

# Workshop





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาณกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc

@somkiat.cc

Home Posts Videos Photos

Liked Following Share ...

+ Add a Button



# ASP.NET



# **ASP.NET**

# **ASP.NET Core**



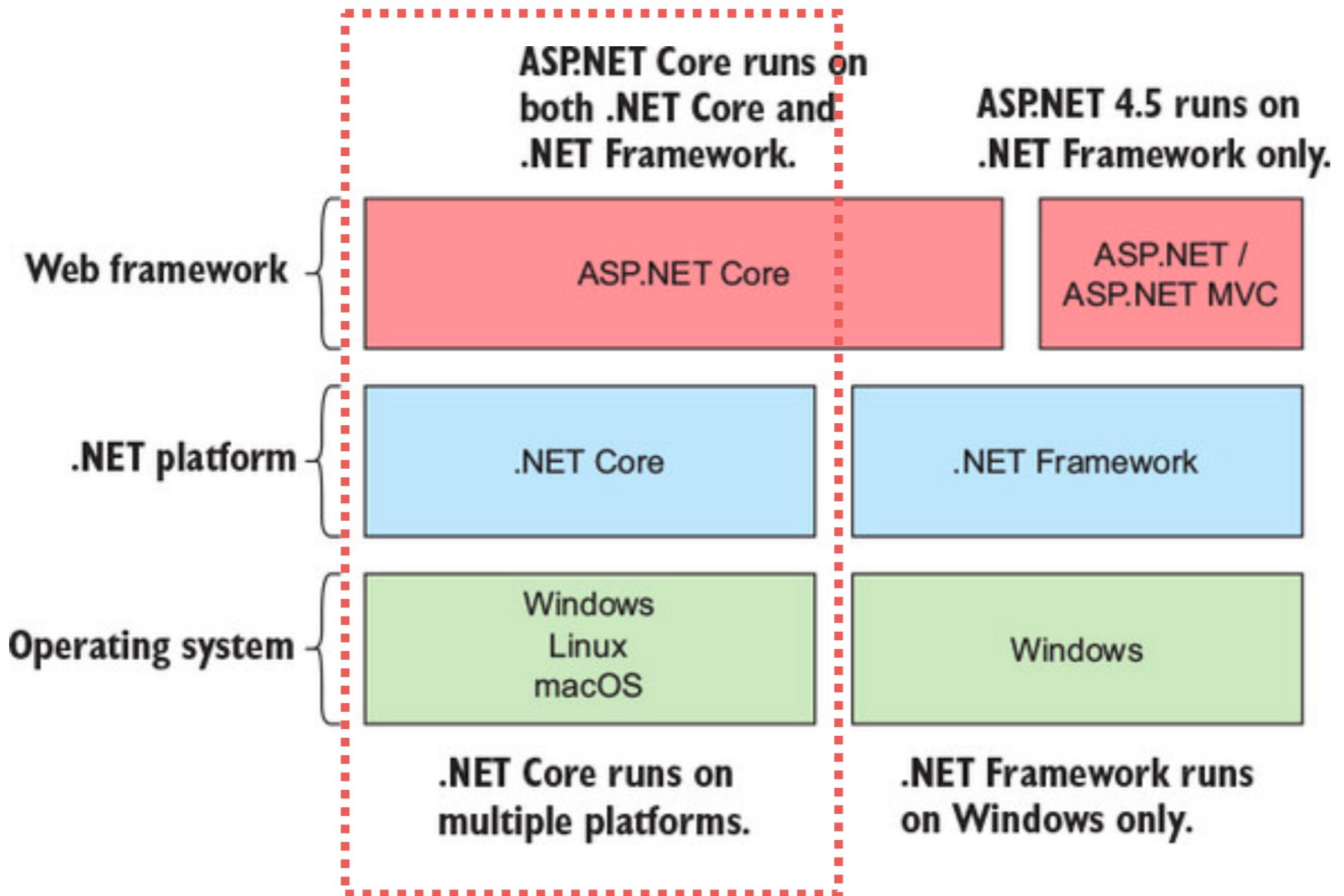
# .NET Core



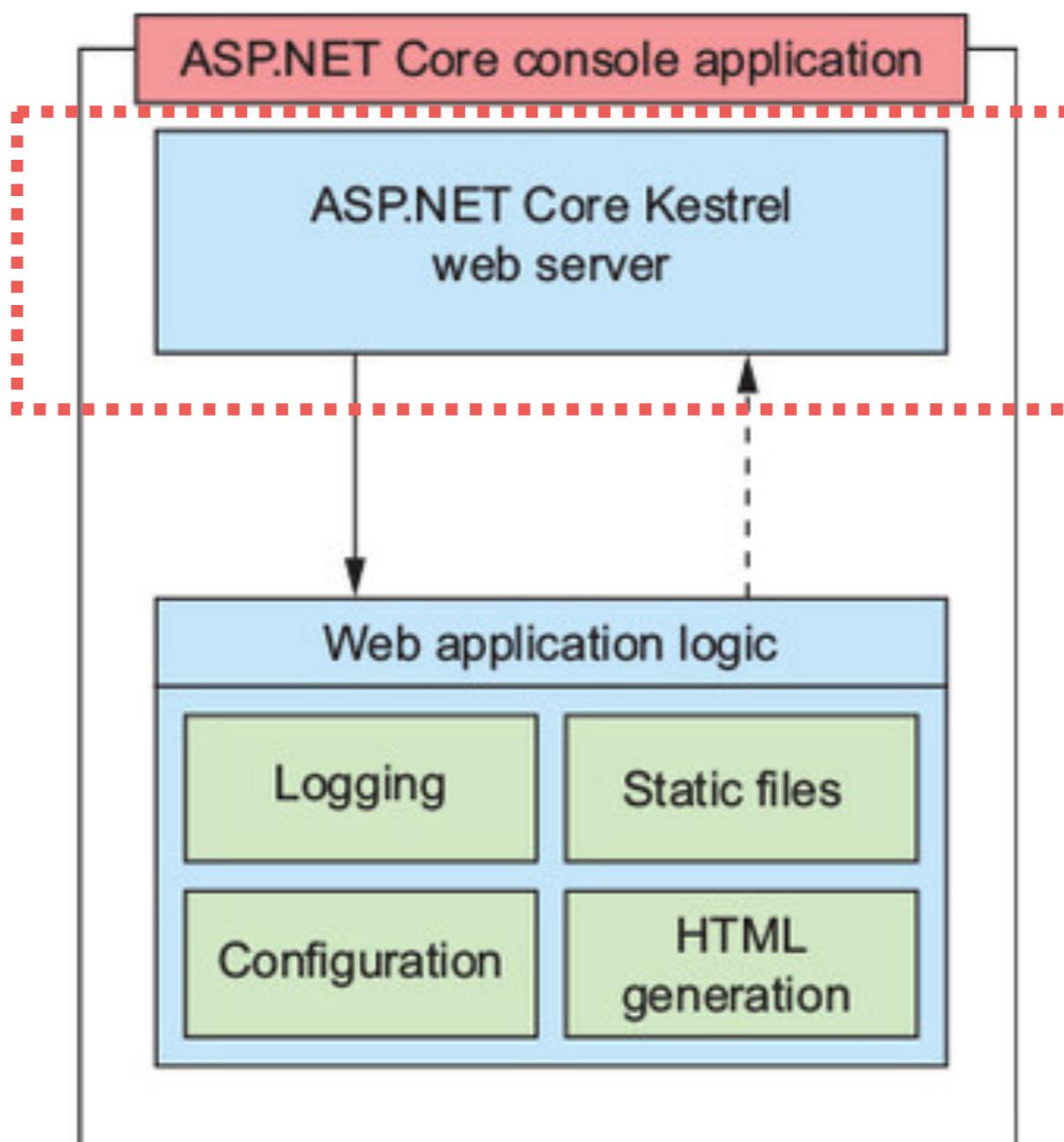
# **.NET 5 (next)**



# .Net Core



# Provide cross-platform web server



# .Net Core project types

Console

Worker (background task)

Web application

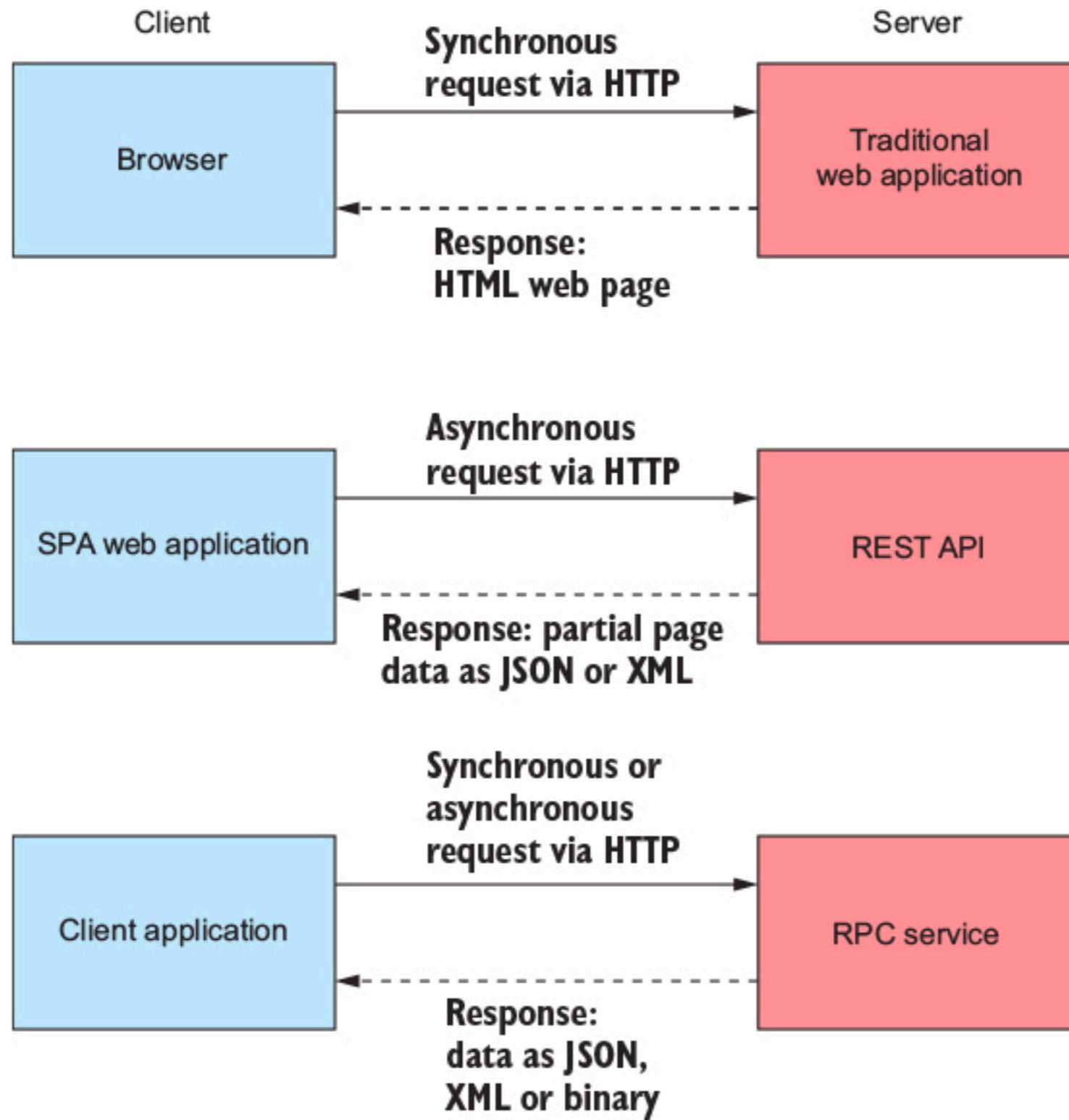
Web + MVC

Web API (RESTful)

etc

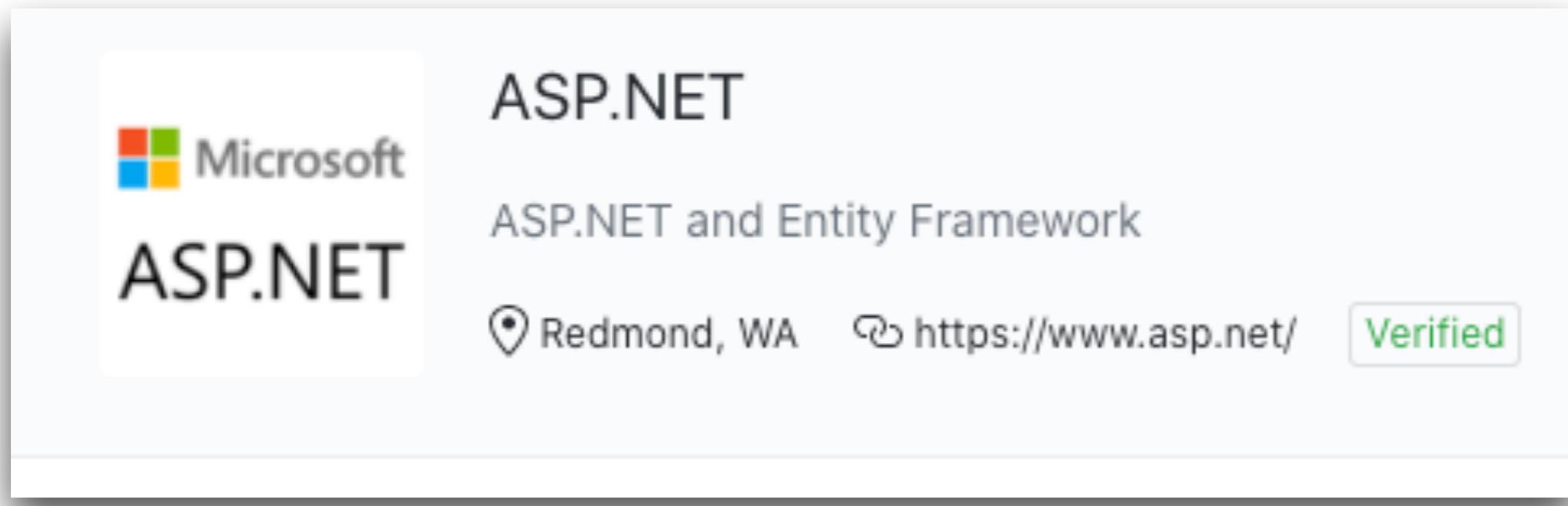


# .Net Core project types



# Library of .Net core

<https://github.com/aspnet>



# Core

# Kestrel web server

# Logging

# Authentication

# Authorization



# Who use .Net Core ?

The screenshot shows a Stack Overflow question titled "Using the antiforgery cookie in ASP.NET Core but with a non-default CookieName". The question discusses changing the default antiforgery cookie name to anonymize it. It has 1 answer and 55 views. The user profile of the asker is shown, along with a sidebar for upcoming events and job ads.

**User notifications**

**Currently logged in user**

**Questions submitted by users**

**User votes update scores on the server**

**Answers submitted by users**

**Viewing statistics**

**Events and jobs based on location and user profile**

Using the antiforgery cookie in ASP.NET Core but with a non-default CookieName

I'm thinking about changing name of the default antiforgery cookie in ASP.NET Core.  
The reason why I would like to change the cookie name is to anonymize the cookie, in my opinion there is no reason why end users should be able to determine the responsibility of this cookie.  
Microsoft.AspNetCore.Antiforgery.AntiforgeryOptions.CookieName  
1. How do I change the name of the antiforgery cookie? I guess it should be done in the Startup.cs file in somehow?  
2. What possible implications could occur by changing name the default antiforgery cookie?  
3. How do I use the antiforgery cookie in ASP.NET Core?

asked 4 hours ago  
Jones Avelison 374 ● 11 ● 26

share edit flag  
add a comment

1 Answer

You can set a different name in your `Startup.ConfigureServices`.as in:  
`services.AddAntiforgery(options => options.CookieName = "MyAntiforgeryCookie")`

Want a job?  
Applications Developer  
Whiteworth Brook Ltd. ● 9.1km from user, UK  
£25,000 - £30,000  
[View Job](#)

<https://stackoverflow.com/>



# Provide command-line

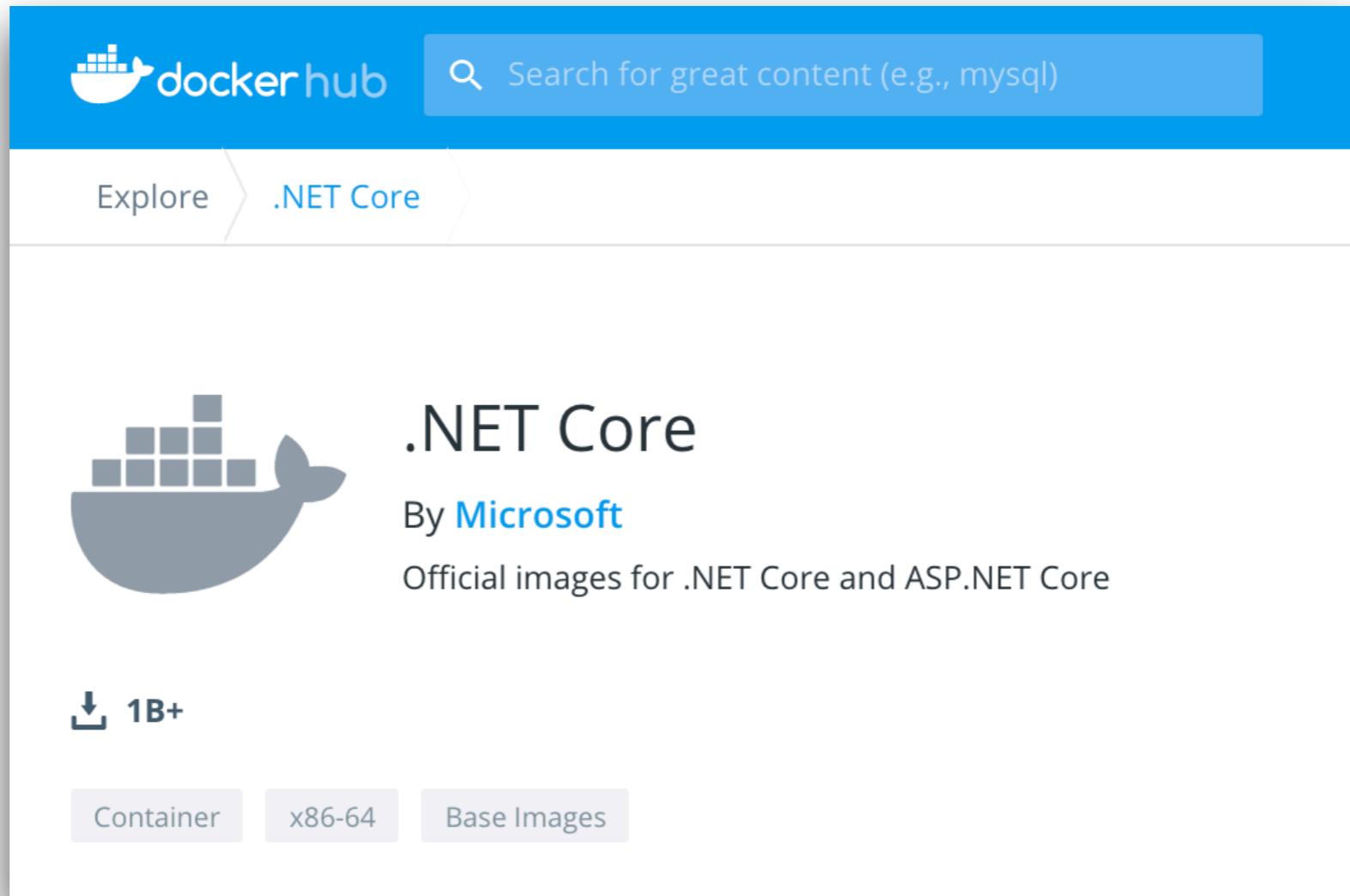
## \$dotnet

```
1 Usage: dotnet [options]
2 Usage: dotnet [path-to-application]
3
4 Options:
5   -h|--help           Display help.
6   --info              Display .NET Core information.
7   --list-sdks         Display the installed SDKs.
8   --list-runtimes    Display the installed runtimes.
9
10 path-to-application:
11   The path to an application .dll file to execute.
```

<https://docs.microsoft.com/en-us/dotnet/core/tools/>



# Build and deploy with containers



[https://hub.docker.com/\\_/microsoft-dotnet-core](https://hub.docker.com/_/microsoft-dotnet-core)



# **Benefits of .Net Core**

Cross platform development and deploy

Open source

Focus on performance as a feature

Simplified hosting model

Short release cycle

Modular feature



# Release schedule

## .NET Schedule

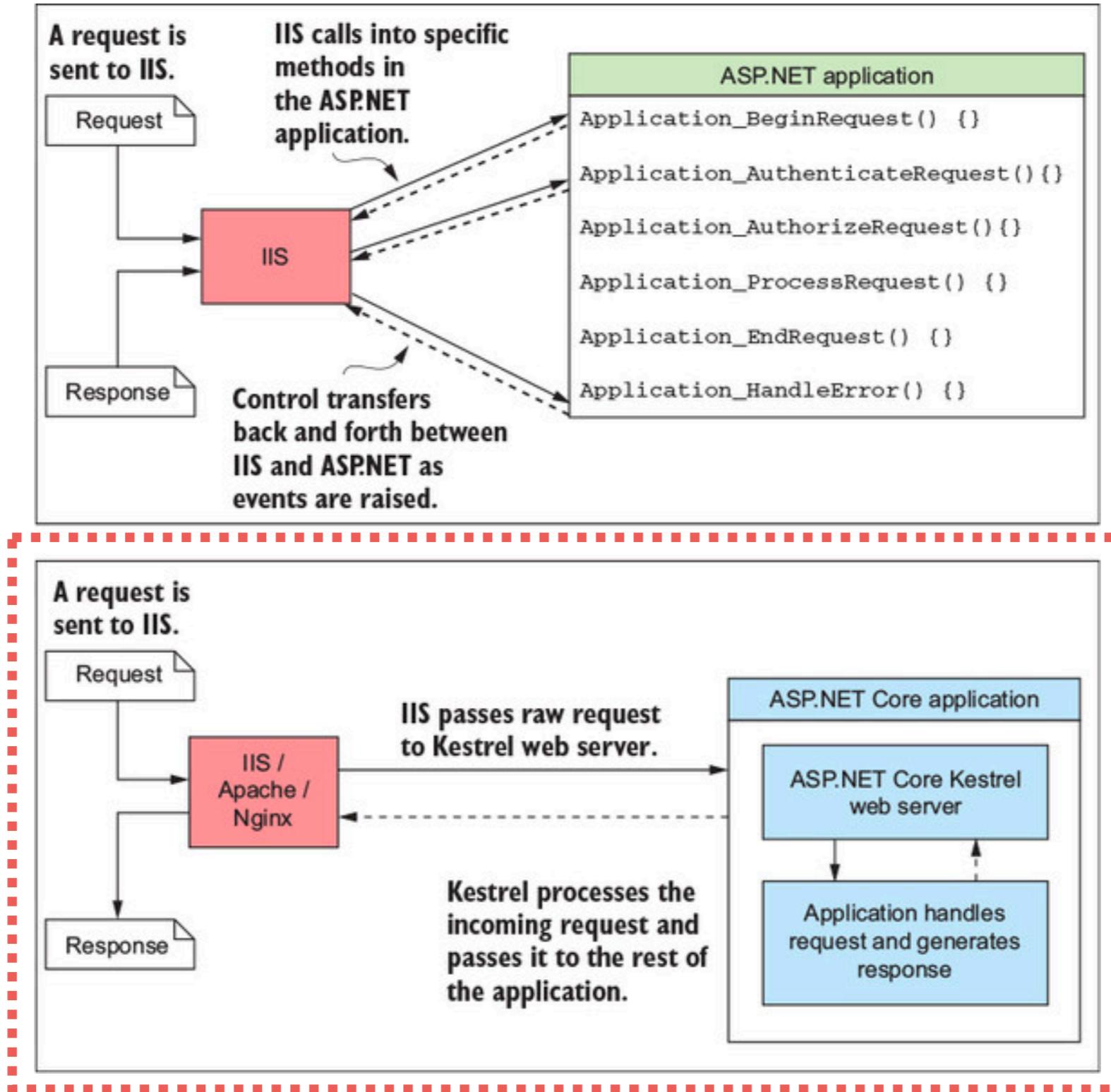


- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

<https://dotnet.microsoft.com/platform/support/policy/dotnet-core>



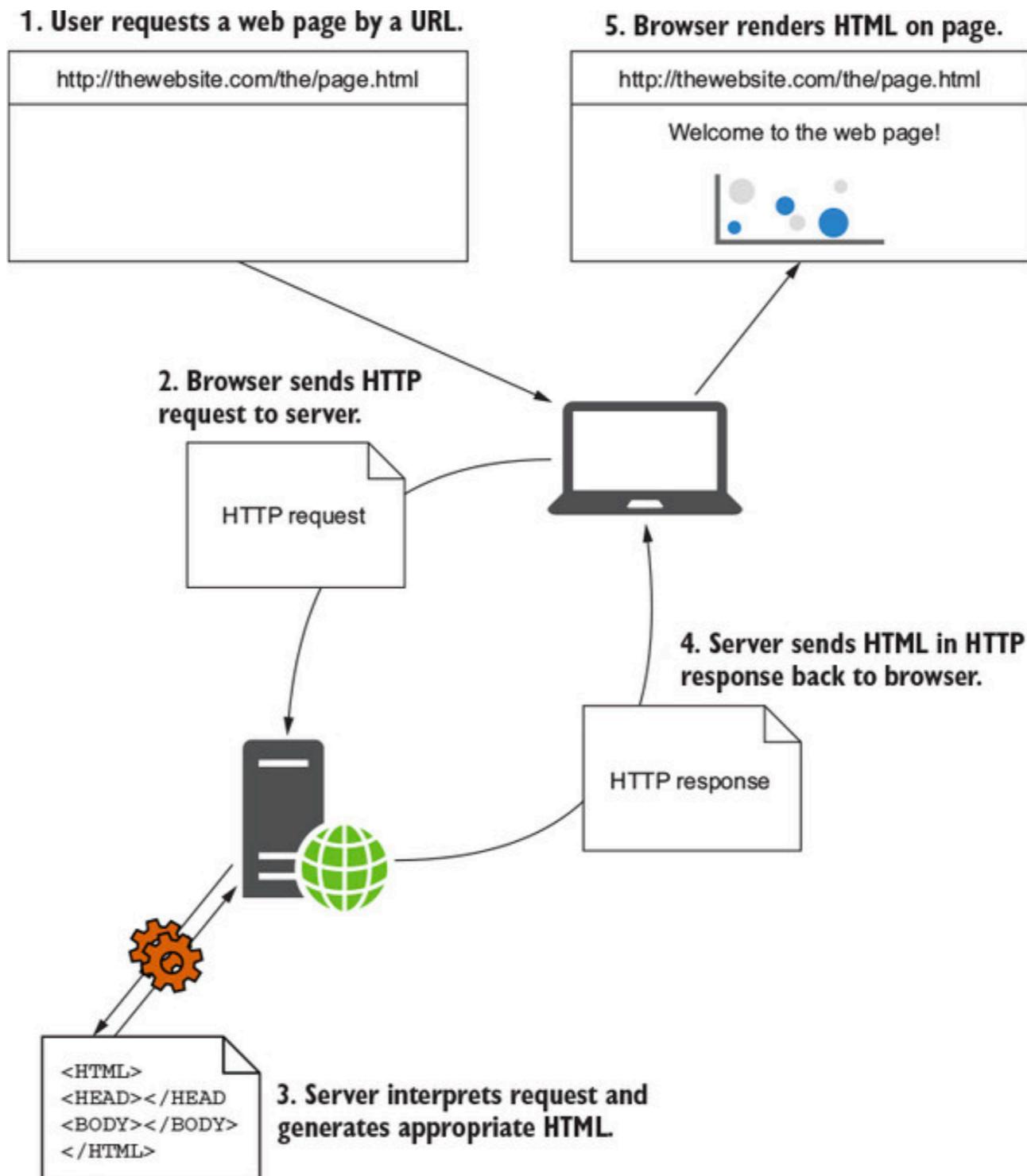
# Deployment (IIS is optional)



