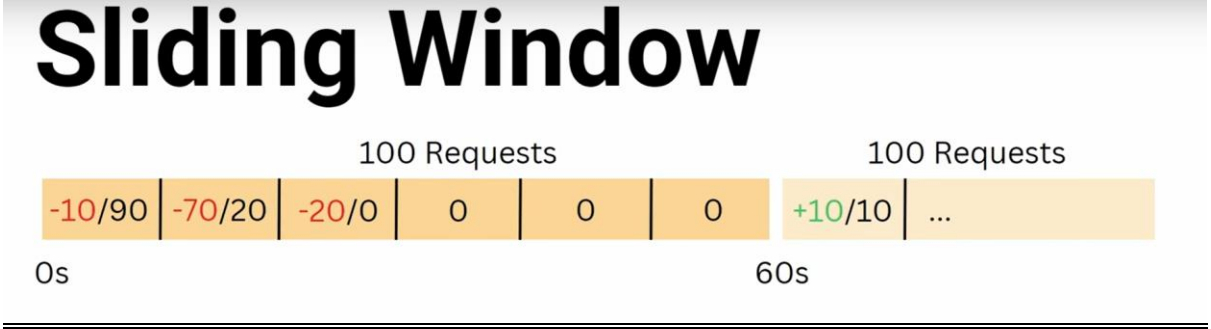
```
app.UseRateLimiter();
```

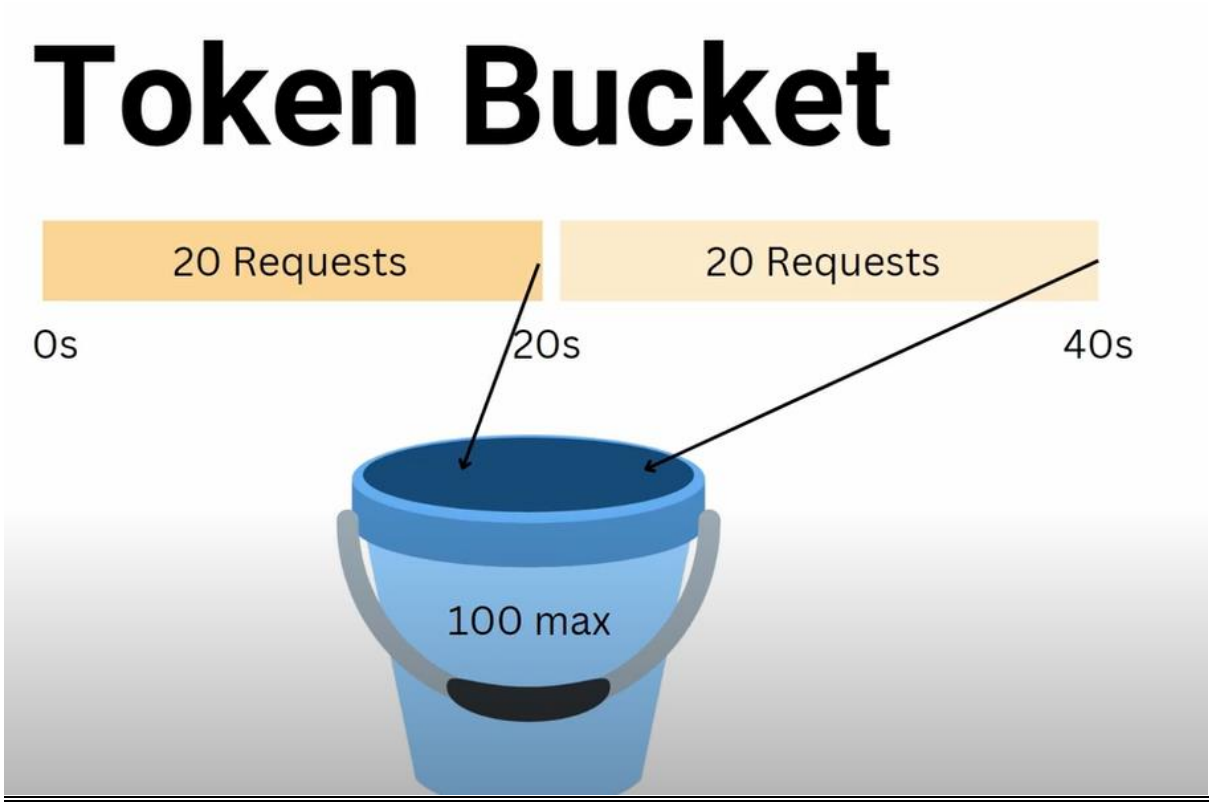
In minimal API –

```
app.MapGet("/weatherforecast", () =>
{
    var forecast = Enumerable.Range(1, 5).Select(index =>
        new WeatherForecast
        (
            DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            Random.Shared.Next(-20, 55),
            summaries[Random.Shared.Next(summaries.Length)]
        ))
        .ToArray();
    return forecast;
})
.RequireRateLimiting("fixed")
```

In controller API –

```
[ApiController]
[Route("[controller]")]
[EnableRateLimiting("fixed")]
3 references
public class WeatherForecastController : ControllerBase
```





API versioning –

Add asp versioning http

```
builder.Services.AddApiVersioning();

var app = builder.Build();

var versionSet = app.NewApiVersionSet()
    .HasApiVersion(1)
    .HasApiVersion(2)
    .Build();

app.MapGet("hello", () => "Hello world")
    .WithApiVersionSet(versionSet)
    .MapToApiVersion(1);

app.MapGet("hello", () => "Hello world v2")
    .WithApiVersionSet(versionSet)
    .MapToApiVersion(2);

app.Run();
```