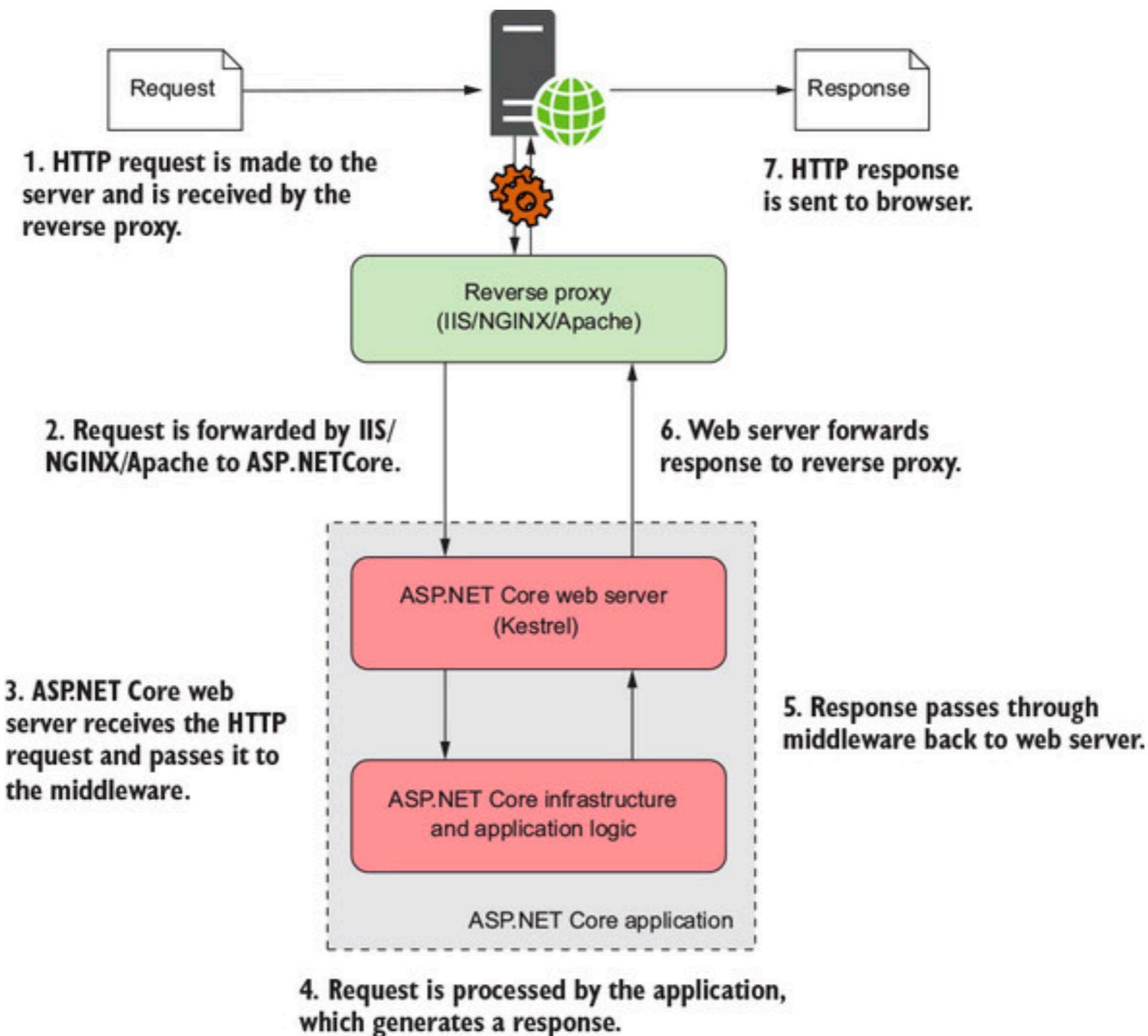
# Develop web MVC project



# Web :: request and response



# .Net Core process



# Topics

Create web project  
Running application  
Understanding the component of application  
Using command-line tools



# Create web application

## 1. Generate

Create the application from template

## 2. Restore

Restore all packages and dependencies with NuGet

## 3. Build

Compile the application and generate all the assets

## 4. Run

Run the compiled application



# Command-line tools

\$dotnet new

\$dotnet restore

\$dotnet build

\$dotnet run



# Create web project

Choose .Net Core => Web application

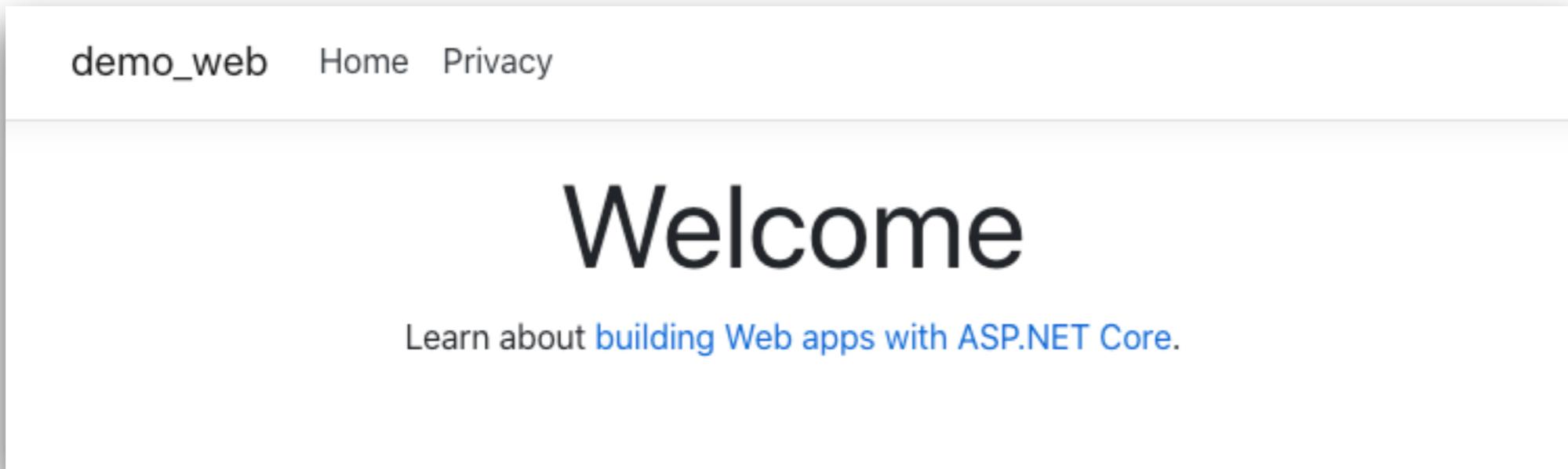
With No-authentication

Uncheck Enable Docker support (if not need)



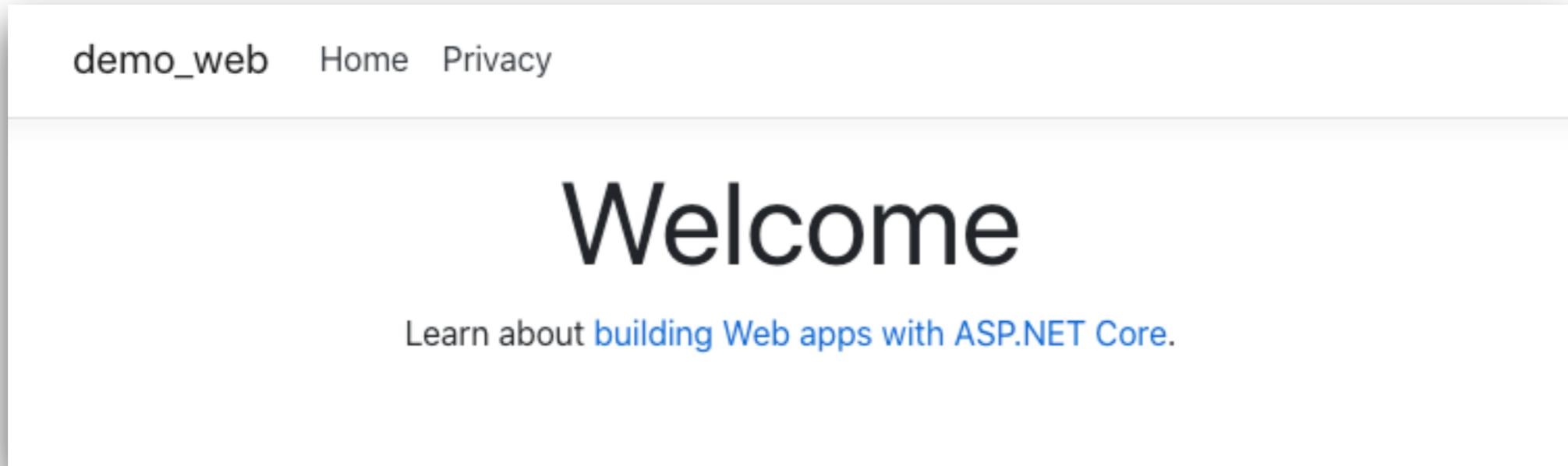
# Run application

<https://localhost:5001/>



# Run in command-line

```
$dotnet run -p <project file.csproj>
```



# Problem

Certificate problem when run project !!!

\$dotnet dev-certs https



# Application flow

Program.cs



# Program.cs

```
public class Program          Start point of application
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[]
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
    }
}
```



# Program.cs

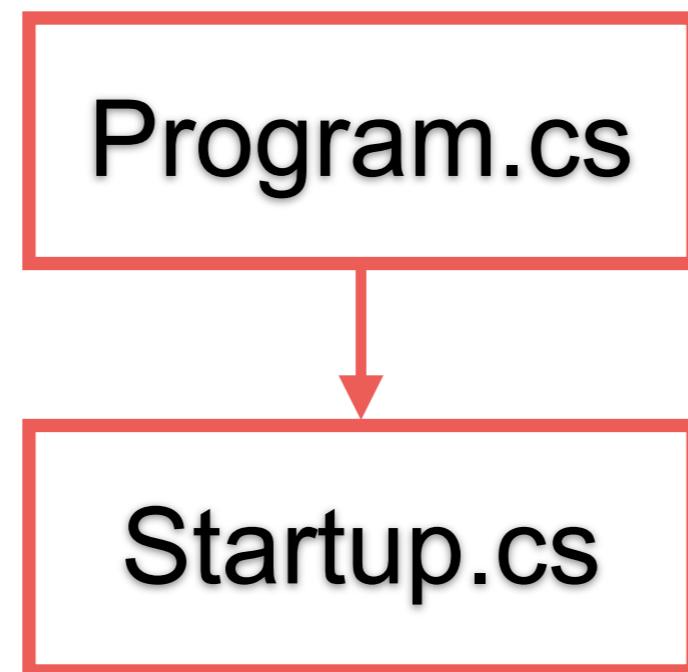
```
public class Program
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[])
    {
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
    }
}
```

Contain host config and Kestrel server



# Application flow



# Startup.cs

Config app's services

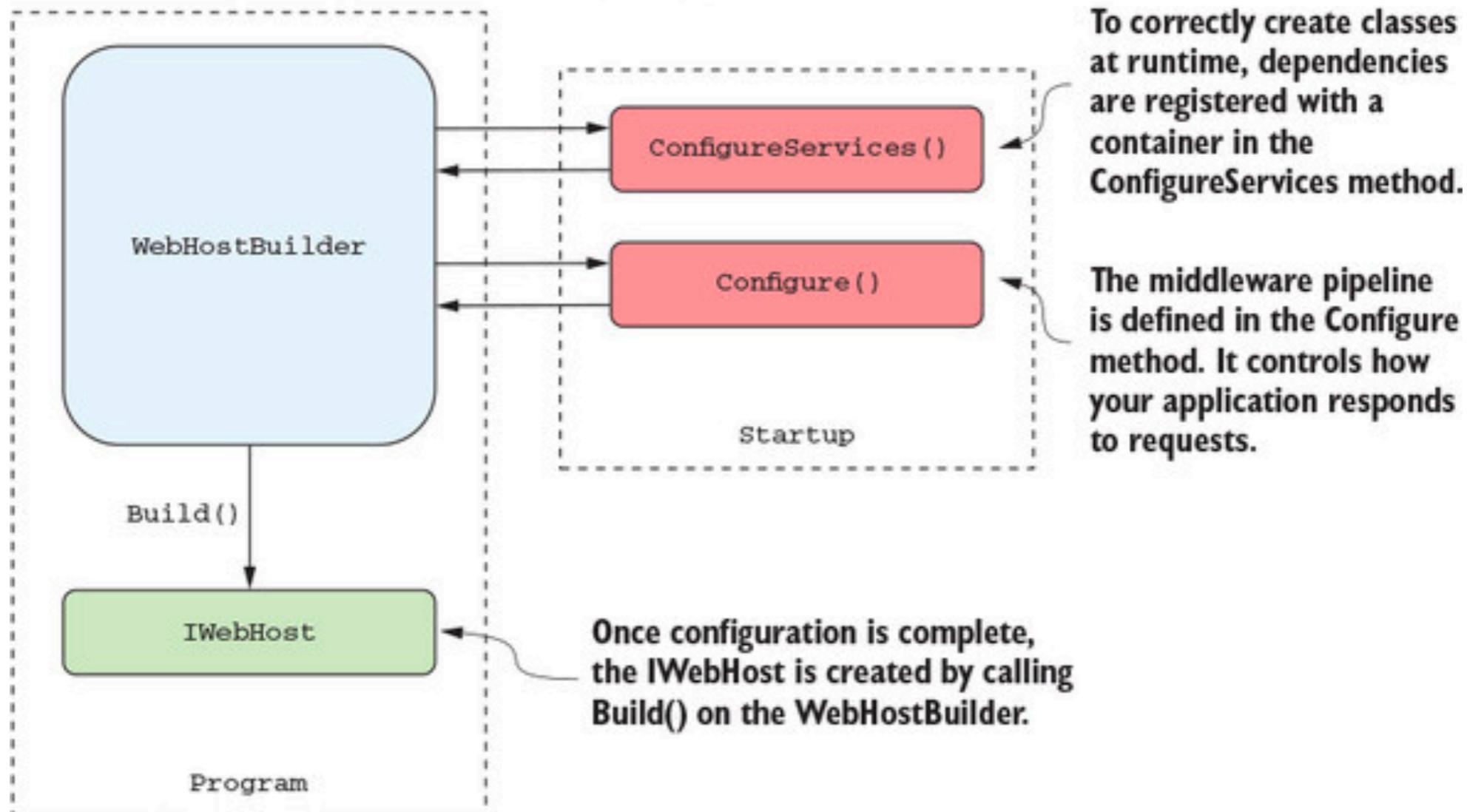
Define middleware pipeline



# Process flow

The `IWebHost` is created in `Program` using the `Builder` pattern, and the `CreateDefaultBuilder` helper method.

The `WebHostBuilder` calls out to `Startup` to configure your application.



# **Program.cs vs Startup.cs**

## **Program.cs**

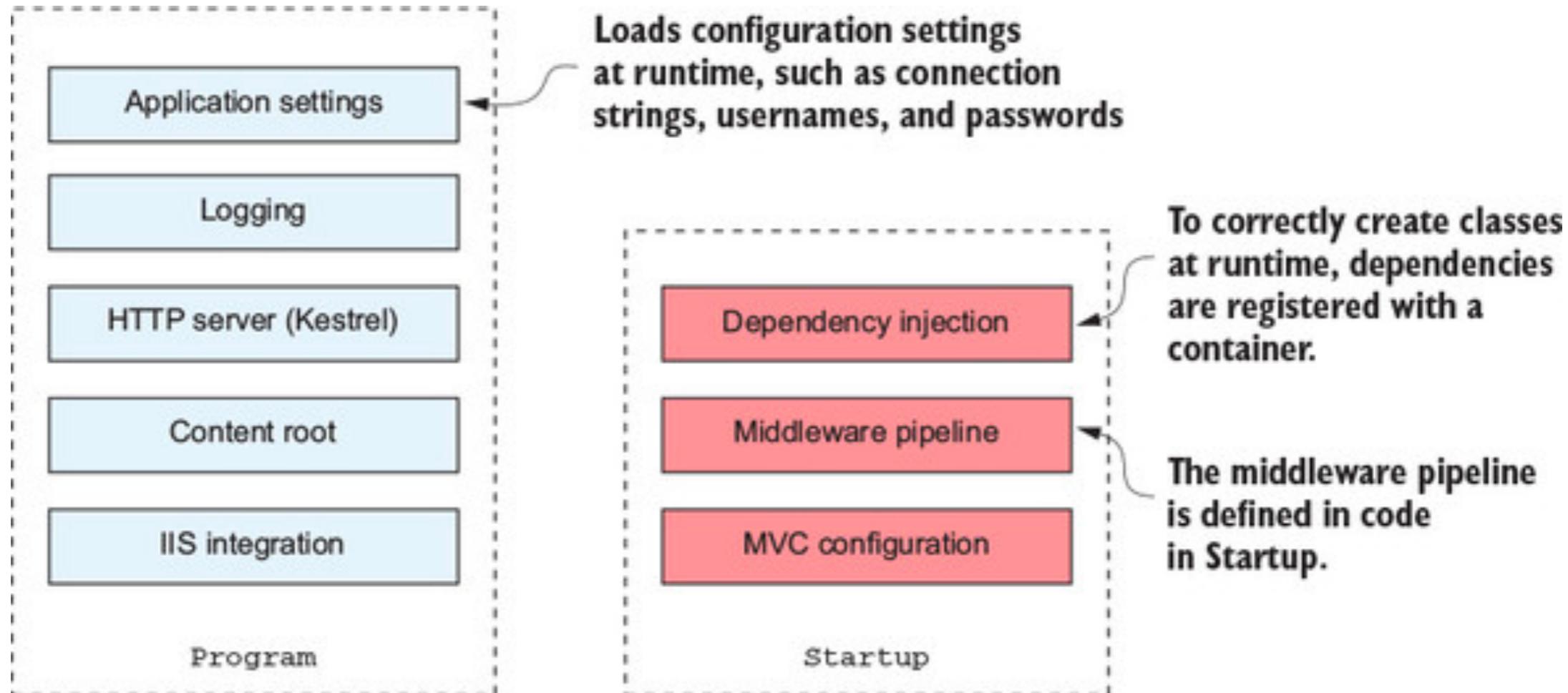
Config infrastructure of app such as HTTP server  
Integration with IIS and configuration sources

## **Startup.cs**

Define components and features your app uses and  
middleware pipeline



# Program.cs vs Startup.cs



**Program.cs** is used to configure infrastructure that rarely changes over the lifetime of a project.

**Startup** is used to configure the majority of your application's custom behavior.



# Startup.cs

Load app's configuration

Read from file **appsettings.json**

Read from **environment variables**

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration>



# Config for multiple env

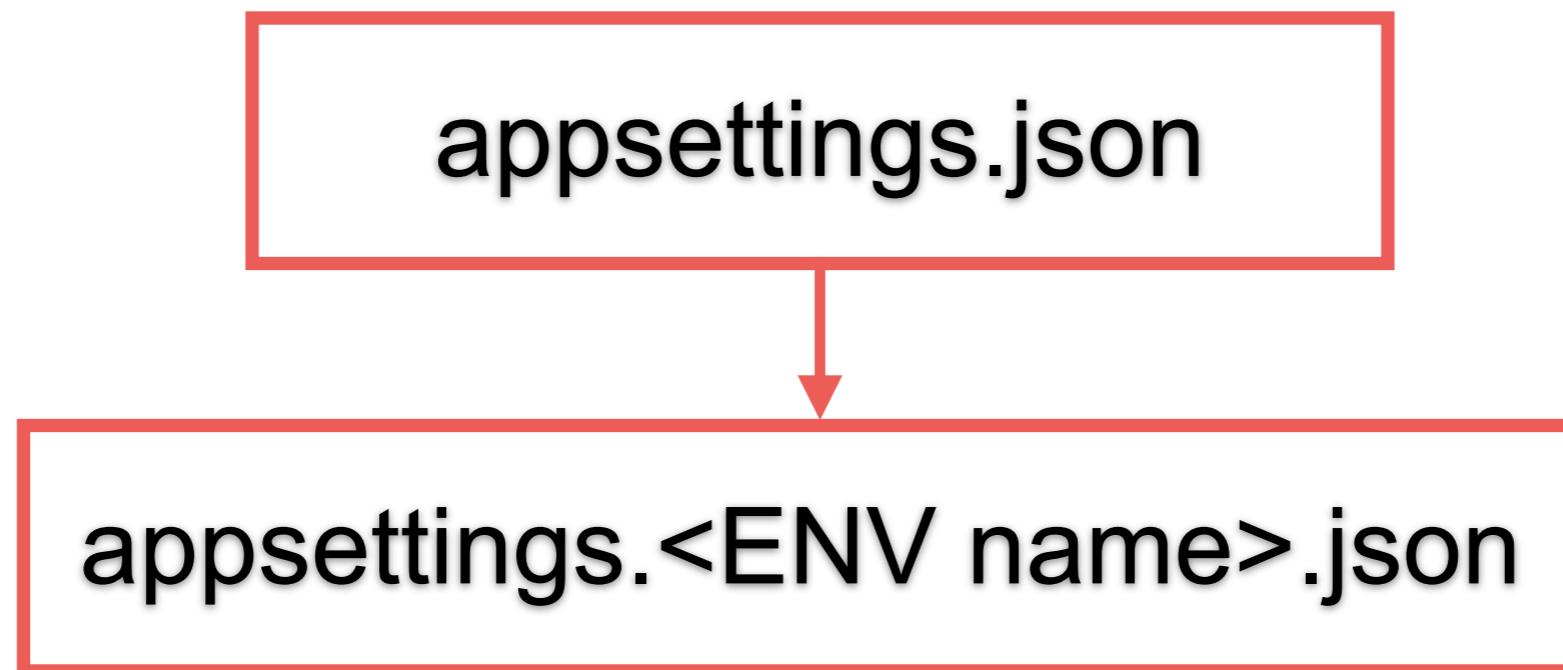
appsettings.<ENV name>.json

appsettings.Production.json

appsettings.Development.json



# Step to read config file



# Read config in Startup.cs

```
{  
  "Person": {  
    "Name": "Somkiat",  
    "Title": "Developer"  
  },  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft": "Warning",  
      "Microsoft.Hosting.Lifetime": "Information"  
    }  
  }  
}
```

appsettings.Development.json

Person.cs

```
public class Person  
{  
  public string Name { get; set; }  
  public string Title { get; set; }  
}
```



# Read config in Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    // Load configuration
    var person = Configuration.GetSection("Person").Get<Person>();
    Console.WriteLine(person.Name + "\n");
    Console.WriteLine(person.Title + "\n");

    services.AddControllers();
}
```



# Startup.cs

## Config app's services

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```

You can add the new service



# 1. Create TestService.cs

```
public class TestService
{
    private readonly ILogger<TestService> logger;

    public TestService(ILogger<TestService> logger)
    {
        this.logger = logger;
    }

    public void Test()
    {
        logger.LogWarning("I'm writing a message");
    }
}
```



## 2. Register service in Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddScoped<TestService>();
}
```



# 3. Use in controller class

```
public class HomeController : Controller
```

Dependency Injection

```
{  
    private readonly ILogger<HomeController> _logger;  
  
    private readonly TestService service;  
    private readonly ILogger<HomeController> logger;  
    public HomeController(TestService service, ILogger<HomeController> logger)  
    {  
        this.logger = logger;  
        this.service = service;  
    }  
}
```

```
    public IActionResult Index()  
    {  
        service.Test();  
        return View();  
    }
```



# 3. Use in controller class

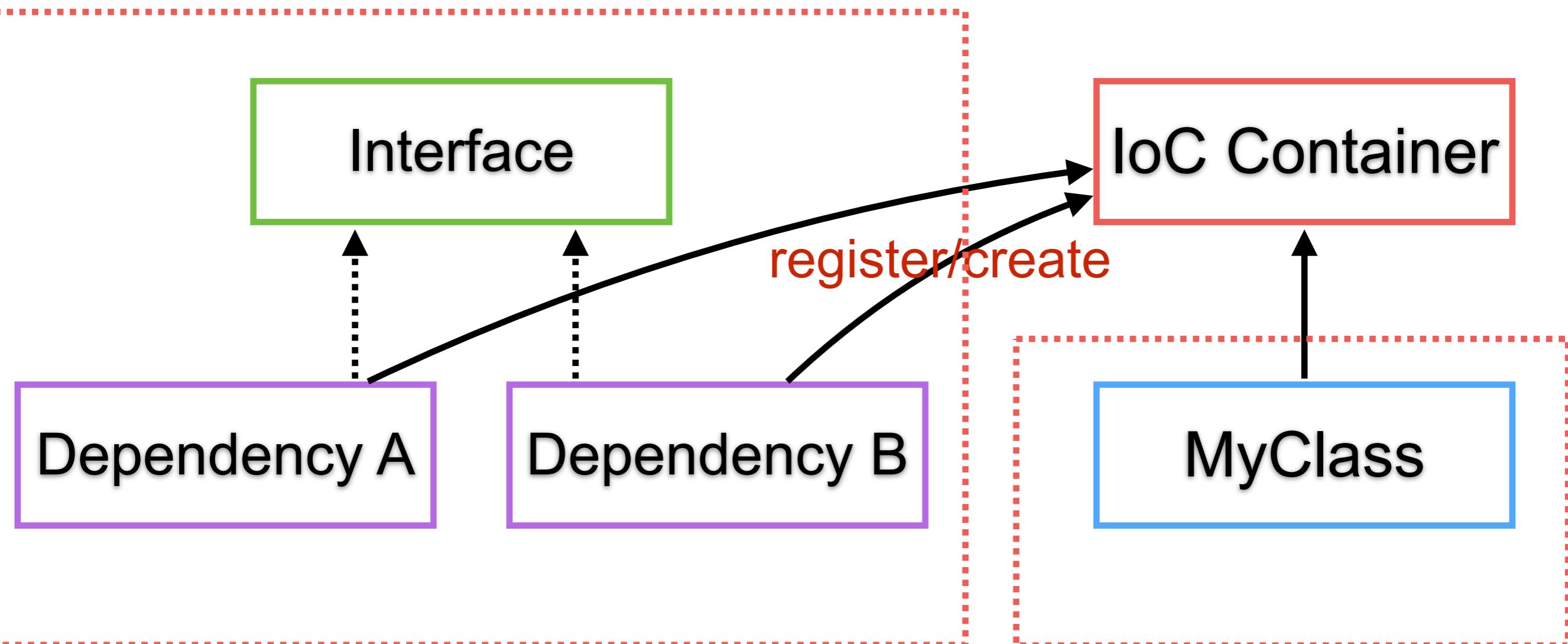
```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly TestService service;
    private readonly ILogger<HomeController> logger;
    public HomeController(TestService service, ILogger<HomeController> logger)
    {
        this.logger = logger;
        this.service = service;
    }

    public IActionResult Index()
    {
        service.Test();
        return View();
    }
}
```

Call service



# Inversion of Control



“Loose coupling”



# Startup.cs

## Define middleware pipeline

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this.
        app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```



# Startup.cs => Configure

**IApplicationBuilder**

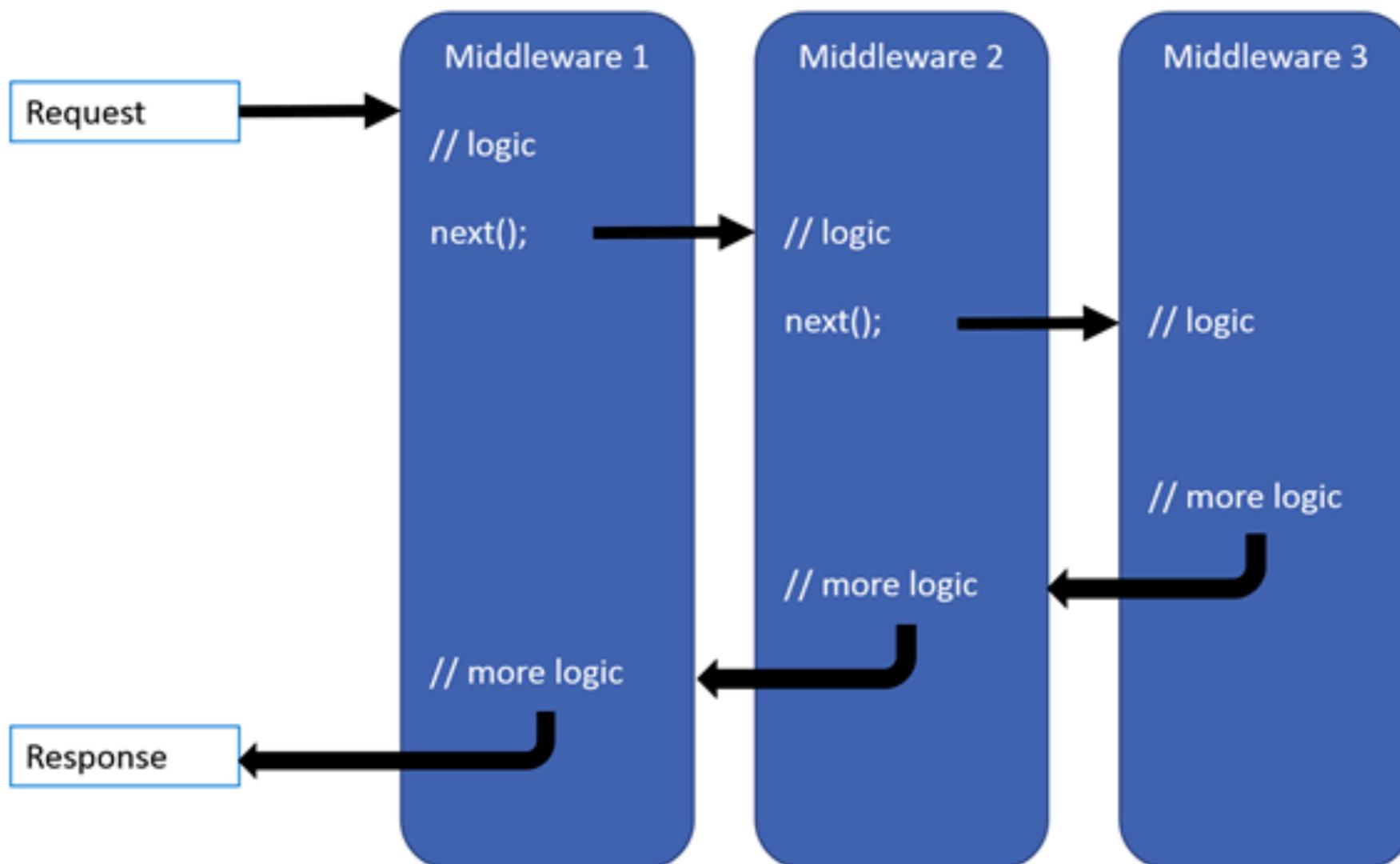
Define ordered  
of middleware

**IWebHostEnvironment**

EnvironmentName  
ContentRootPath  
WebRootPath



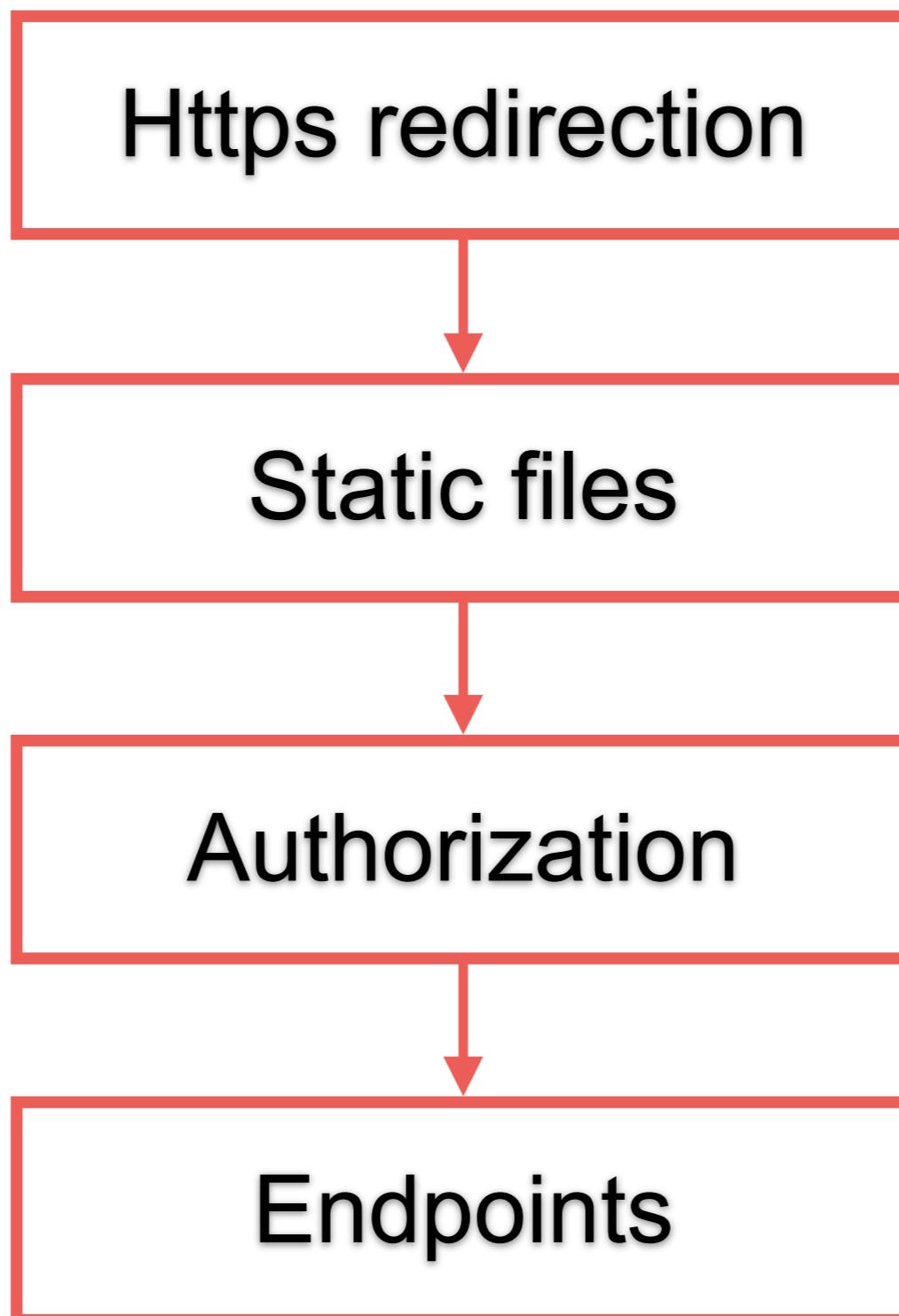
# Middleware



<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware>



# Middleware



<https://docs.microsoft.com/en-us/aspnet/core/security/enforcing-ssl>



# Static File Middleware

HTML, CSS, JavaScript, Image  
Default path in folder **wwwroot**  
Change with **UseWebRoot()**

```
public static IHostBuilder CreateHostBuilder(string args)
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
    {
        webBuilder.UseWebRoot("myroot");
        webBuilder.UseStartup<Startup>();
    });
}
```

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/static-files>



# Static File Middleware

Manage HTTP response header

Manage authorisation

Enable directory browsing

UseFileServer



# Routing middleware

To match the URLs of incoming requests and map to actions

Must to use **UseRouting** and **UseEndpoints**

```
app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing>



# Routing and Endpoints

```
app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

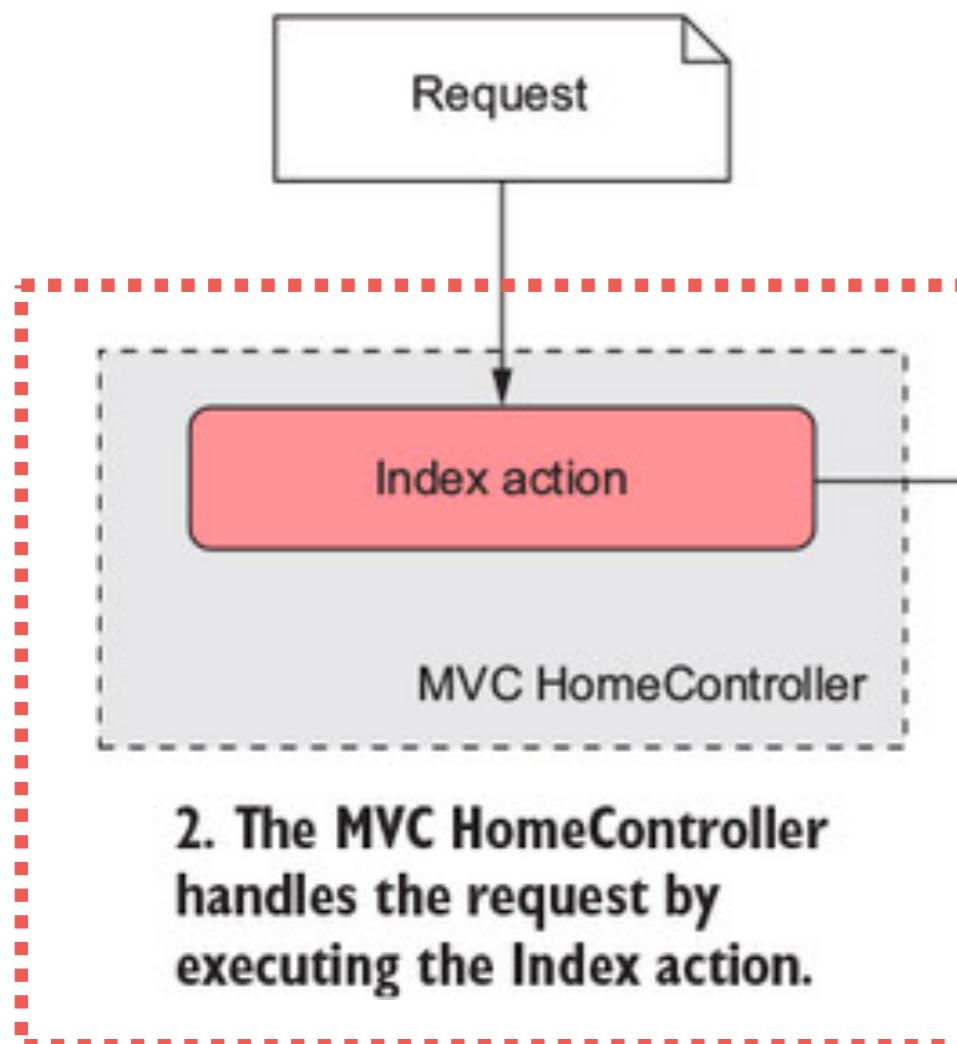
Controller default = Home  
Action default = Index

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing>

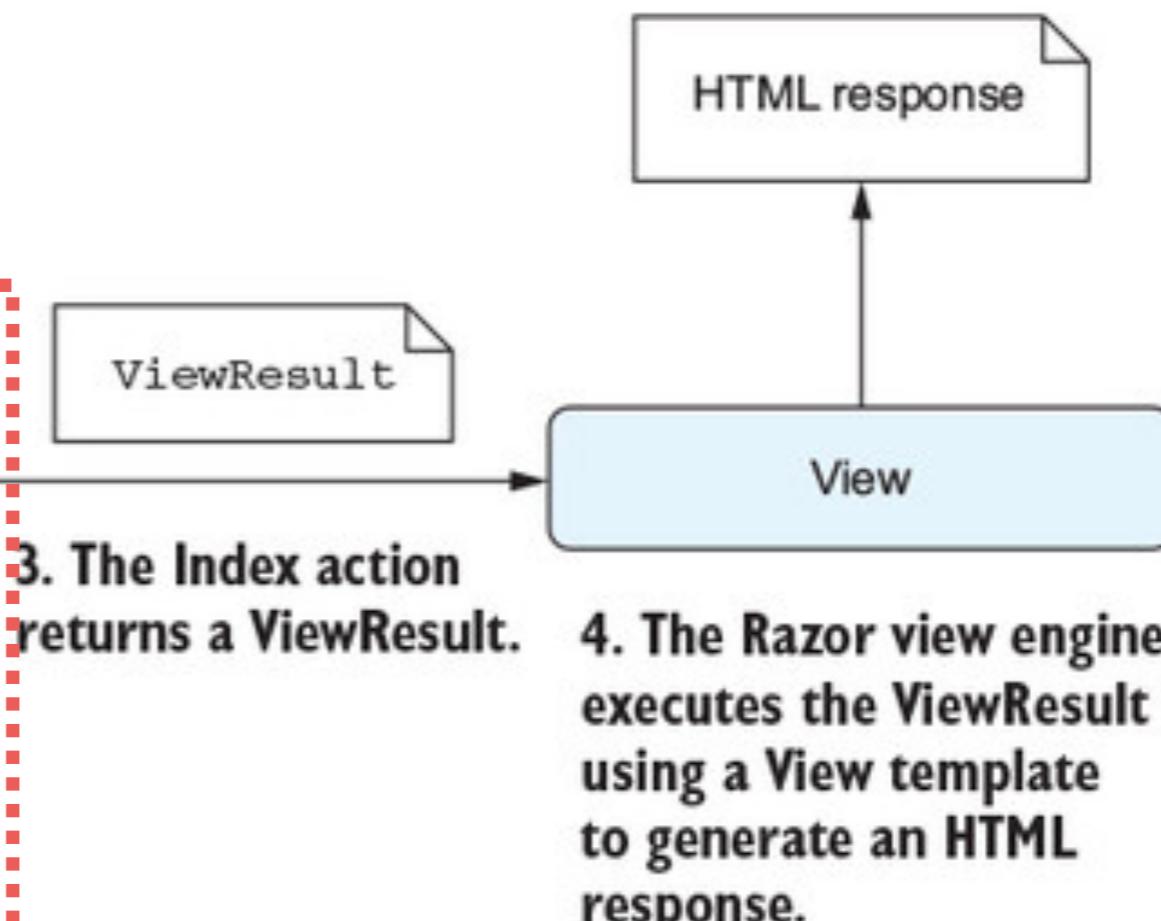


# Route request to MVC controller

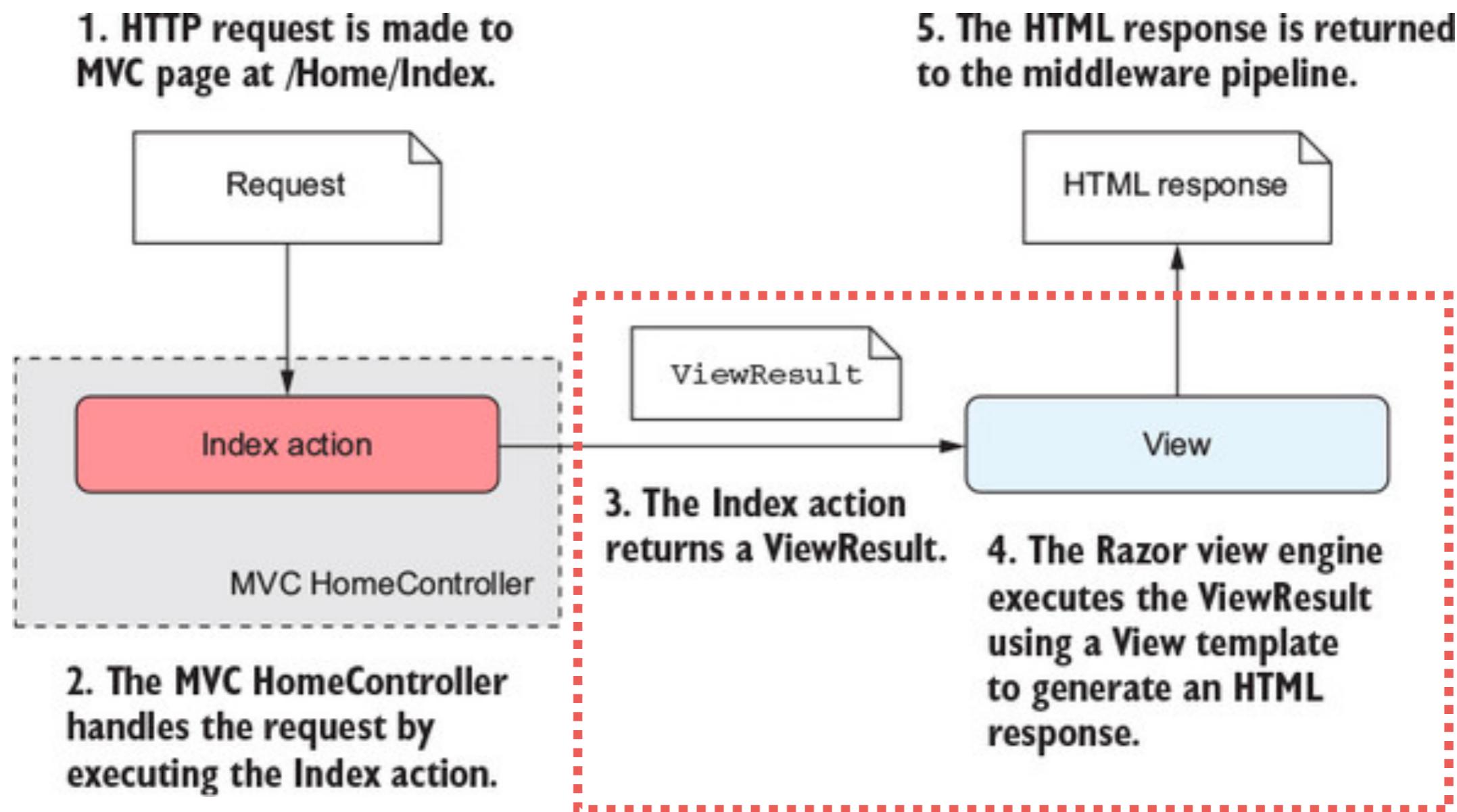
1. HTTP request is made to MVC page at /Home/Index.



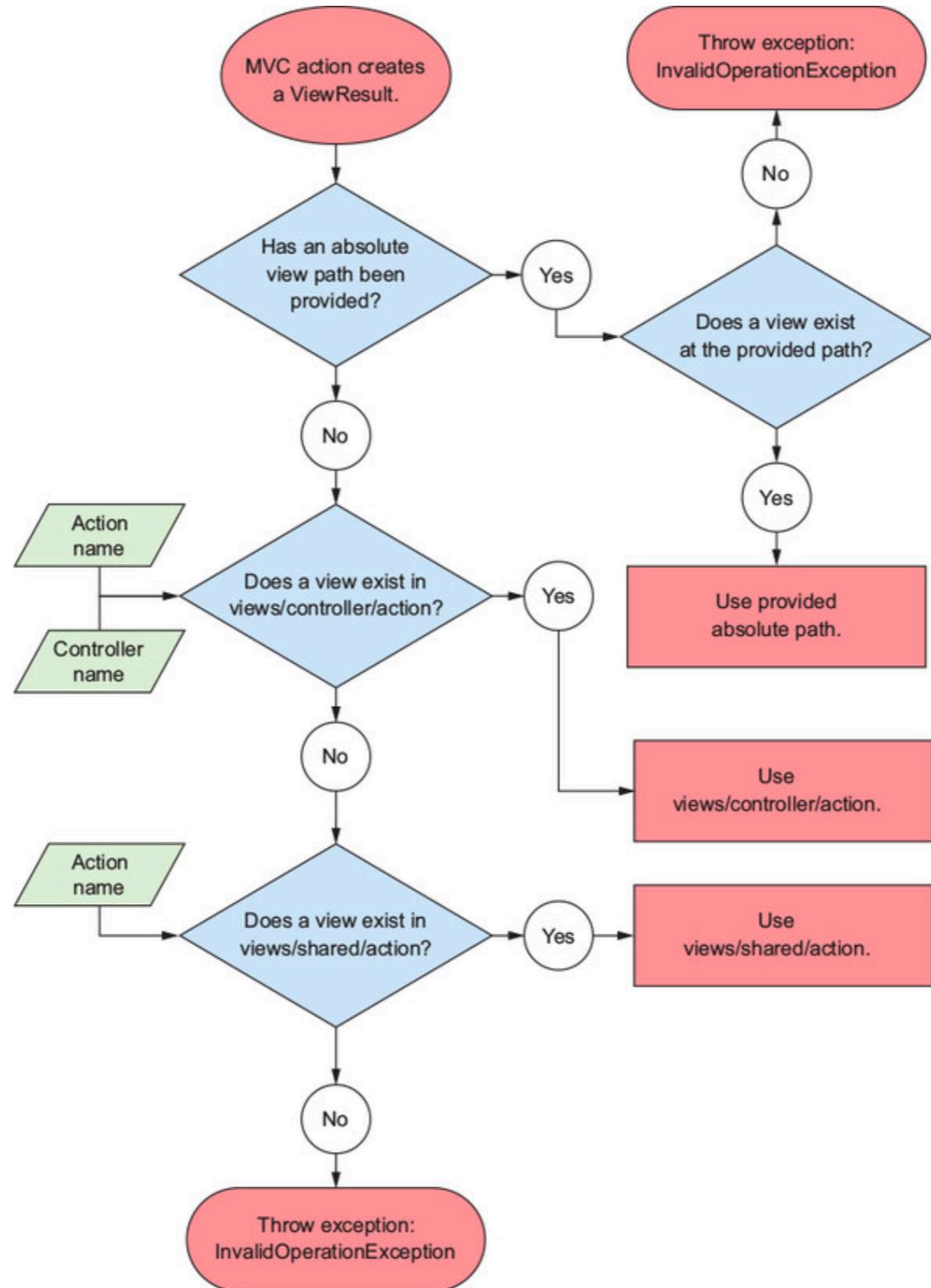
5. The HTML response is returned to the middleware pipeline.



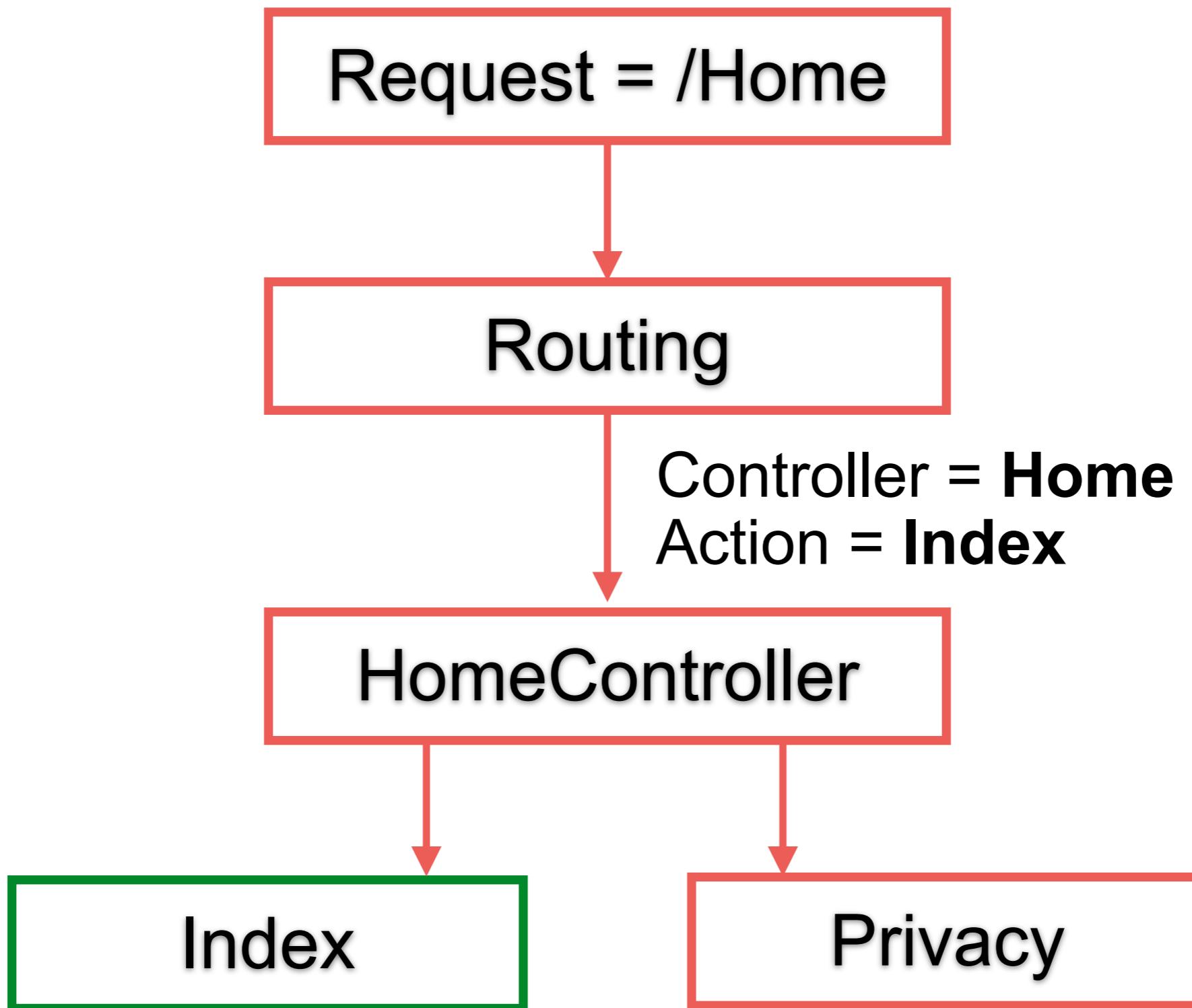
# Map Controller action to view



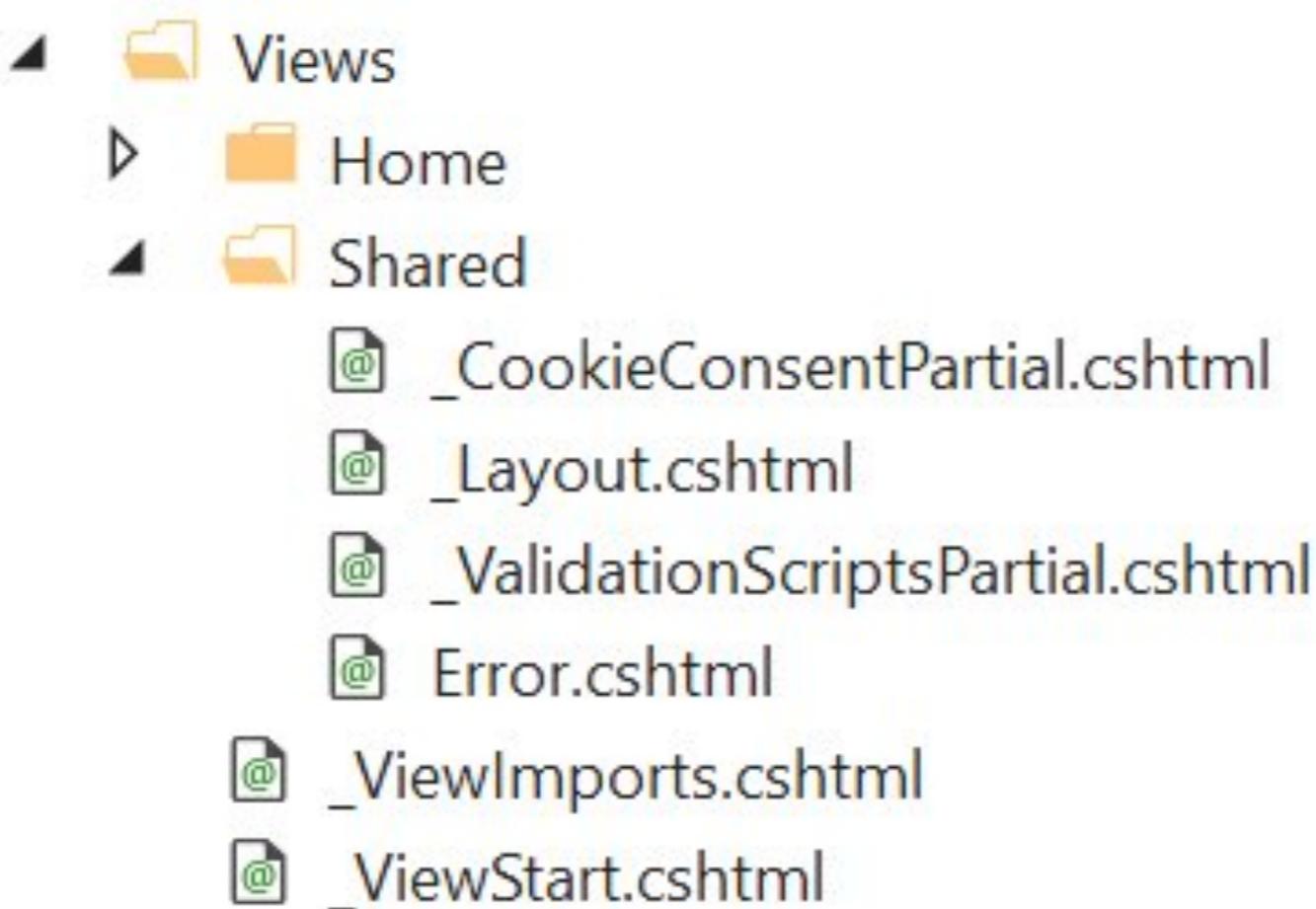
# Map Controller action to view



# Map Controller action to view



# Views in .Net Core MVC



# Views as Razor templates

Views are .cshtml

Can use C# code in Razor markup

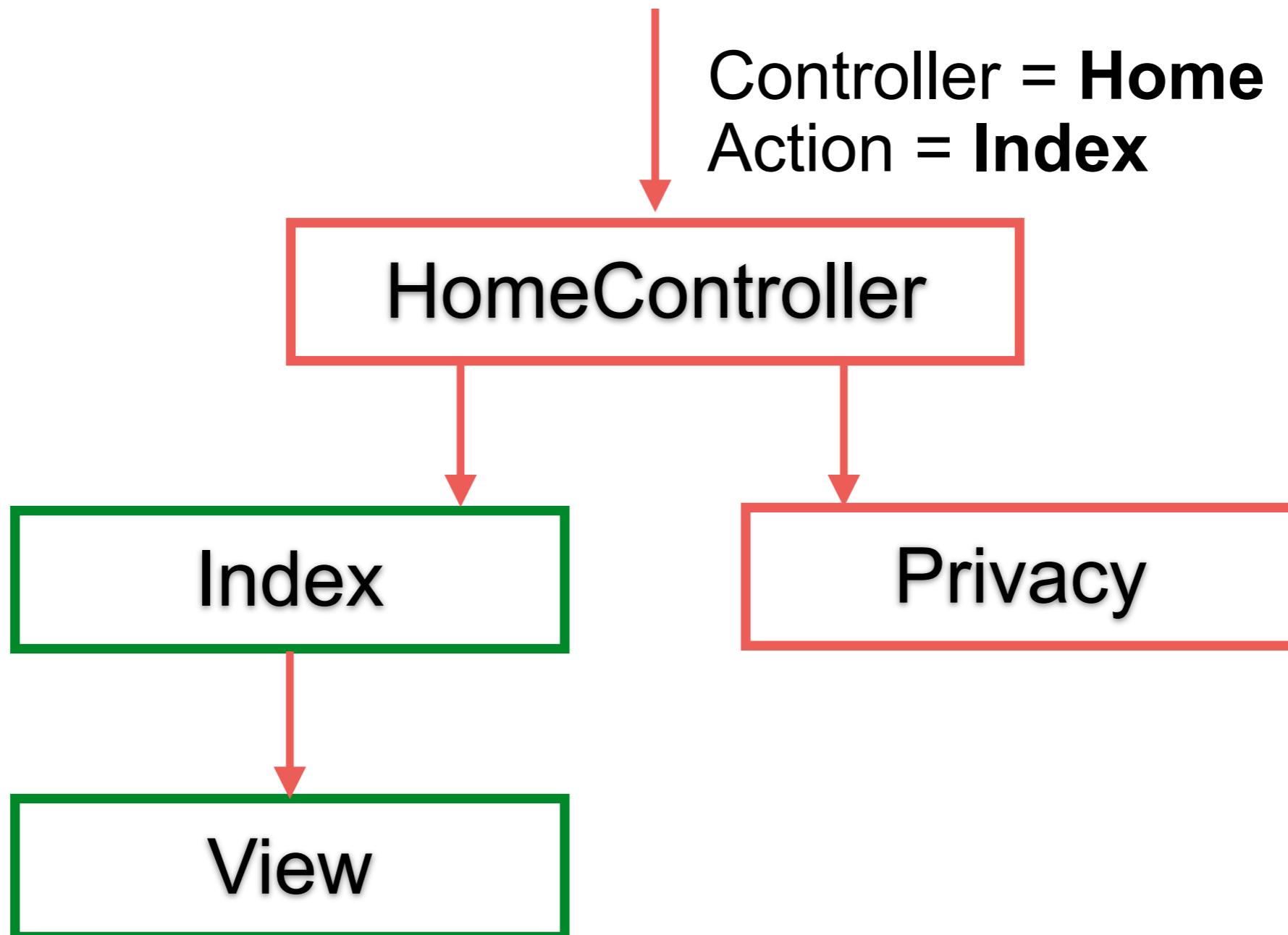
In folder /Views/<controller name>

- <Controller action name 1>.cshtml
- <Controller action name 2>.cshtml
- <Controller action name 3>.cshtml

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview>



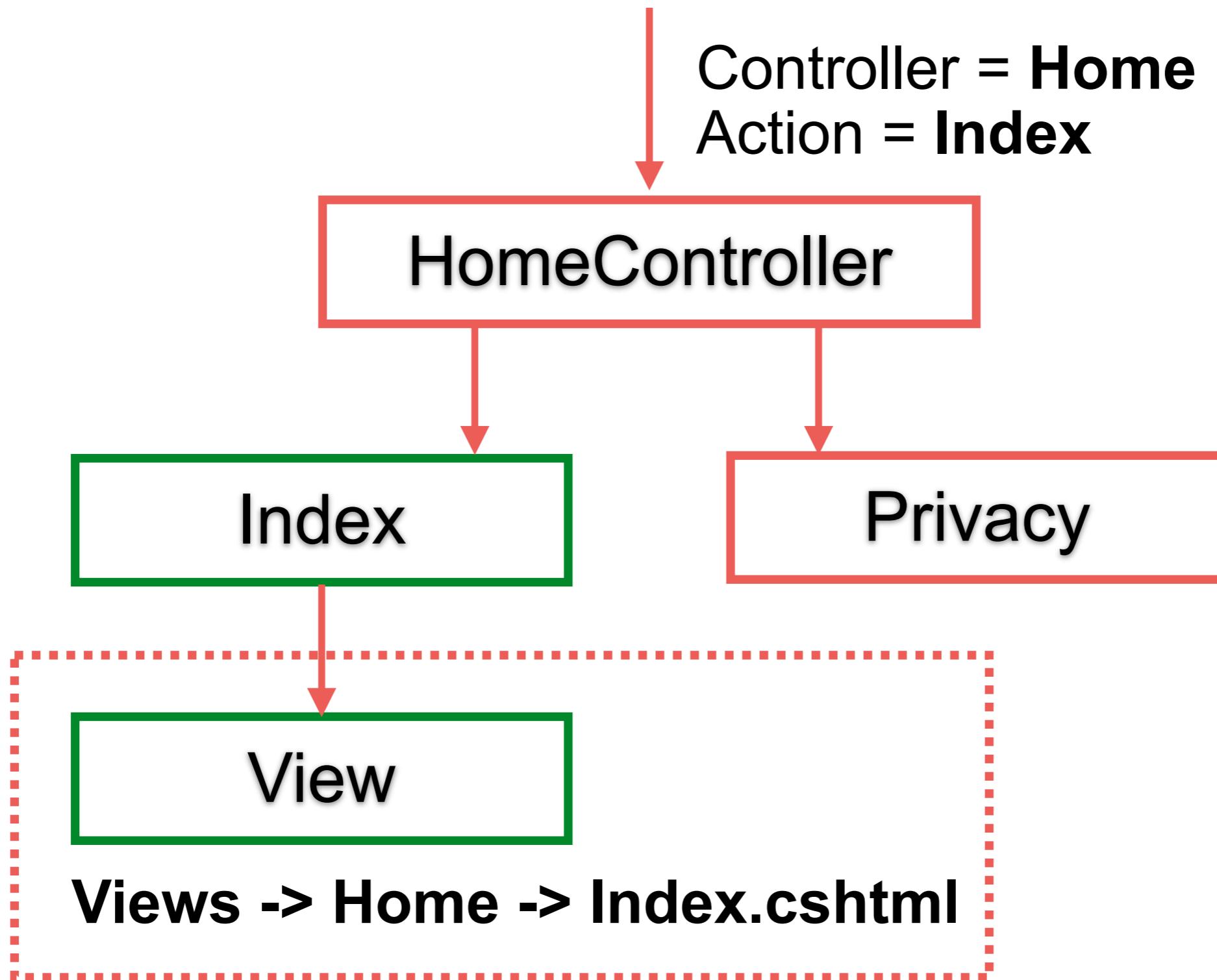
# Map Controller action to view



**Views -> Home -> Index.cshtml**



# How to load view ?



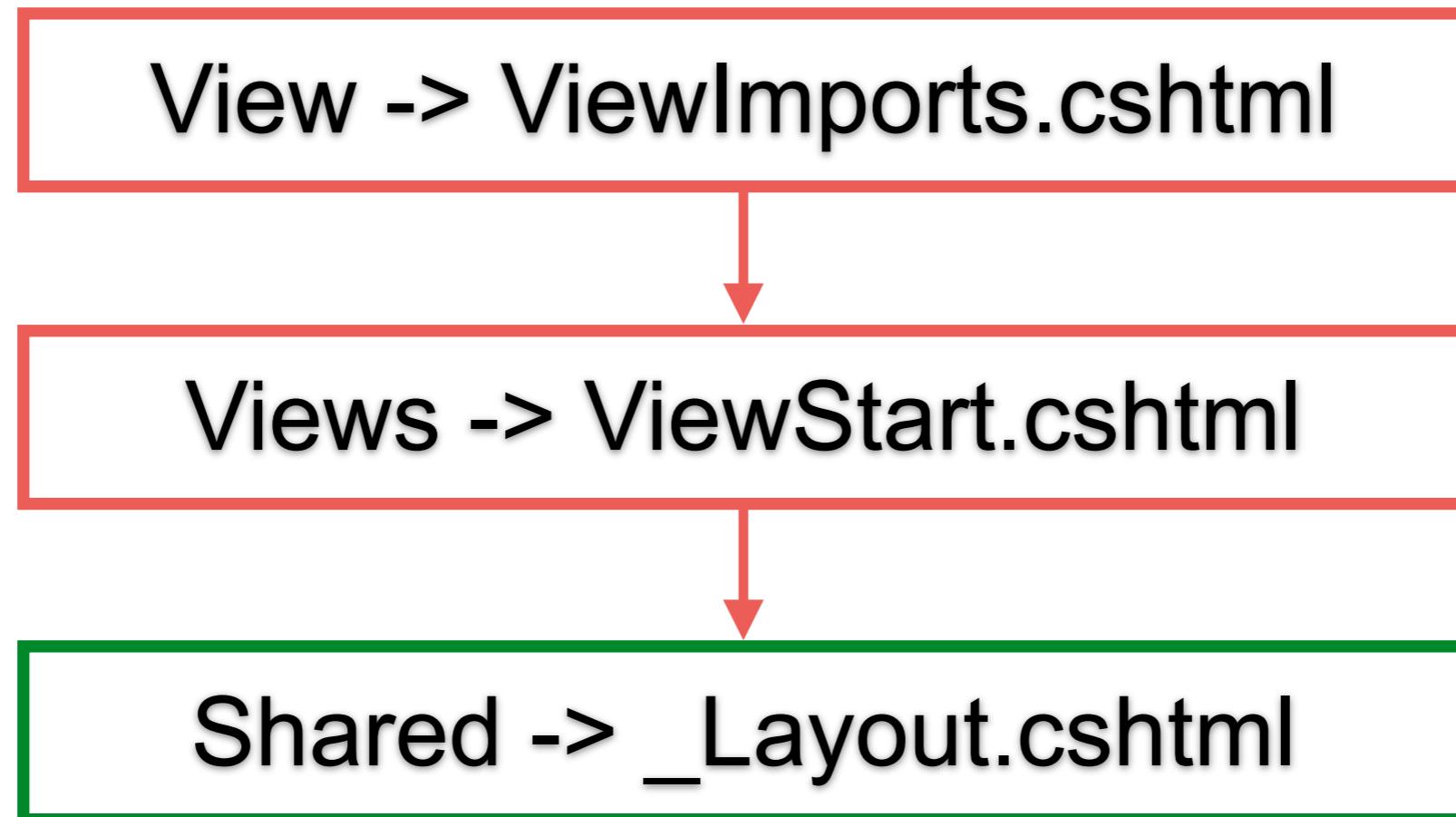
# Index.cshtml

Razor template with C# code

```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Learn about <a href="https://docs.micro  
</div>
```



# Flow of view



<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/layout>



# ViewStart.cshtml

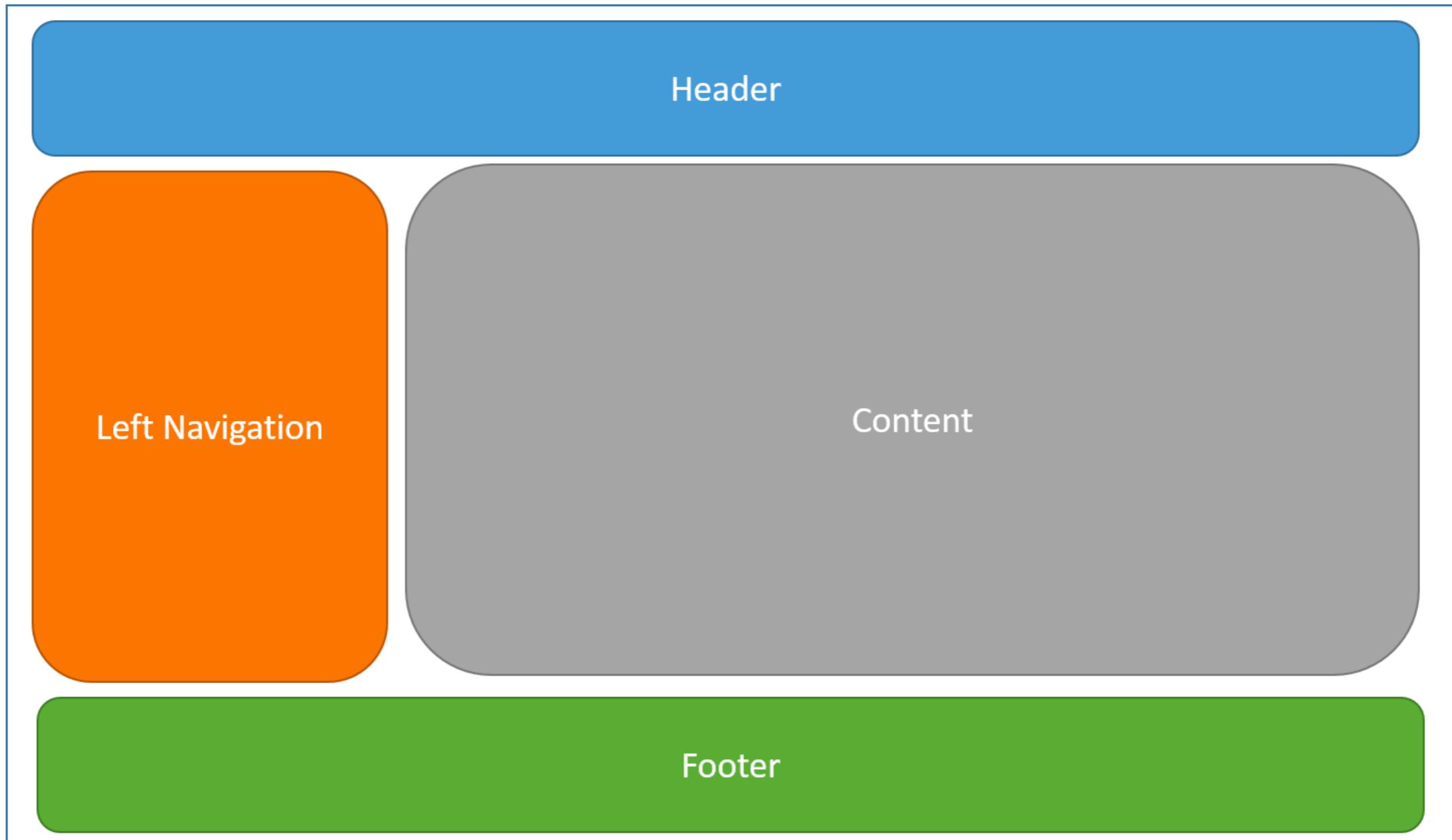
```
@{  
    Layout = "_Layout";  
}
```

The layout specified can use a **full** and **partial** path

*Views/Shared/\_Layout.cshtml*  
*Pages/Shared/\_Layout.cshtml*



# Layout



<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/layout>



# @RenderBody()

```
    </div>
</nav>
</header>

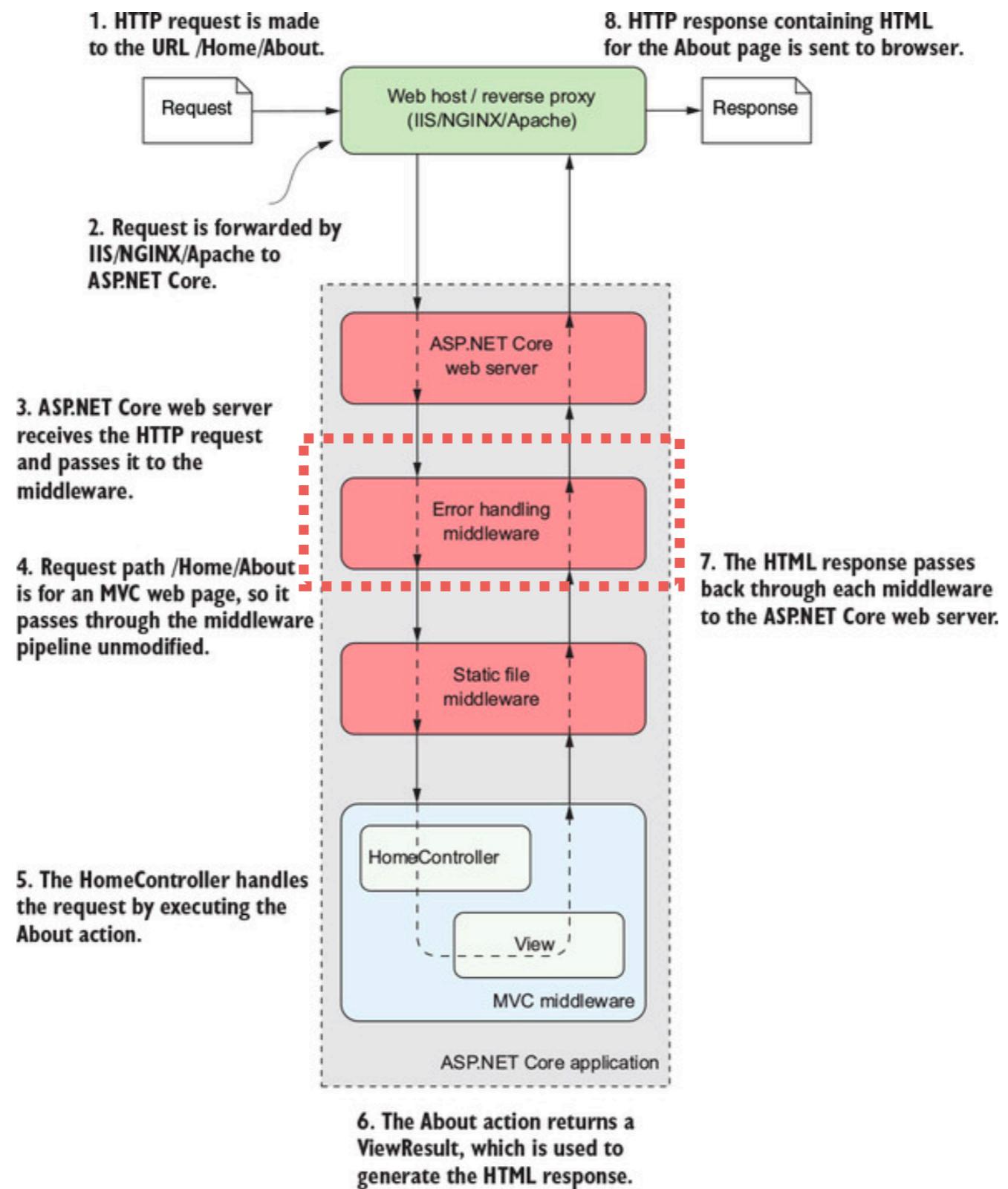

<main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top footer text-muted">
    <div class="container">


```

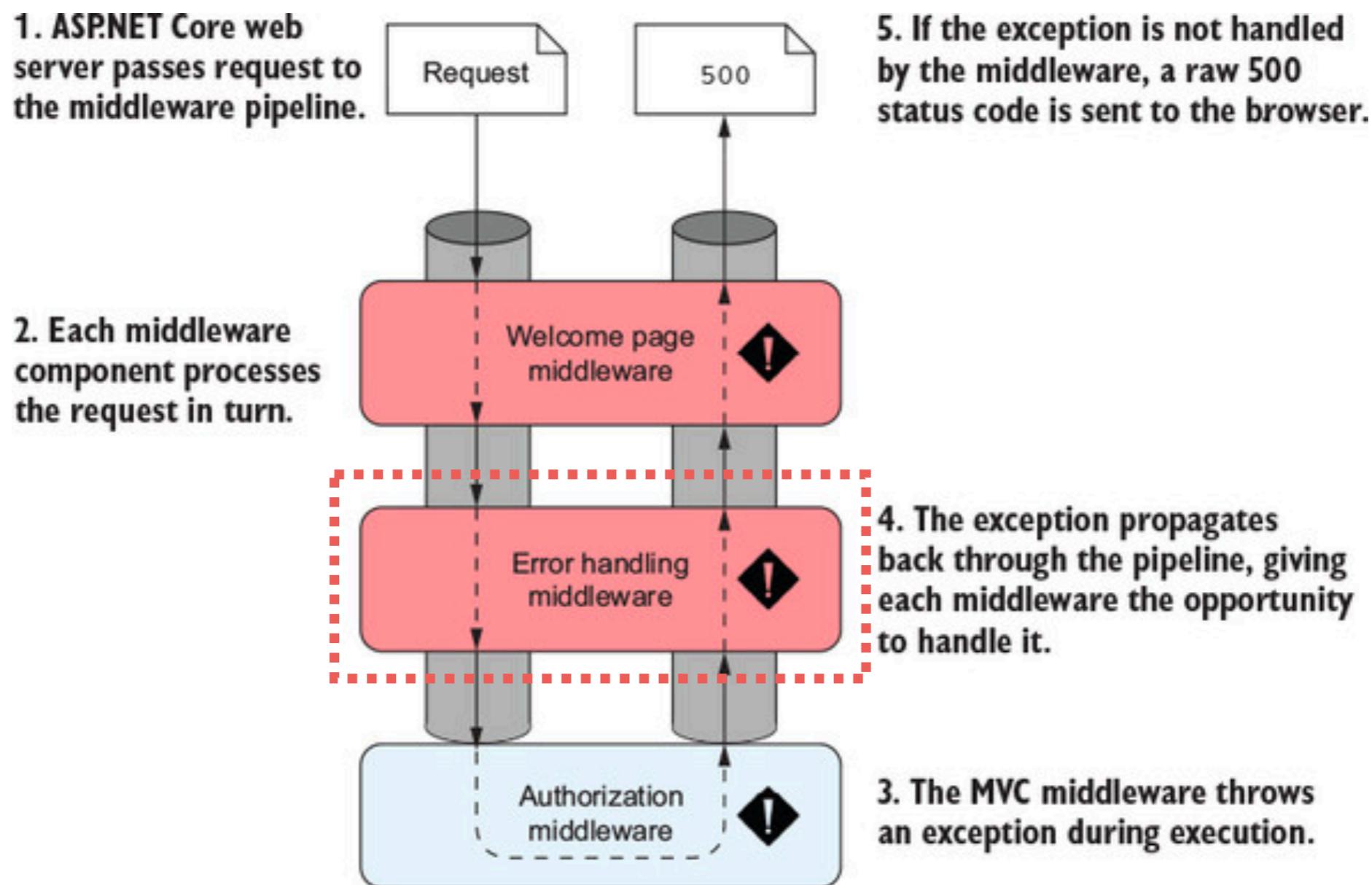


# Summary request to HomeController



# Error handling

## Using middleware to handle an error



# Error handling

UseStatusCodePages

UseStatusCodePagesWithRedirect

UseStatusCodePagesWithReExecute

*Declare before Static of MVC middleware*

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling>



# Error handling

Development mode  
Production mode



# Config exception handling

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this.
        app.UseHsts();
    }
}
```



# Config exception handling

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this
        app.UseHsts();
    }
}
```

Development env



# Config exception handling

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this
        app.UseHsts();
    }
}
```

Production env



# HTTP response status code

Code	Description
1xx	Information
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

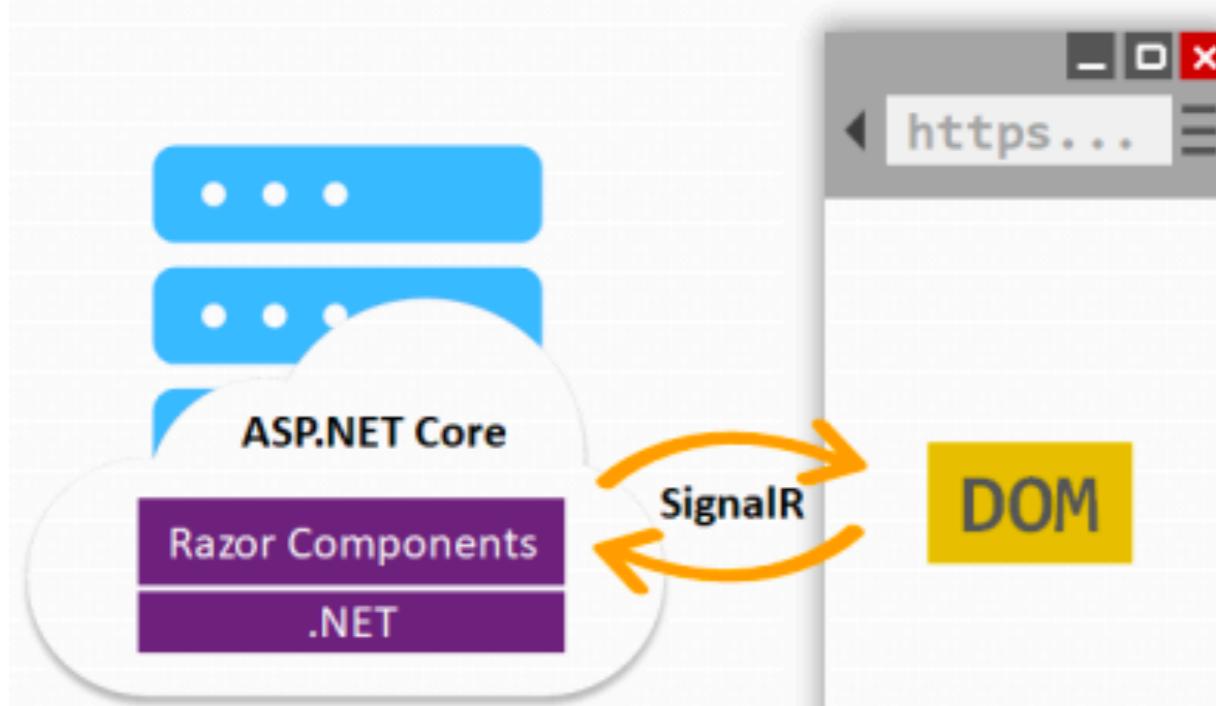


# Blazor in .Net Core 3.1

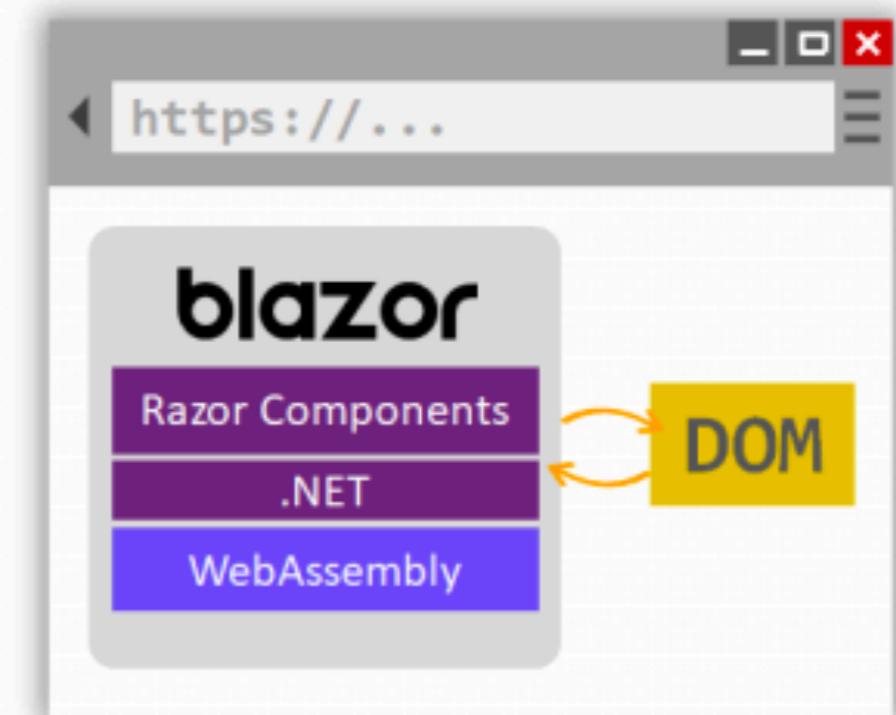


# Blazor in .Net Core 3.1

## Server-side Blazor



## Client-Side Blazor



<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor>



.Net Core

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# .Net Core Blazor

Create UI using C# instead of JavaScript

Share server and client side app logics

Render UI as HTML and CSS in any devices

Interops with JavaScript

<https://docs.microsoft.com/en-us/aspnet/core/blazor>



# Create and Publish Blazor

```
$dotnet new blazorwasm -o BlazorApp1  
$cd BlazorApp1  
$dotnet publish -c Release
```



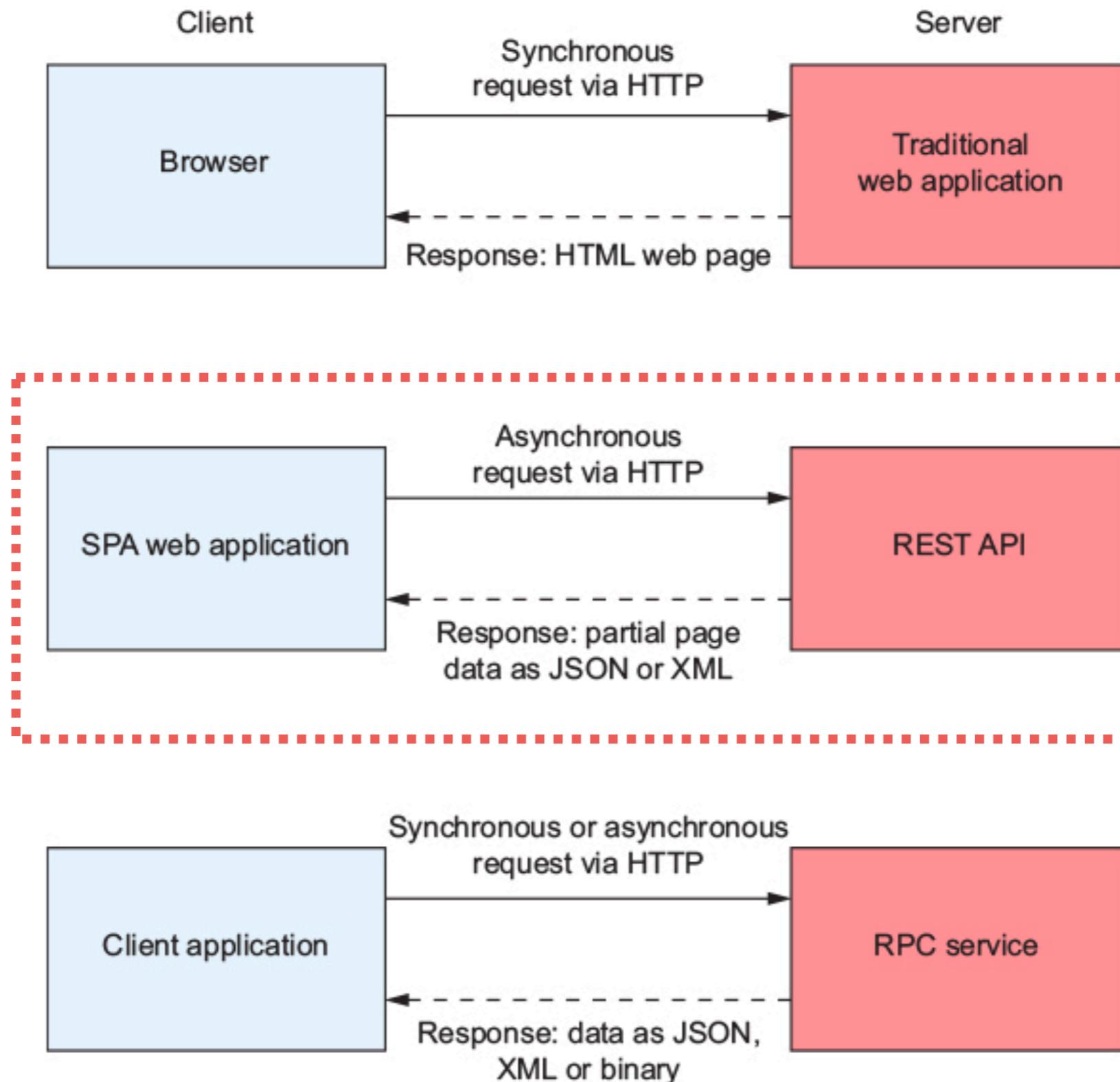
# Develop Web APIs



# Develop Web APIs



# Development



# Startup.cs

Inject the **controllers** to services

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
}
```



# Startup.cs

## Config all middleware

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseCors();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```



# ControllerBase class

All controllers are derive from this class  
Starter controller of API project

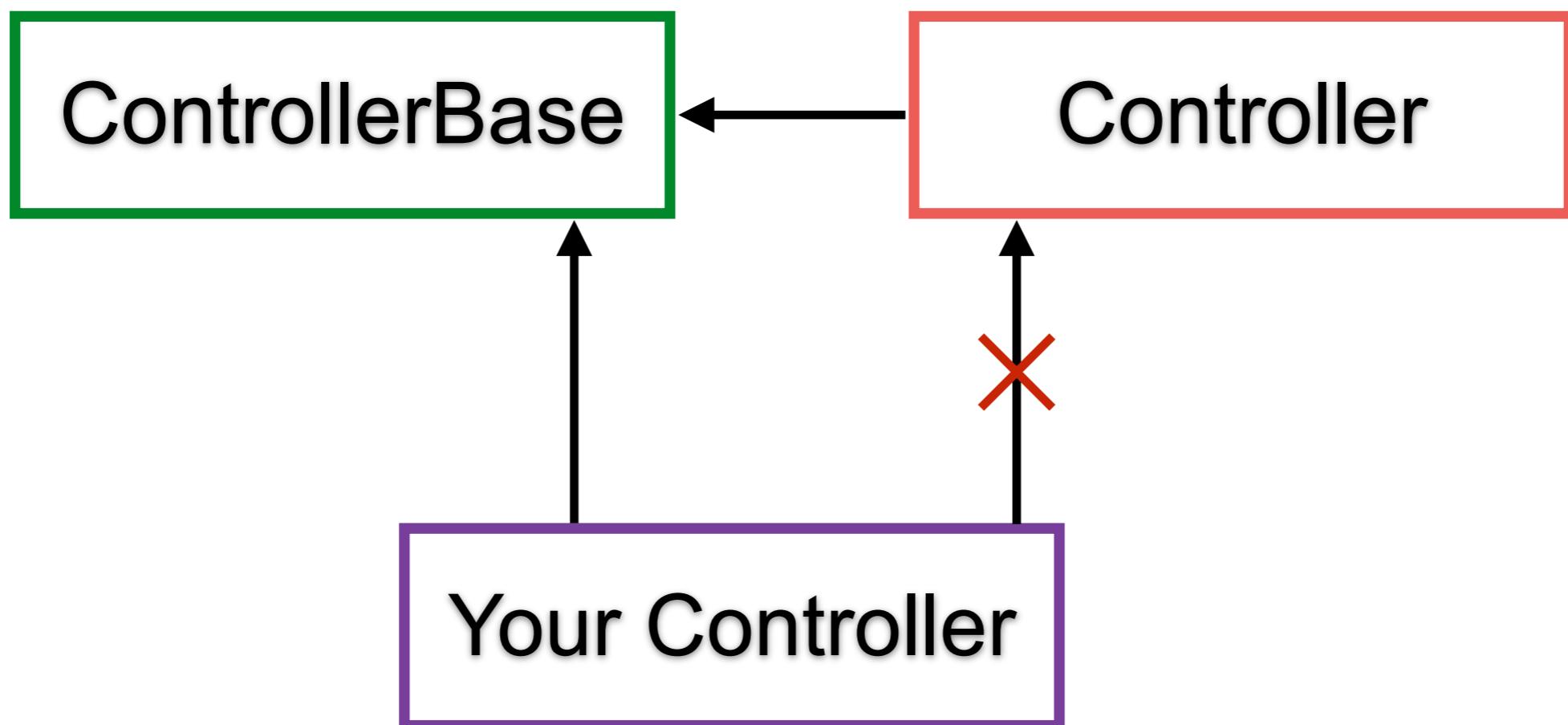
```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
{
```

<https://docs.microsoft.com/en-us/aspnet/core/web-api>



# All controllers

Don't derive from Controller class (For web)



# Action return types of Controller

Specific type  
IActionResult  
ActionResult<T>

<https://docs.microsoft.com/en-us/aspnet/core/web-api/action-return-types>



# Specific type

## List<T>

```
[HttpGet]
[Route("/type1")]
public List<WeatherForecast> GetWithSpecificType()
{
    var rng = new Random();
    return Enumerable.Range(1, 5).Select(index => new Weather
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = rng.Next(-20, 55),
        Summary = Summaries[rng.Next(Summaries.Length)]
    })
    .ToList();
}
```



# Specific type

## IEnumerable<T>

```
[HttpGet]
public IEnumerable<WeatherForecast> Get()
{
    var rng = new Random();
    return Enumerable.Range(1, 5).Select(index => new Weather
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = rng.Next(-20, 55),
        Summary = Summaries[rng.Next(Summaries.Length)]
    })
    .ToArray();
}
```



# Specific type

## IAsyncEnumerable<T>

```
[HttpGet("async")]
public async IAsyncEnumerable<WeatherForecast> GetWithAsync()
{
    var rng = new Random();
    var results = Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = rng.Next(-20, 55),
        Summary = Summaries[rng.Next(Summaries.Length)]
    }).ToList();

    foreach(var item in results)
    {
        yield return item;
    }
}
```



# IActionResult type

Represent HTTP status code

Code	Class name
200	OkObjectResult
400	BadRequestResult
404	NotFoundResult

<https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.iactionresult>



# IActionResult type

Support both Sync and Async

```
[HttpGet("action/result")]
[ProducesResponseType(StatusCodes.Status200K)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public IActionResult GetData()
{
    var rng = new Random();
    if(rng.Next(-5, 5) > 0)
    {
        var results = Enumerable.Range(1, 3).Select(index =>
        {
            Date = DateTime.Now.AddDays(index),
            TemperatureC = rng.Next(-20, 55),
            Summary = Summaries[rng.Next(Summaries.Length)]
        }).ToList();

        return Ok(results);
    }
    return NotFound();
}
```



# IActionResult type

Return HTTP Code = 404

```
{  
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",  
  "title": "Not Found",  
  "status": 404,  
  "traceId": "1379e284f-4c225bd0f80199ab."  
}
```



# ActionResult<T> type

Introduce in .Net Core 2.1

Improvement of IActionResult

Support both Sync and Async



# ActionResult<T> type

Introduce in .Net Core 2.1

```
[HttpGet("action/result2")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public ActionResult<IEnumerable<WeatherForecast>> GetData2()
{
    var rng = new Random();
    if (rng.Next(-5, 5) > 0)
    {
        var results = Enumerable.Range(1, 3).Select(index => new WeatherFore
        {
            Date = DateTime.Now.AddDays(index),
            TemperatureC = rng.Next(-20, 55),
            Summary = Summaries[rng.Next(Summaries.Length)]  
}).ToList();

        return results; Don't need Ok()
    }
    return NotFound();
}
```



# Testing APIs with HTTP REPL

<https://docs.microsoft.com/en-us/aspnet/core/web-api/http-repl>



# HTTP REPL

Lightweight and cross-platform CLI tool  
Use for making HTTP requests to test APIs



# Install HTTP REPL

```
$dotnet tool install -g Microsoft.dotnet-httorepl  
$httorepl -h
```



# Testing with Postman



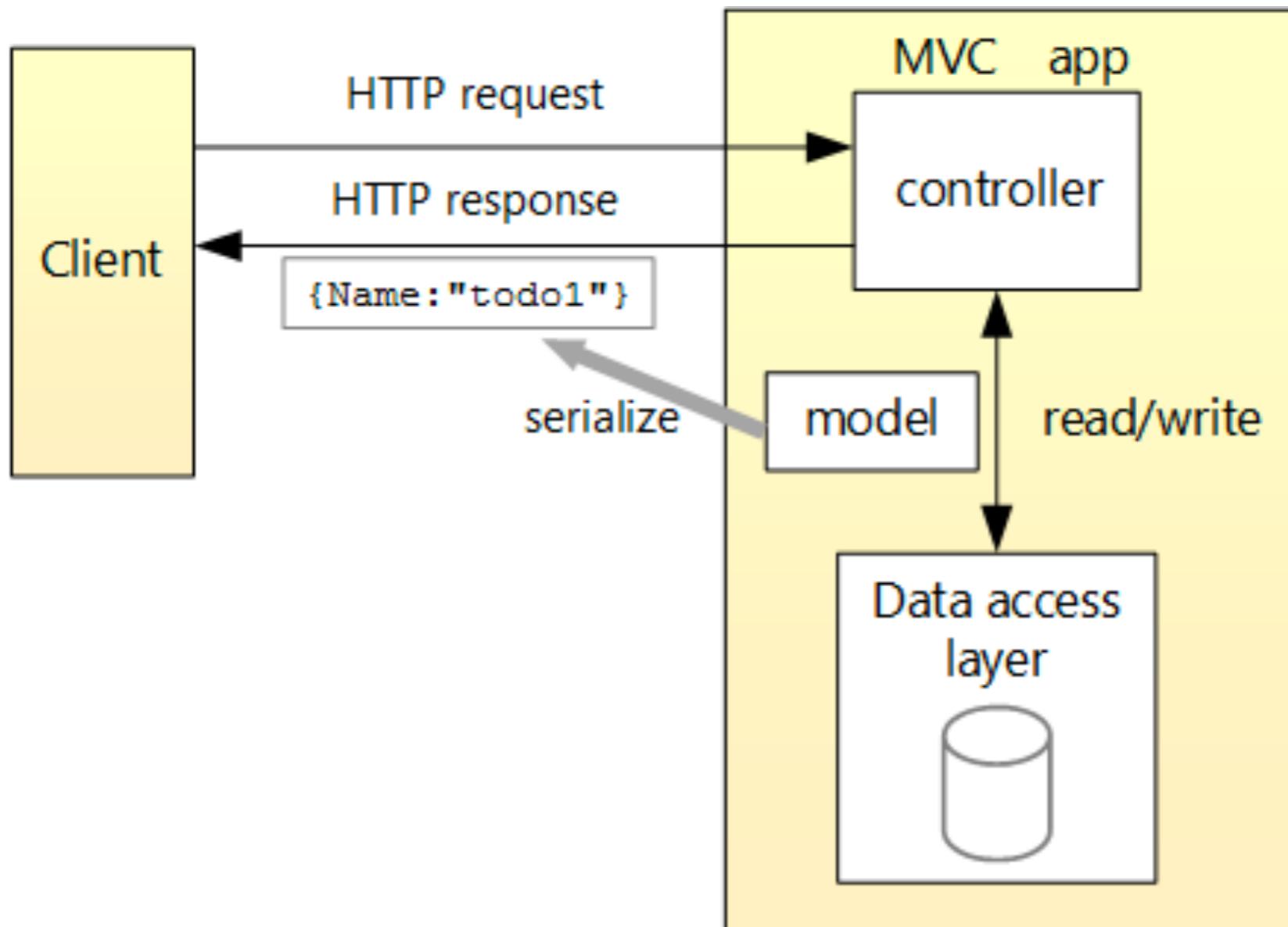
<https://www.postman.com/>



# API Workshop



# API Workshop



<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api>



# Steps

Design your APIs  
Develop  
Testing  
Deploy



# Design your APIs

Add new item

Get an item by ID

Get all items

Update an existing item

Delete an item



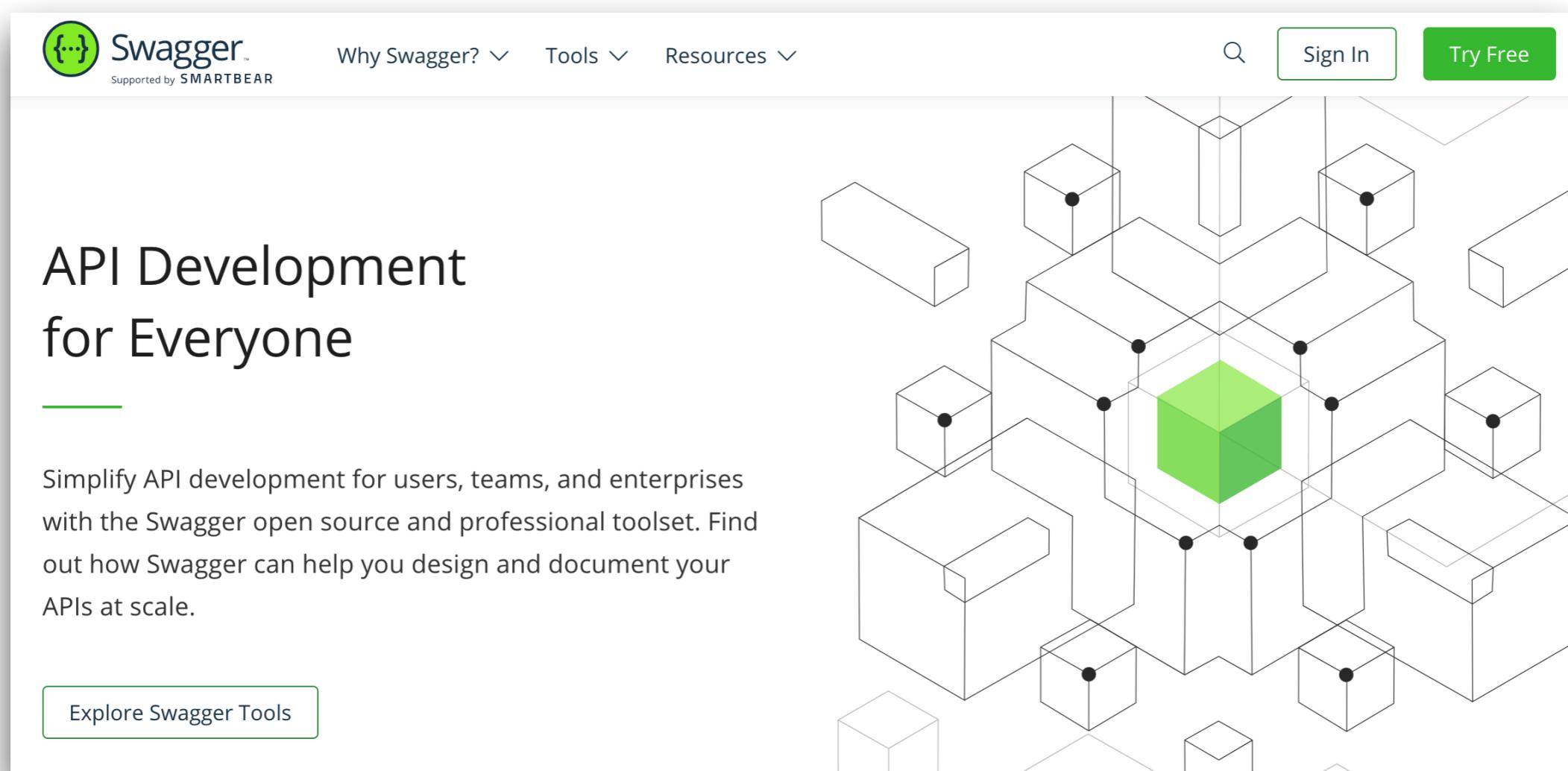
# Design your APIs

API	Description	Request body	Response body
GET /api/item	Get all items	None	List of item
GET /api/item/{id}	Get an item by ID	None	Item
POST /api/item	Add a new item	Item	Item
PUT /api/item/{id}	Update an existing item	Item	None
DELETE /api/item/{id}	Delete an item	None	None



# Try to create API Document

## Using swagger



The screenshot shows the official Swagger website. At the top, there's a navigation bar with the Swagger logo (a green circle with white brackets), the text "Supported by SMARTBEAR", and links for "Why Swagger?", "Tools", and "Resources". On the right side of the header are a search icon, a "Sign In" button, and a green "Try Free" button. The main content area features a large, stylized 3D wireframe cube composed of many smaller cubes. One central cube is highlighted in green. To the left of the cube, the text "API Development for Everyone" is displayed above a horizontal line, followed by a paragraph about Swagger's purpose and a "Explore Swagger Tools" button.

<https://swagger.io/>



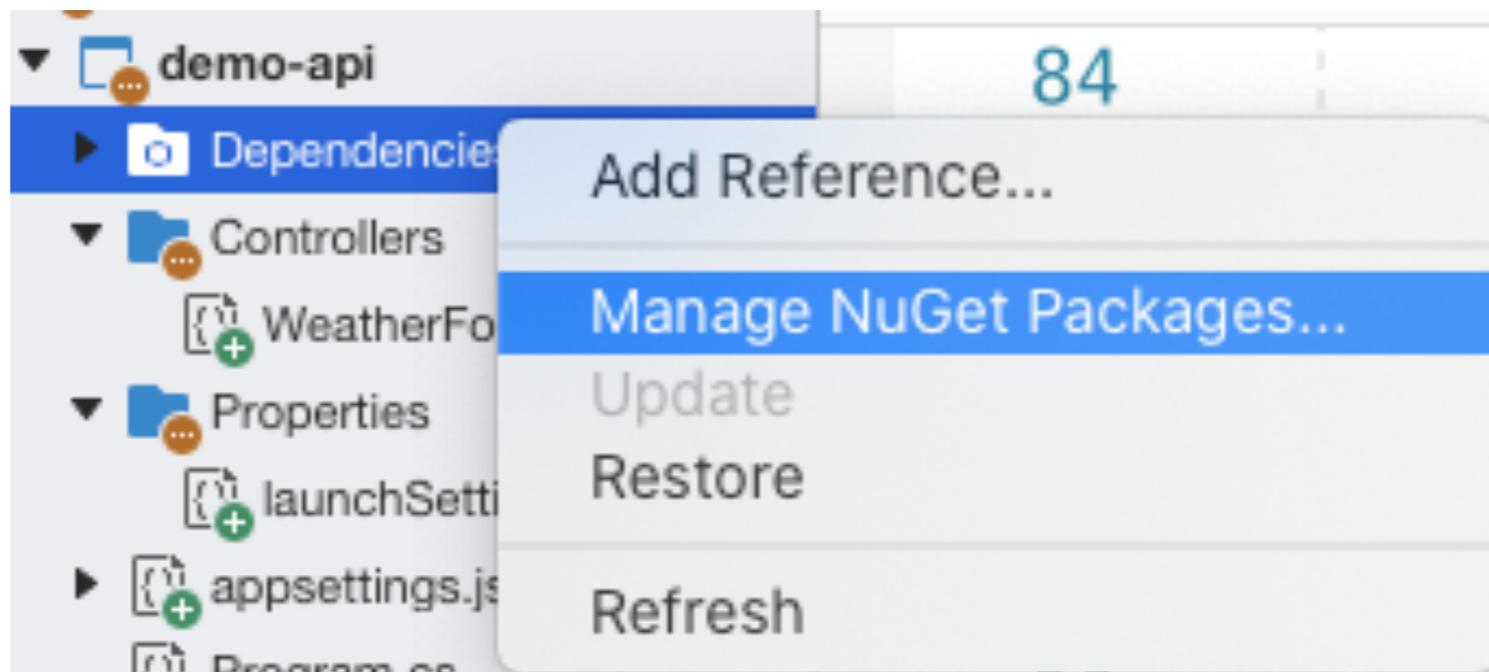
# Swagger in .Net Core

Swashbuckle  
**NSwag**



# Install NSwag

Go to Package Manager Console  
Or Manage NuGet Package



# Install NSwag

## Manage NuGet Package

The screenshot shows the 'Manage NuGet Packages – demo-api' dialog box from the NuGet.org website. The search bar at the top right contains the text 'NSwag'. The left pane lists several NSwag packages:

- NSwag.AspNetCore** (5,157,399) - NSwag: The OpenAPI/Swagger API toolchain for .NET and TypeScript. This package is selected, indicated by a checked checkbox icon.
- NSwag.Generation** (2,328,261) - NSwag: The OpenAPI/Swagger API toolchain for .NET and TypeScript. An unchecked checkbox icon is next to it.
- NSwag.SwaggerGeneration** (3,439,859) - NSwag: The Swagger API toolchain for .NET and TypeScript. An unchecked checkbox icon is next to it.
- NSwag.MSBuild** (2,205,481) - NSwag: The OpenAPI/Swagger API toolchain for .NET and TypeScript. An unchecked checkbox icon is next to it.
- NSwag.SwaggerGeneration.WebApi** (3,309,895) - NSwag: The Swagger API toolchain for .NET and TypeScript. An unchecked checkbox icon is next to it.
- NSwag.SwaggerGeneration.AspNetCore** (2,810,454) - NSwag: The Swagger API toolchain for .NET and TypeScript. An unchecked checkbox icon is next to it.

The right pane displays detailed information for the selected package, **NSwag.AspNetCore**:

- ID**: NSwag.AspNetCore
- Author**: Rico Suter
- Published**: 5/8/2020
- Downloads**: 5,157,399
- License**: [View License](#)
- Project Page**: [Visit Page](#)
- Dependencies**:
  - NSwag.Annotations (>= 13.5.0)
  - NSwag.Core (>= 13.5.0)
  - NSwag.Generation.AspNetCore (>= 13.5.0)
  - NSwag.Generation (>= 13.5.0)
  - Microsoft.AspNetCore.Mvc.Core (>= 1.0.3)
  - Microsoft.AspNetCore.Mvc.Formatters.Json (>= 1.0.3)
  - Microsoft.AspNetCore.StaticFiles (>= 1.0.2)
  - Microsoft.Extensions.ApiDescription.Server (>= 3.0.0)
  - Microsoft.Extensions.FileProviders.Embedded (>= 1.0.1)
  - NETStandard.Library (>= 1.6.1)
  - System.IO.FileSystem (>= 4.3.0)
  - System.Xml.XPath.XDocument (>= 4.0.1)

At the bottom of the right pane, there is a dropdown menu labeled 'New Version:' with the value '13.5.0 (latest stable)'.

At the bottom left of the main area, there is a checkbox labeled 'Show pre-release packages'. At the bottom right, there are 'Close' and 'Add Package' buttons.



# Config swagger middleware

## Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    services.AddSwaggerDocument();
}

public void Configure(IApplicationBuilder app, IWebHostEnvirc
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseOpenApi();
    app.UseSwaggerUi3();
}
```



# Swagger UI

<https://localhost:5001/swagger>

The screenshot shows the Swagger UI interface for a project titled "My Title" version 1.0.0. The base URL is set to localhost:5001 and the swagger definition file is /swagger/v1/swagger.json. The "Schemes" dropdown is set to HTTPS. The main content area displays the "WeatherForecast" endpoint with five listed operations:

- GET /WeatherForecast
- GET /WeatherForecast/async
- GET /type1
- GET /WeatherForecast/action/result
- GET /WeatherForecast/action/result2



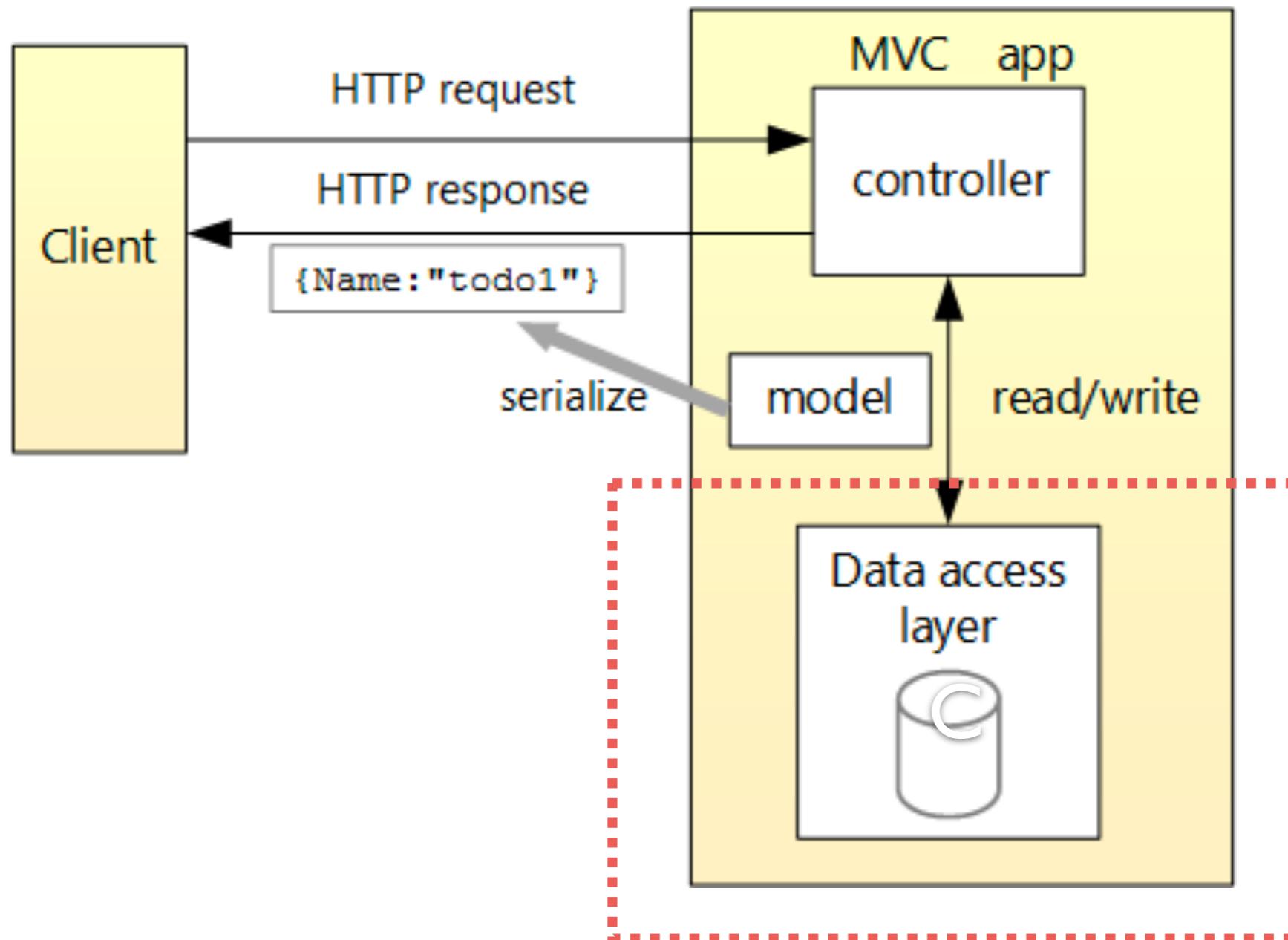
# Swagger specification

<https://localhost:5001/swagger/v1/swagger.json>

```
{  
    "x-generator": "NSwag v13.5.0.0 (NJsonSchema v10.1.15.0 (Newtonsoft.Json v9.0.0.0))"  
    "swagger": "2.0",  
    "info": {  
        "title": "My Title",  
        "version": "1.0.0"  
    },  
    "host": "localhost:5001",  
    "schemes": [  
        "https"  
    ],  
    "produces": [  
        "text/plain",  
        "application/json",  
        "text/json"  
    ],  
    "paths": {  
        "/WeatherForecast": {  
            "get": {  
                "tags": [  
                    "WeatherForecast"  
                ],  
                "summary": "Get weather forecast",  
                "description": "This endpoint returns the current weather forecast.",  
                "responses": {  
                    "200": {  
                        "description": "Success response",  
                        "schema": {  
                            "type": "array",  
                            "items": {  
                                "type": "object",  
                                "properties": {  
                                    "id": {  
                                        "type": "integer",  
                                        "format": "int32"  
                                    },  
                                    "temperature": {  
                                        "type": "number",  
                                        "format": "float"  
                                    },  
                                    "weatherCondition": {  
                                        "type": "string"  
                                    }  
                                }  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



# Develop your APIs



# Install Microsoft.EntityFrameworkCore

## Use In-memory database

The screenshot shows the 'Manage NuGet Packages' dialog from the 'nuget.org' source. The search term 'Microsoft.EntityFrameworkCore.InMemory' is entered in the search bar. The results list the following packages:

ID	Name	Version	Description
Microsoft.EntityFrameworkCore.InMemory	Microsoft.EntityFrameworkCore.InMemory	42,188,437	In-memory database provider for Entity Framework Core (to be used for testing purposes).
Microsoft.EntityFrameworkCore.Sqlite	Microsoft.EntityFrameworkCore.Sqlite	25,402,643	SQLite database provider for Entity Framework Core.
Microsoft.EntityFrameworkCore.Sqlite.Core	Microsoft.EntityFrameworkCore.Sqlite.Core	24,734,153	SQLite database provider for Entity Framework Core.
Microsoft.EntityFrameworkCore.Relational.Design	Microsoft.EntityFrameworkCore.Relational.Design	10,226,551	Shared design-time Entity Framework Core components for relational database providers.
Microsoft.VisualStudio.Web.CodeGeneration	Microsoft.VisualStudio.Web.CodeGeneration	41,489,701	Contains Entity Framework Core Services used by ASP.NET Core Code Generators.
Microsoft.EntityFrameworkCore.Tools.DotNet	Microsoft.EntityFrameworkCore.Tools.DotNet	5,025,046	Entity Framework Core .NET Command Line Tools. Includes dotnet-ef.

On the right side of the dialog, the details for the selected package, 'Microsoft.EntityFrameworkCore.InMemory', are displayed:

Property	Value
ID	Microsoft.EntityFrameworkCore.InMemory
Author	Microsoft
Published	5/12/2020
Downloads	42,188,437
License	<a href="#">View License</a>
Project Page	<a href="#">Visit Page</a>
Dependencies	Microsoft.EntityFrameworkCore (>= 3.1.4)

At the bottom of the dialog, there are buttons for 'Close' and 'Add Packages'. A dropdown menu shows '3.1.4 (latest stable)'.



# Access data from Database

Model class of Table in database  
Config database context to project  
Generate controller



# Table Item

Create class **Item.cs**

```
public class Item
{
    public long Id { get; set; }
    public string Name { get; set; }
    public bool IsComplete { get; set; }
}
```



# Create database context

## Create class ItemContext.cs

```
public class ItemContext : DbContext
{
    public ItemContext(DbContextOptions<ItemContext> options)
        : base(options)
    {
    }

    public DbSet<Item> Items { get; set; }
}
```



# Config database context

## In class Startup.cs

```
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;

namespace demo_api
{
    public class Startup
    {
        public Startup(IConfiguration configuration) ...
        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<ItemContext>(opt =>
                opt.UseInMemoryDatabase("DemoProject"));

            services.AddControllers();
            services.AddSwaggerDocument();
        }
    }
}
```



# Generate controller

Select Add -> New Scaffold

Select Scaffolder

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



**API controller with actions using Entity framework**

-  Razor Page using Entity Framework (CRUD)
-  Identity



# Generate controller

API Controller with actions using Entity Framework

---

Model class to use:

DbContext class to use:

Controller name:



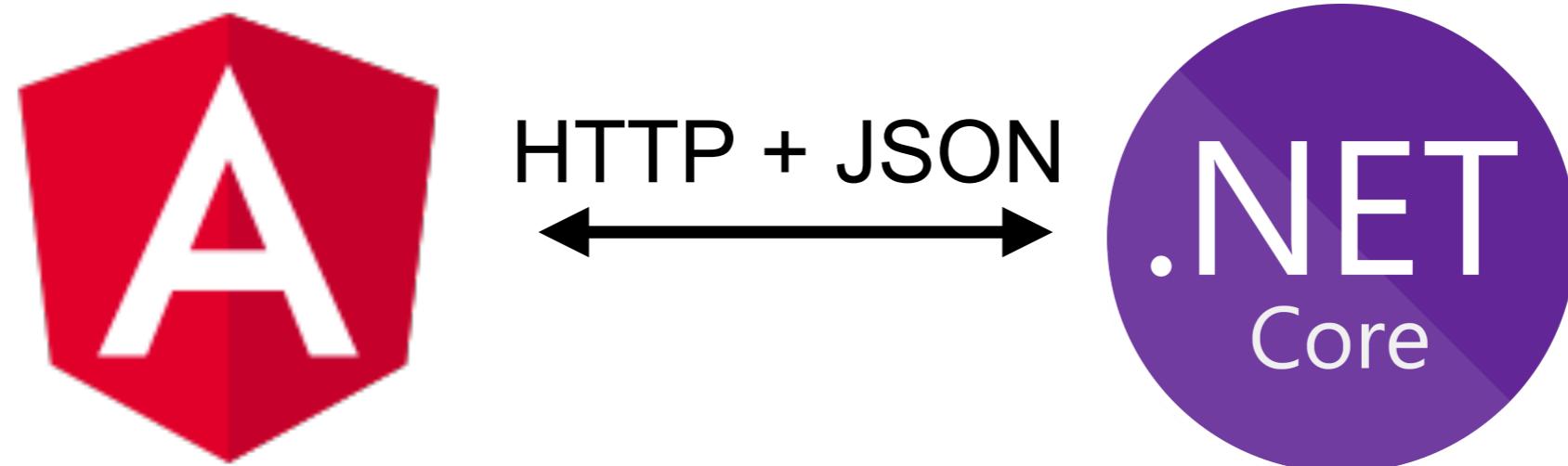
# Testing with Postman



<https://www.postman.com/>



# Web + API



# Working with Angular 9



# Angular

[FEATURES](#)[DOCS](#)[RESOURCES](#)[EVENTS](#)[BLOG](#)

One framework.  
Mobile & desktop.

[GET STARTED](#)

<https://angular.io/>



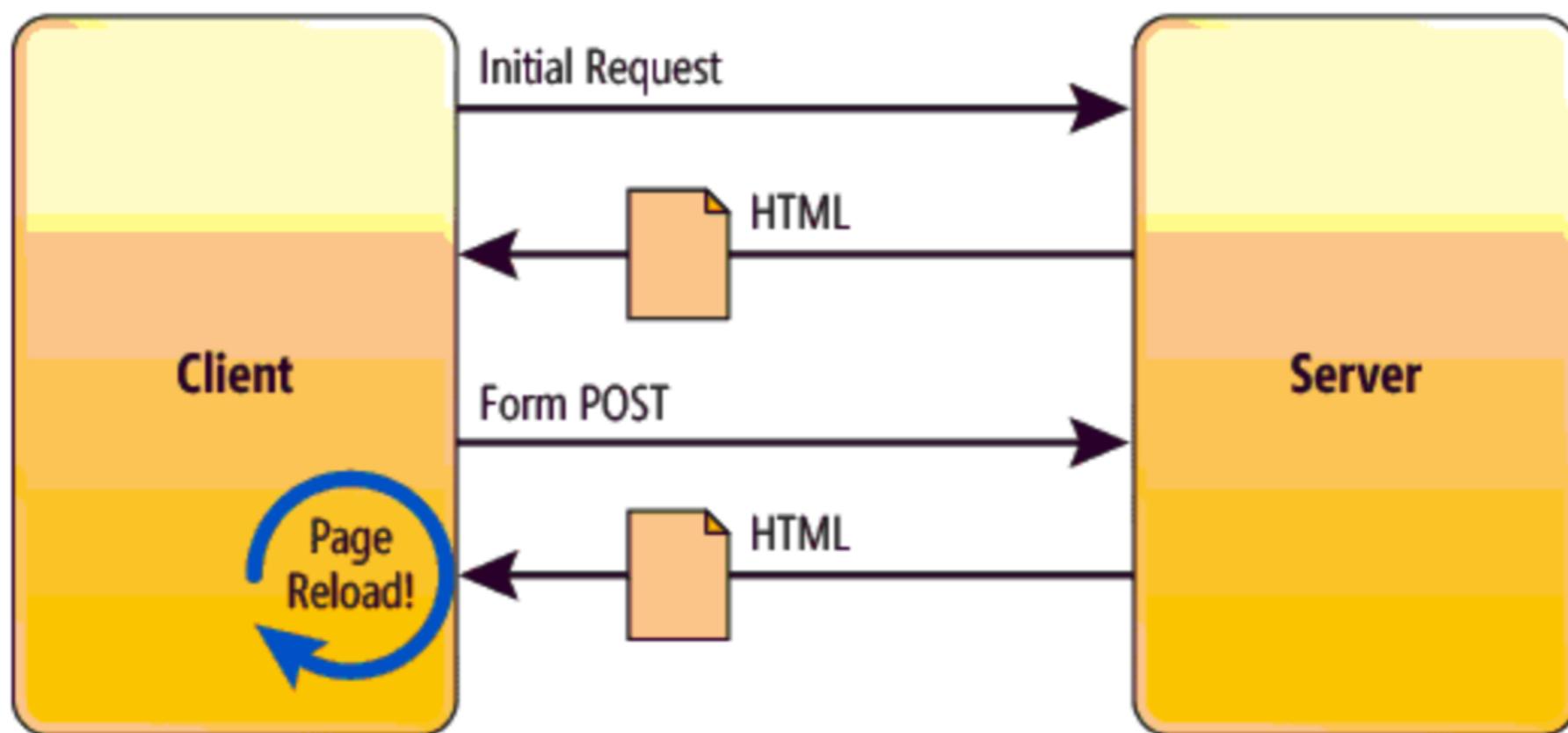
# Angular

**JavaScript framework with allows us to create  
reactive Single Page Application (SPA)**

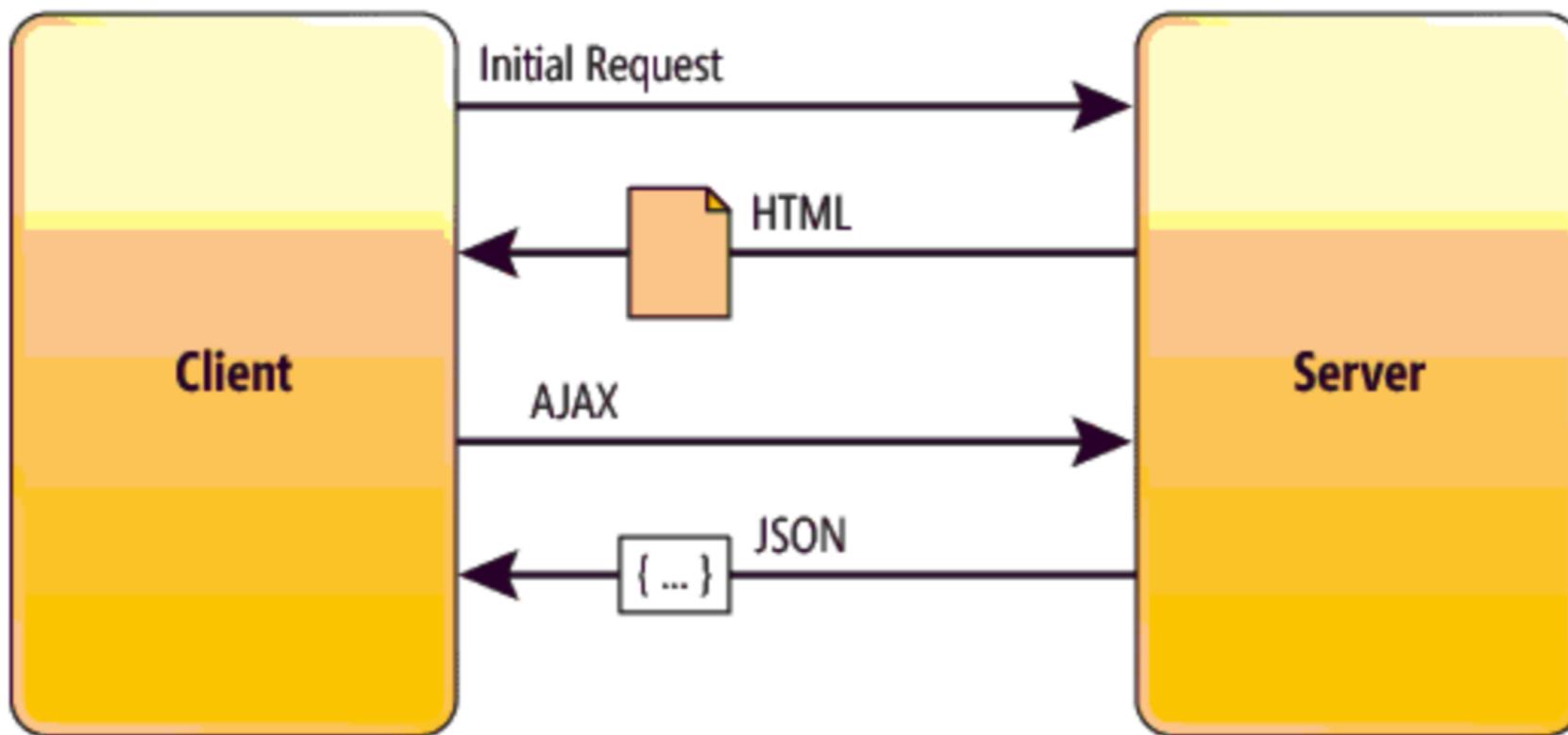
<https://angular.io/>



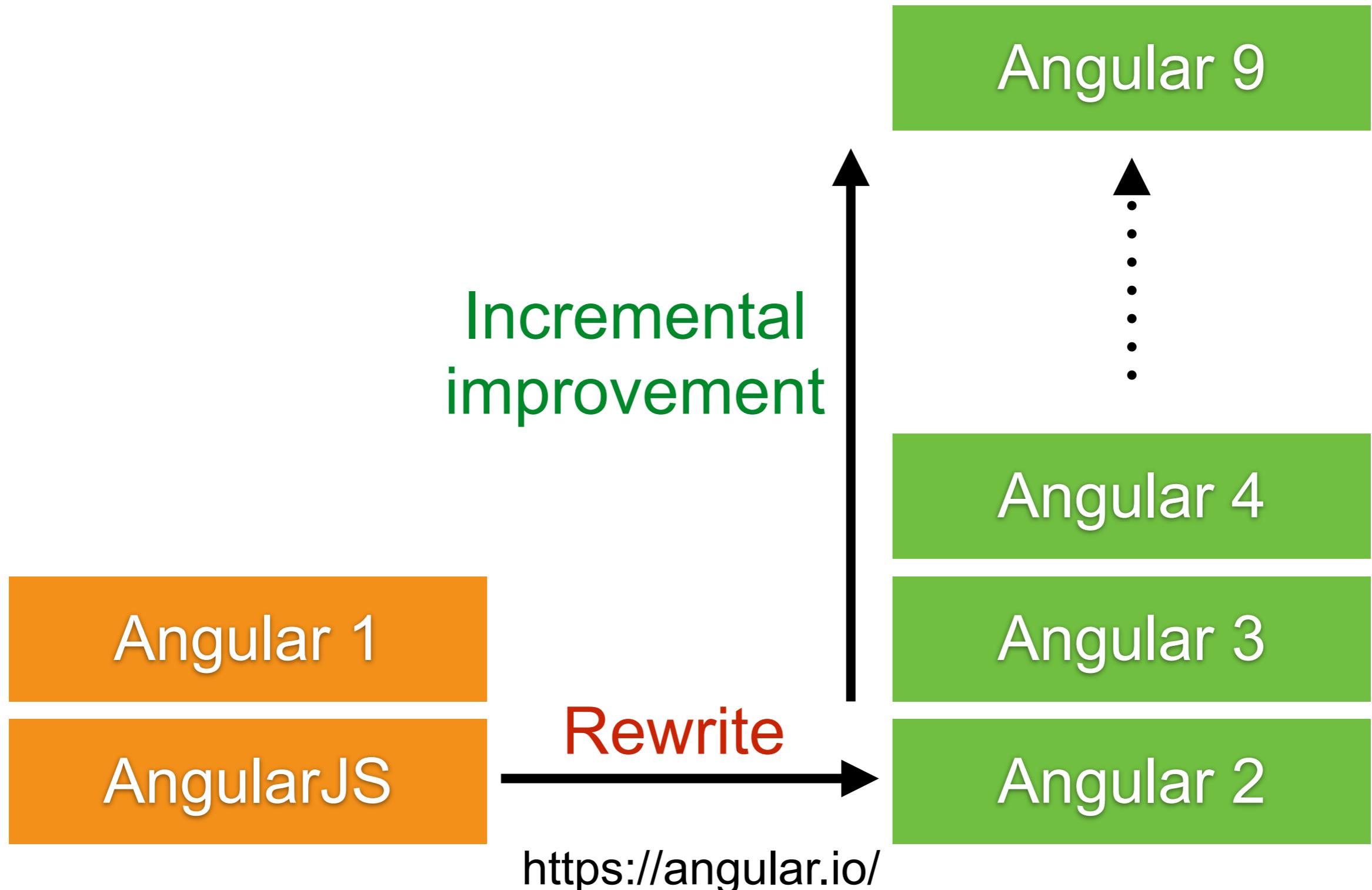
# Traditional



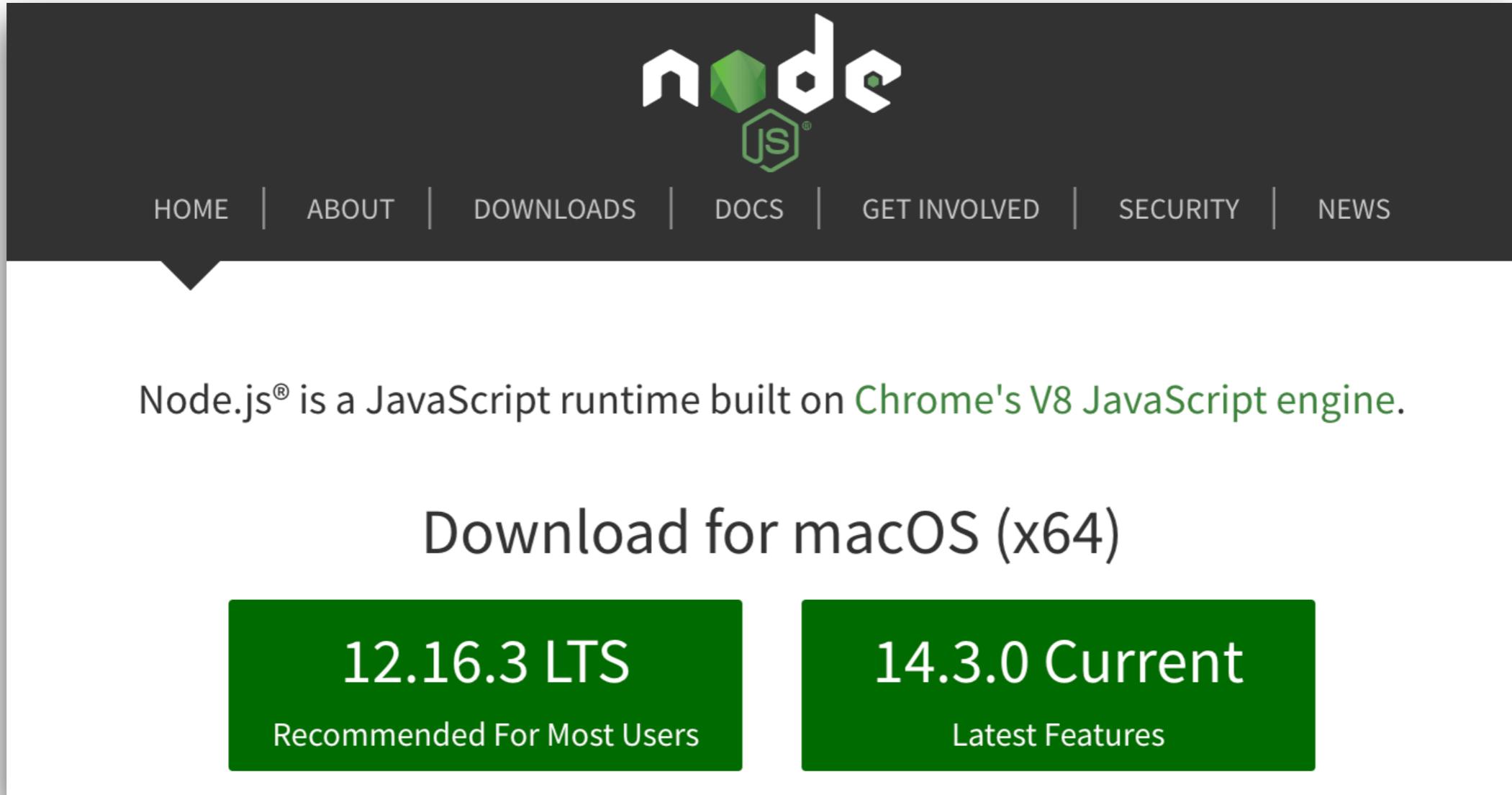
# Single Page Application



# History of Angular



# Software requirement



The screenshot shows the official Node.js website. At the top is a dark navigation bar with the Node.js logo and links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, and NEWS. Below the navigation, a large green header area contains the text "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." In the center, there are two prominent green buttons for downloading Node.js for macOS (x64). The left button is labeled "12.16.3 LTS" and "Recommended For Most Users". The right button is labeled "14.3.0 Current" and "Latest Features".

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for macOS (x64)

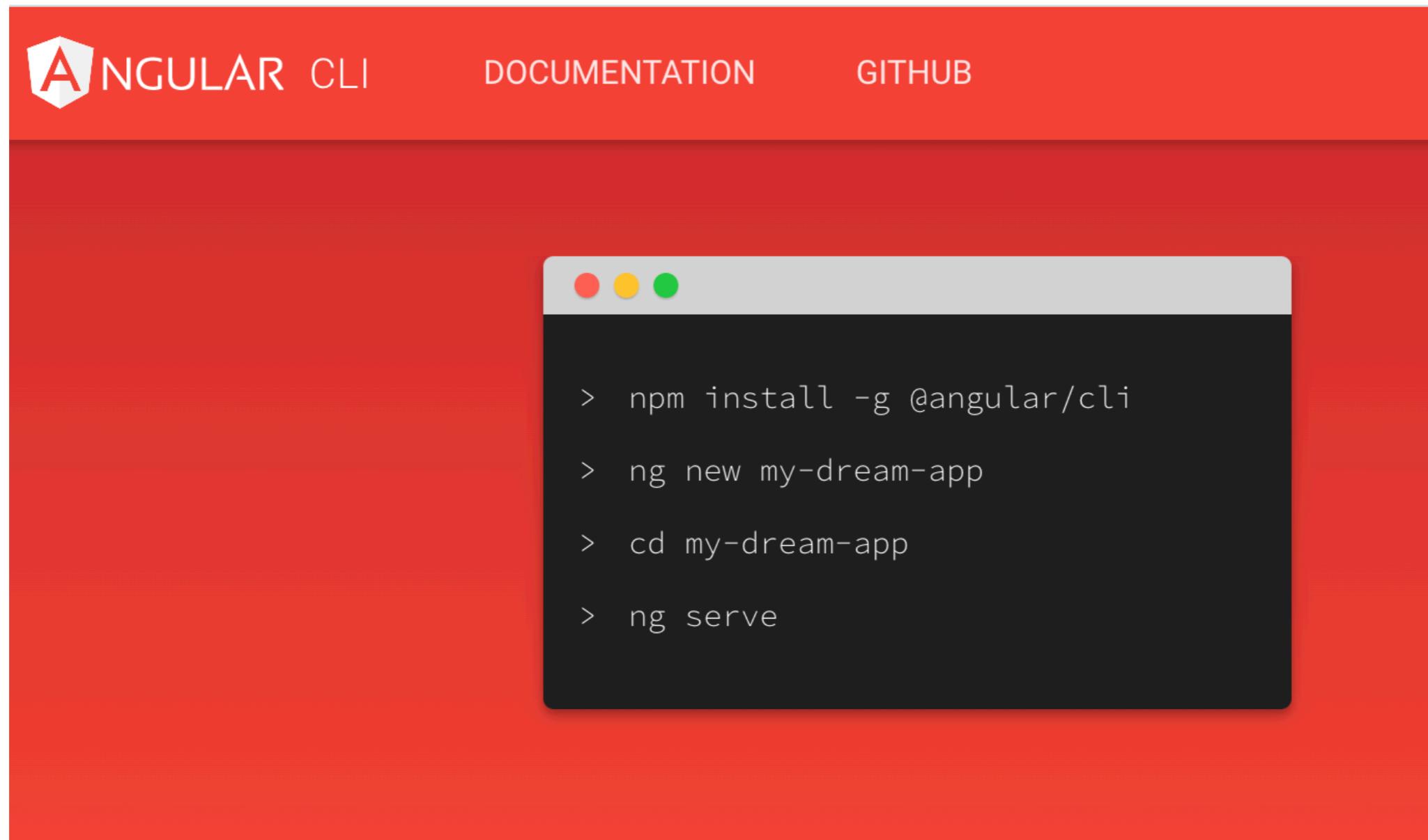
12.16.3 LTS  
Recommended For Most Users

14.3.0 Current  
Latest Features

<https://nodejs.org/en/>



# Create project with Angular CLI



<https://cli.angular.io/>



# Create new project

\$ng new hello-app

```
CREATE hello-app/README.md (1025 bytes)
CREATE hello-app/angular.json (3575 bytes)
CREATE hello-app/package.json (1313 bytes)
CREATE hello-app/tsconfig.json (384 bytes)
CREATE hello-app/tslint.json (2805 bytes)
CREATE hello-app/.editorconfig (245 bytes)
CREATE hello-app/.gitignore (503 bytes)
CREATE hello-app/src/environments/environment.prod.ts (51 bytes)
CREATE hello-app/src/environments/environment.ts (631 bytes)
CREATE hello-app/src/favicon.ico (5430 bytes)
CREATE hello-app/src/index.html (295 bytes)
```



# Run project

```
$cd hello-app  
$ng serve
```

\*\* Angular Live Development Server is listening on localhost:4200, open your browser to <http://localhost:4200>

Date: 2018-07-19T17:42:59.617Z

Hash: cf107798cf25722bc556

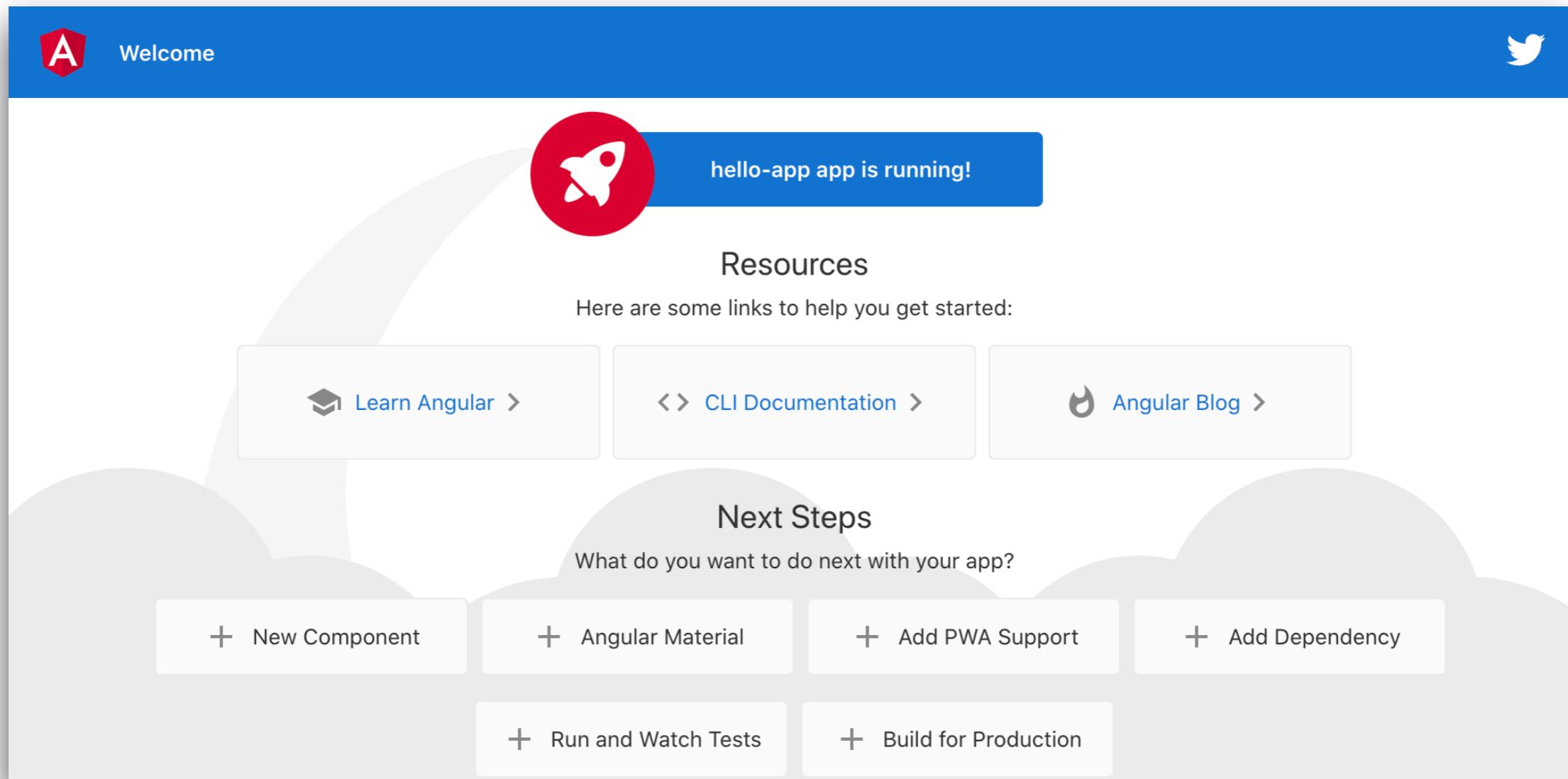
Time: 22285ms

```
chunk {main} main.js, main.js.map (main) 10.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.06 MB [initial] [rendered]
i 「wdm」: Compiled successfully.
```



# Open in browser

<http://localhost:4200/>



# Call .Net Core API with service

## Create APIService.ts

```
constructor(private http: HttpClient) { }

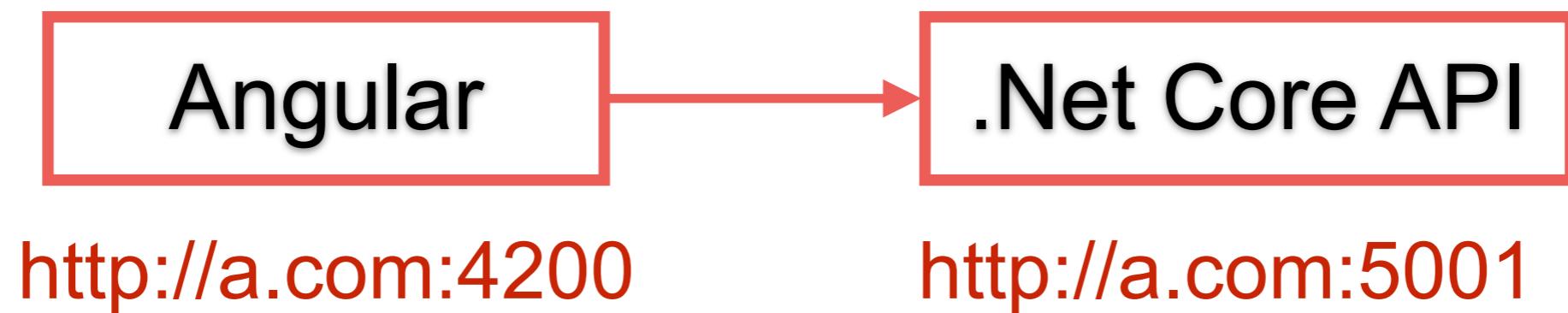
httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
};

 GetAll(): Observable<Weather> {
  return this.http.get<Weather>(this.baseurl + '/weatherforecast')
    .pipe(
      retry(1),
      catchError(this.errorHandler)
    );
}
```

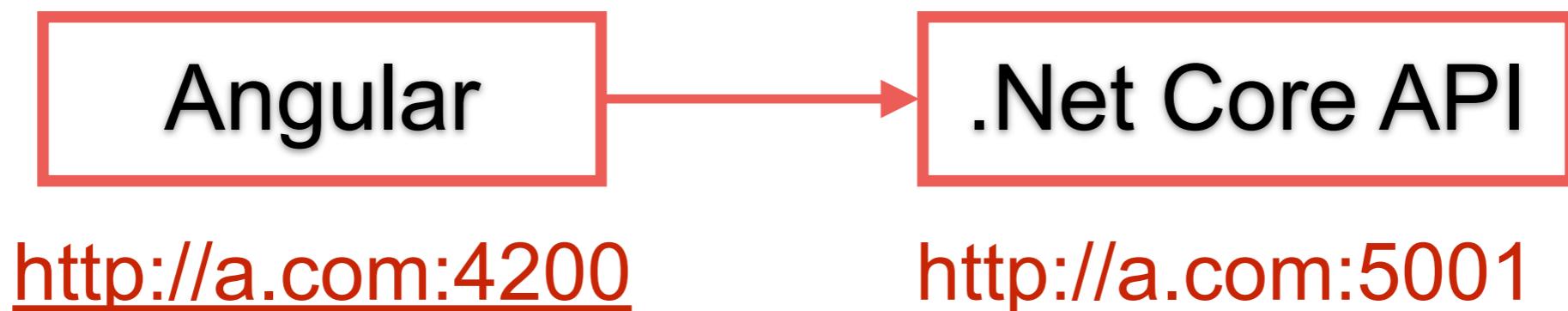
<https://github.com/up1/demo-angular-9-http>



# Call .Net Core API



# Call .Net Core API



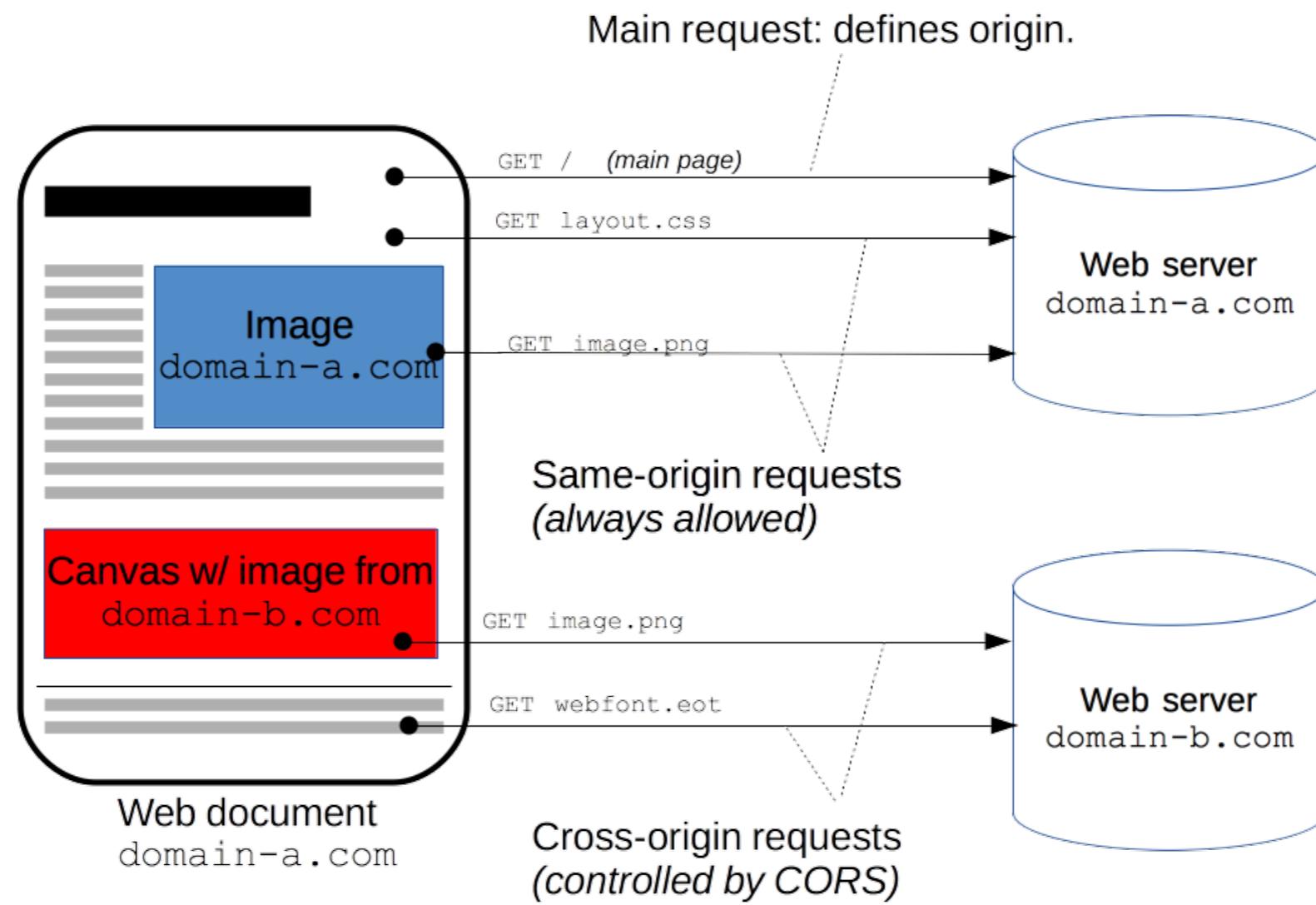
- ✖ Access to XMLHttpRequest at '<https://localhost:5001/weatherforecast>' from origin '<http://localhost:4200>' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

<https://docs.microsoft.com/en-us/aspnet/core/security/cors>



# CORS

## Cross-Origin Resource Sharing



<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



# Using reverse proxy



# Enable CORS in .Net Core

Middleware  
Endpoint routing  
EnableCors attribute

<https://docs.microsoft.com/en-us/aspnet/core/security/cors>



# Add CORS middleware

In Startup.cs

*Allow only from specified source*

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddDefaultPolicy(
            builder =>
            {
                builder.WithOrigins("http://localhost:4200");
            });
    });

    services.AddControllers();
}
```



# Add CORS middleware

## In Startup.cs

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

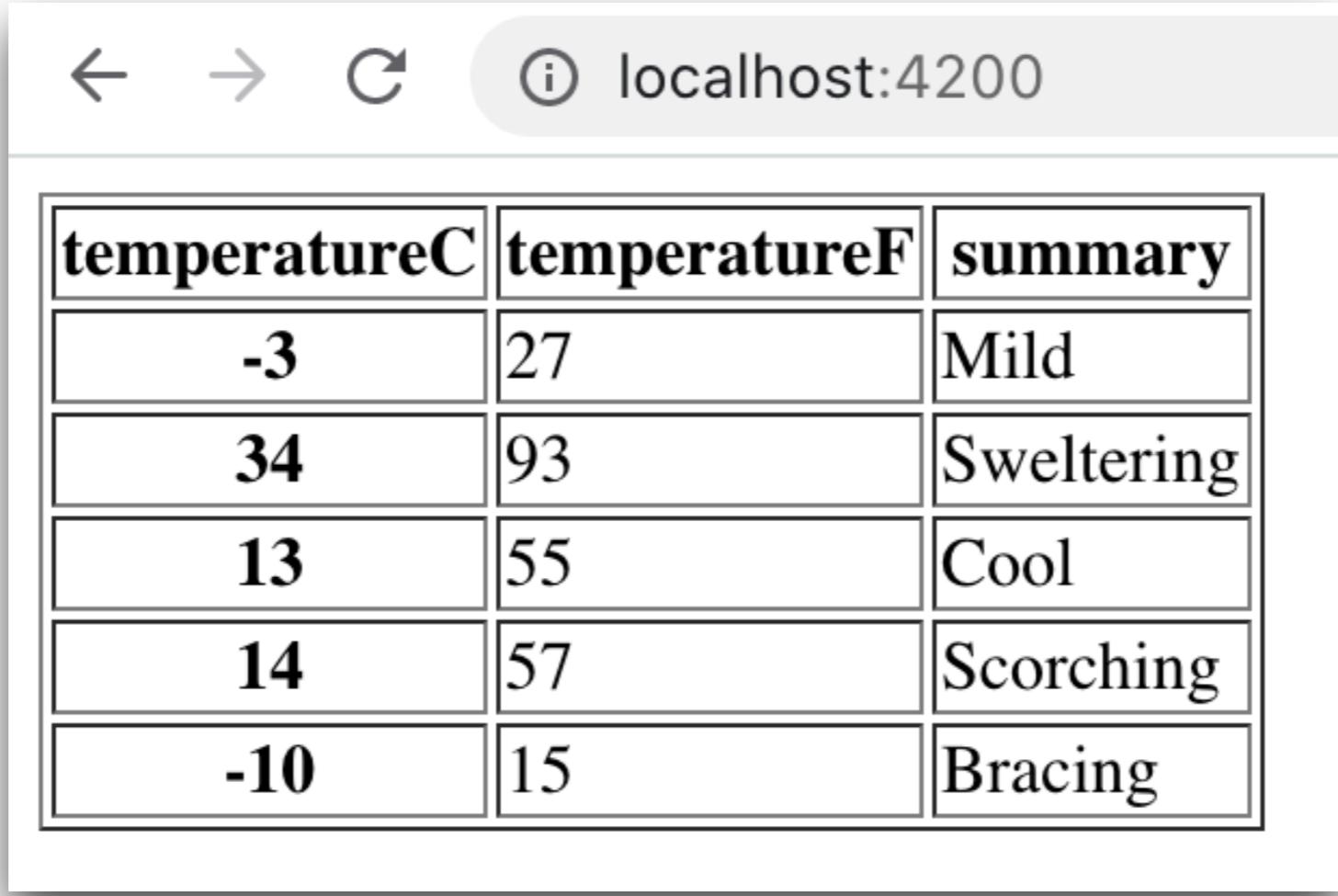
    app.UseHttpsRedirection();

    app.UseRouting();
    app.UseCors();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```



# Restart API and see result



A screenshot of a web browser window titled "localhost:4200". The page displays a table with three columns: "temperatureC", "temperatureF", and "summary". The table has five rows of data. The data is as follows:

temperatureC	temperatureF	summary
-3	27	Mild
34	93	Sweltering
13	55	Cool
14	57	Scorching
-10	15	Bracing

<https://github.com/up1/demo-angular-9-http>



# **Workshop JWT with .Net Core API**

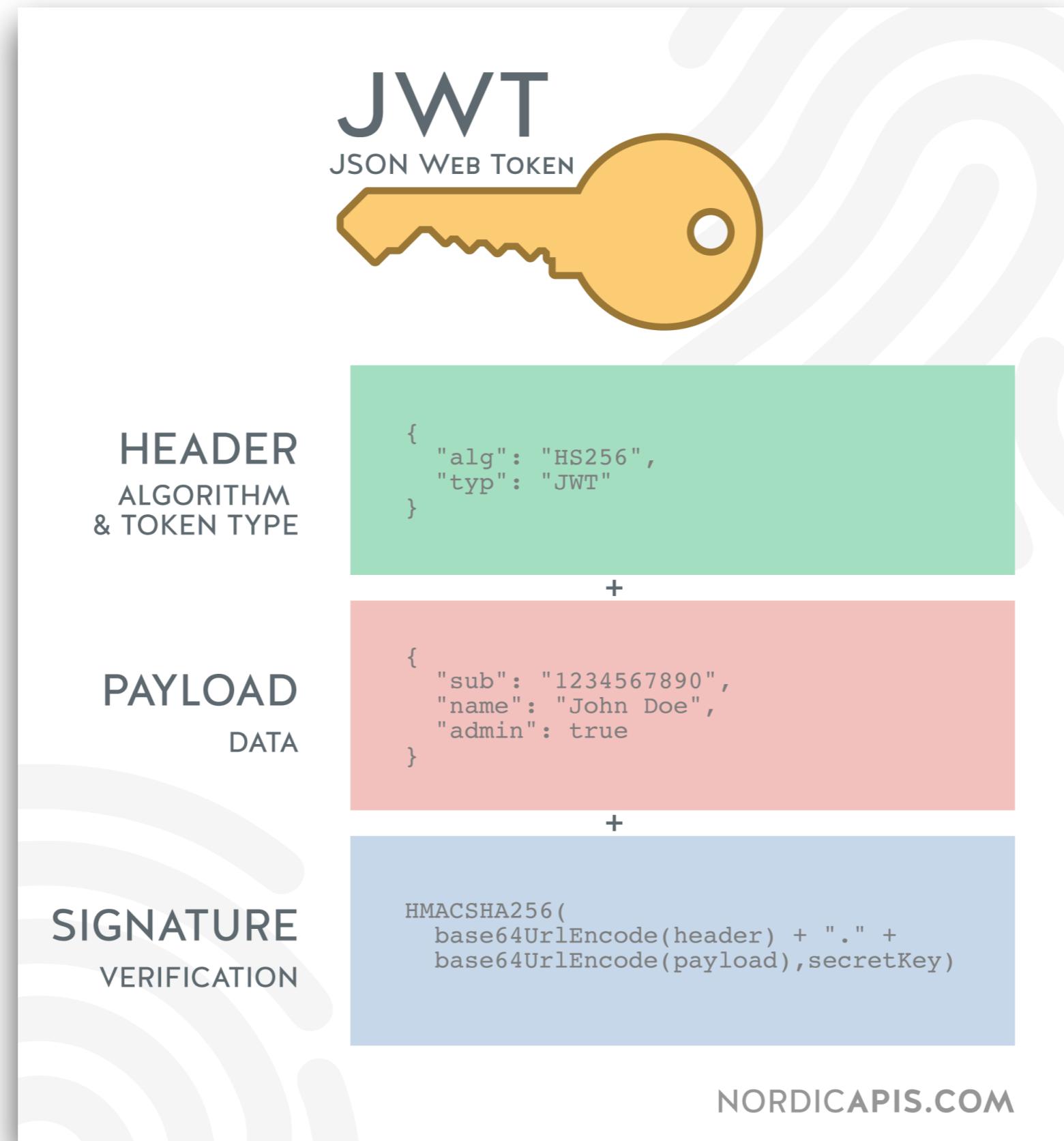


# JWT

**JSON Web Token**  
Every JWT Token has 3 parts  
Header, Payload and Signature

<https://jwt.io/>





# 1. Header

Algorithm use for signing the token  
Type = JWT



## 2. Payload

Contains claims which required by application



# 3. Signature

Encode of header

Encode of payload

Secret

Algorithm in header

*Make sure that token was not changed along  
the way in header*



# Every parts are Based64 URL encoded



ALGORITHM

HS256

## Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Separate by dot (.)

## Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM &amp; TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

✓ Signature Verified

SHARE JWT

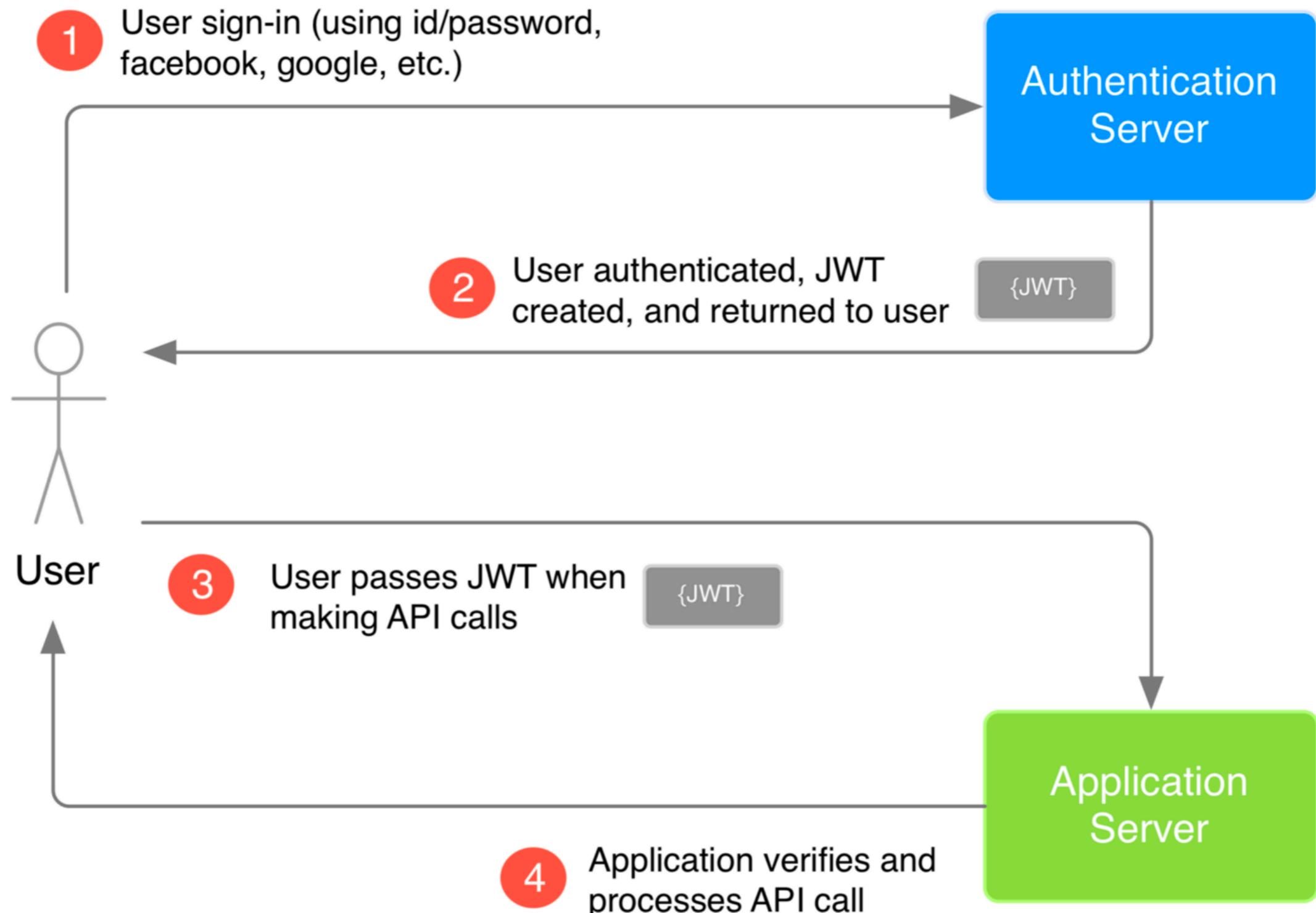
<https://jwt.io/>



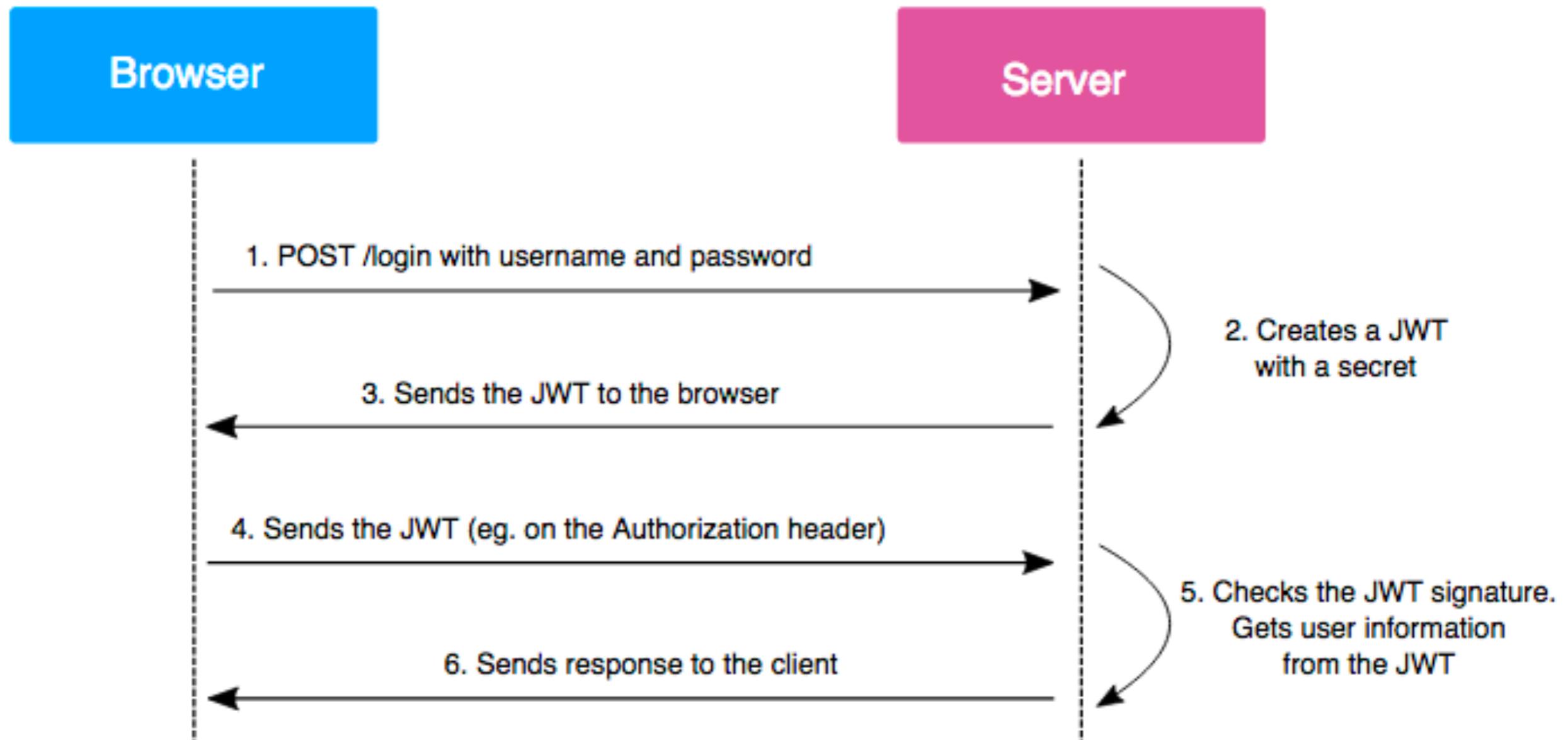
.Net Core

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Flow of JWT

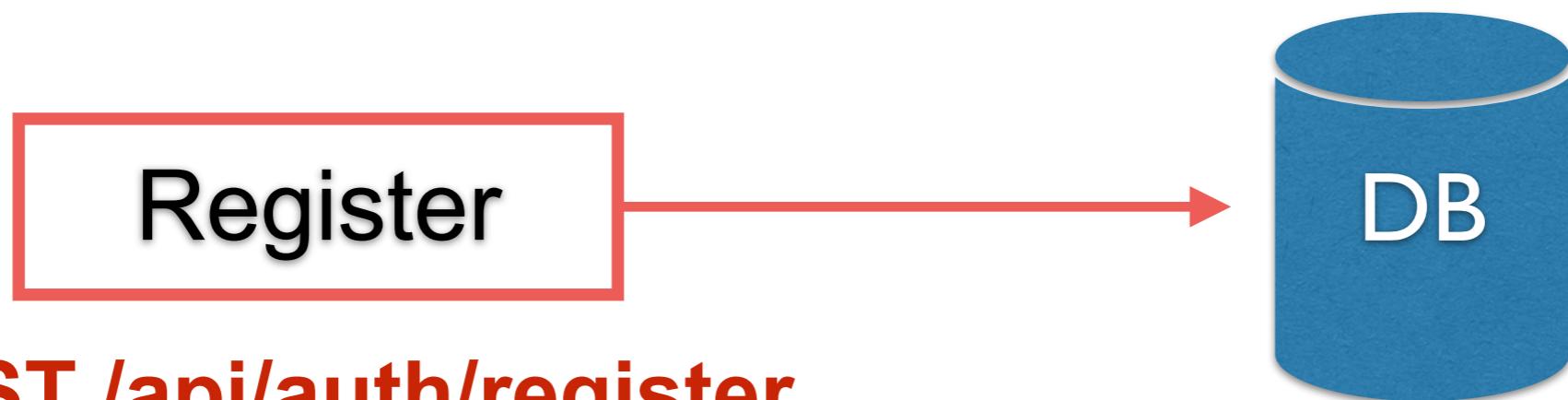


# Flow of JWT



# Step 1

Create/register new user with user and password



**POST /api/auth/register**

*name=demo*

*password=pass*

In-memory database

**Table = AppUser**

*name*

*password*



# Add Package from NuGet

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.InMemory

Microsoft.AspNetCore.Identity.EntityFrameworkCore

System.IdentityModel.Tokens.Jwt

Microsoft.AspNetCore.Authentication.JwtBearer



# Models/AppUser.cs

```
public class AppUser : IdentityUser
{
    public string Name { get; set; }
    public string Password { get; set; }
}
```



# Data/ApplicationDbContext.cs

```
public class ApplicationDbContext : IdentityDbContext<AppUser>
{
    public ApplicationDbContext(
        DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }
}
```



# Startup.cs

## Using In-memory database

```
public void ConfigureServices(IServiceCollection services)
{
    // Use In-memory database
    services.AddDbContext<ApplicationDbContext>(config =>
    {
        config.UseInMemoryDatabase("MemoryBaseDataBase");
    });
}
```



# Startup.cs

## Add Identity and configuration

```
// Add Identity
services.AddIdentity<AppUser, IdentityRole>(config =>
{
    // User defined password policy settings.
    config.Password.RequiredLength = 4;
    config.Password.RequireDigit = false;
    config.Password.RequireNonAlphanumeric = false;
    config.Password.RequireUppercase = false;
})
    .AddEntityFrameworkStores<ApplicationContext>()
    .AddDefaultTokenProviders();
```

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity>



# Startup.cs

## Add JWT configuration

```
// Add JWT token
var jwtSection = Configuration.GetSection("JwtBearerTokenSettings");
services.Configure<JwtBearerTokenSettings>(jwtSection);
var jwtBearerTokenSettings = jwtSection.Get<JwtBearerTokenSettings>();
var key = Encoding.ASCII.GetBytes(jwtBearerTokenSettings.SecretKey);

services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.RequireHttpsMetadata = false;
    options.SaveToken = true;
    options.TokenValidationParameters = new TokenValidationParameters()
    {
        ValidIssuer = jwtBearerTokenSettings.Issuer,
        ValidAudience = jwtBearerTokenSettings.Audience,
        IssuerSigningKey = new SymmetricSecurityKey(key),
    };
});
```



# Generated tables from Identity

Roles  
RolesClaims  
UserClaims  
UserRoles  
UserTokens



# AuthController.cs

## Register new user

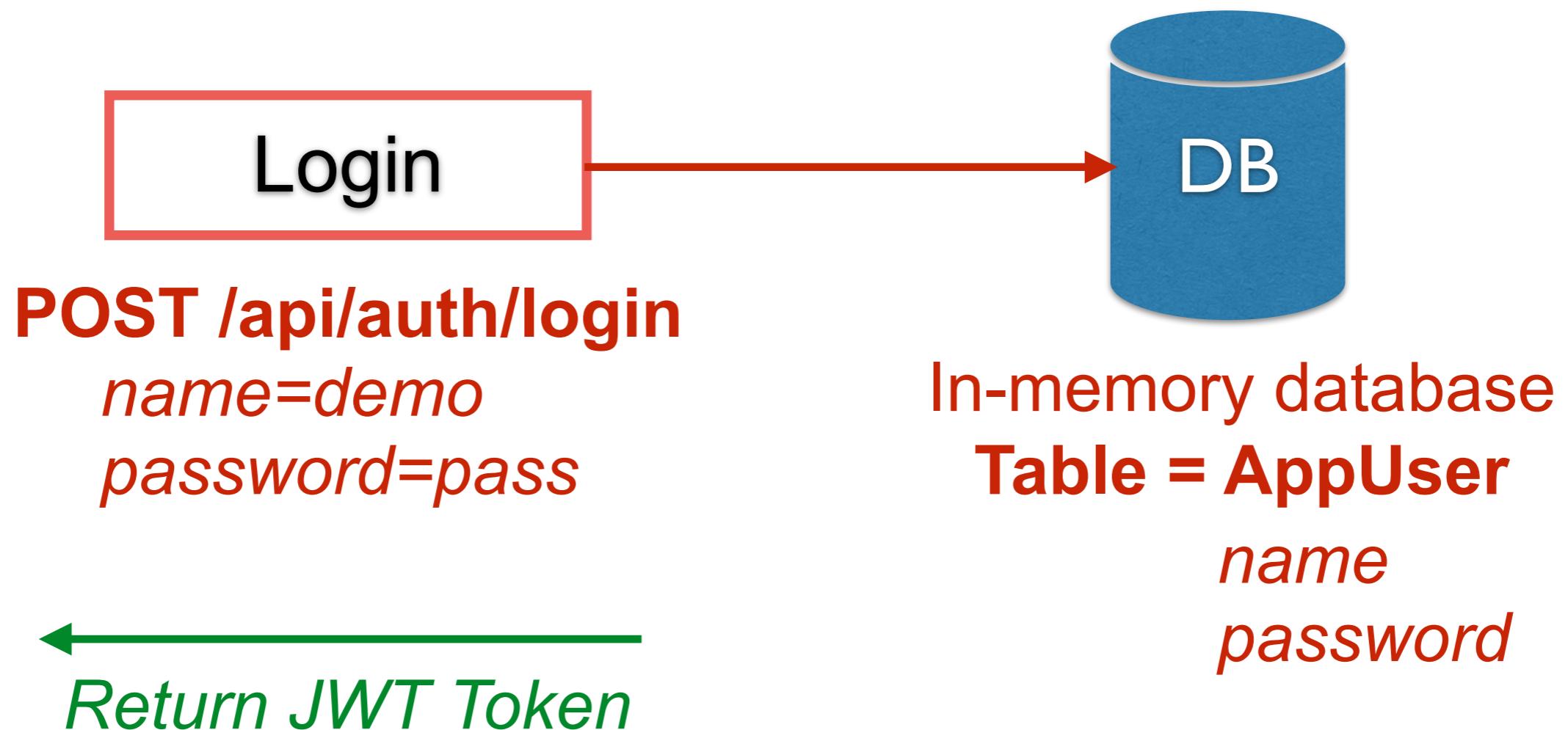
```
[HttpPost]
[Route("register")]
public async Task<IActionResult> Register([FromBody] AppUser appUser)
{
    if (appUser == null)
    {
        return new BadRequestObjectResult(new { Message = "User validation failed" });
    }
    var newUser = new AppUser
    {
        Id = new Random().Next(1, 100).ToString(),
        Name = appUser.Name,
        UserName = appUser.Name,
        Password = appUser.Password,
        Email = appUser.Email
    };

    var result = await userManager.CreateAsync(newUser, appUser.Password);
    if (!result.Succeeded)
    {
        var dictionary = new ModelStateDictionary();
        foreach (IdentityError error in result.Errors)
```



# Step 2

## Login with user and password



# AuthController.cs

## Login with user and password

```
[HttpPost]
[Route("login")]
public async Task<IActionResult> Login([FromBody] AppUser appUser)
{
    if (appUser == null)
    {
        return new BadRequestObjectResult(new { Message = "Login validate failed" });
    }

    var identityUser = await userManager.FindByNameAsync(appUser.Name);
    if(identityUser == null)
    {
        return new BadRequestObjectResult(new { Message = "User not found" });
    }

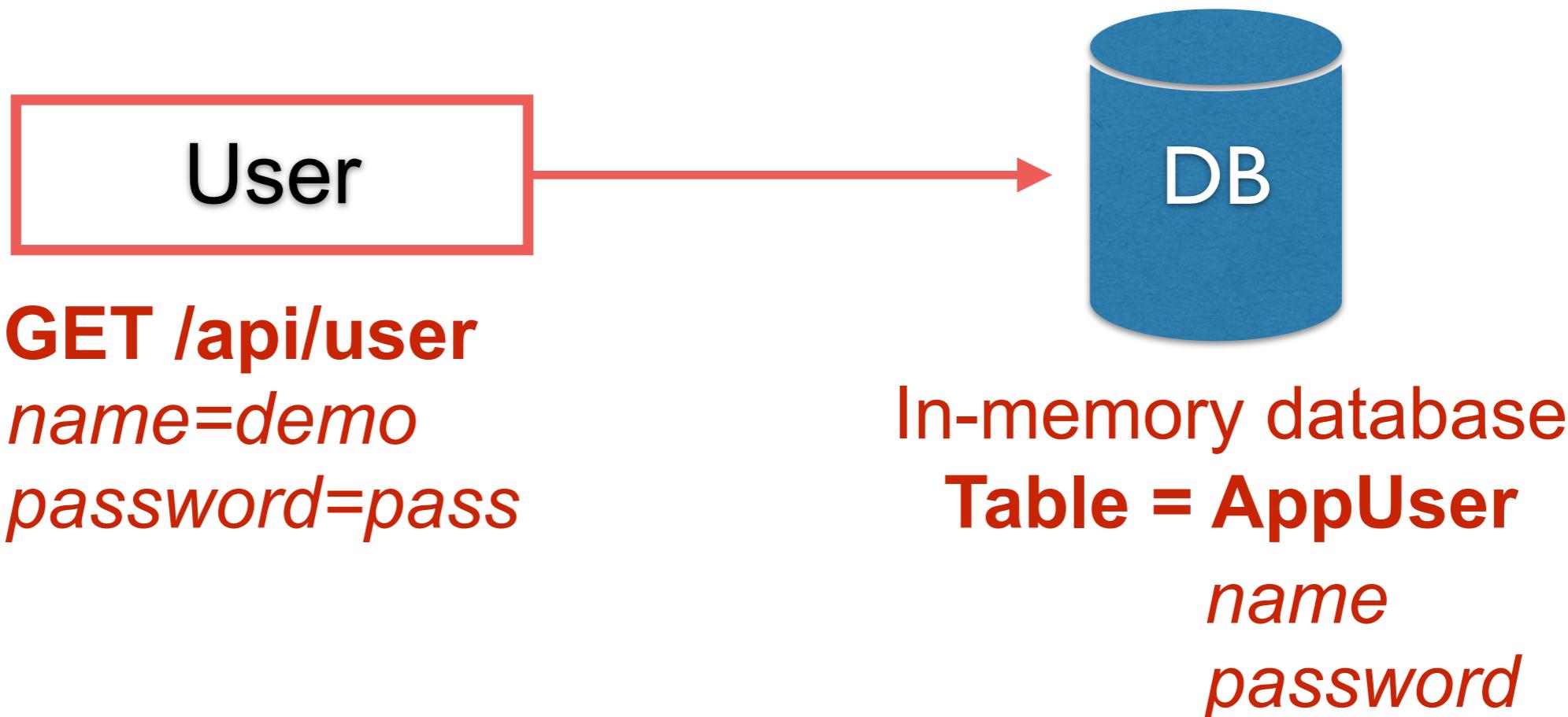
    var result = userManager.PasswordHasher.VerifyHashedPassword(identityUser, identityUser
    if(result == PasswordVerificationResult.Failed)
    {
        return new BadRequestObjectResult(new { Message = "Password incorrect" });
    }

    var token = GenerateToken(identityUser);
    return Ok(new { Token = token, Message = "Success" });
}
```



# Step 3

## Access to API with Token



# Call API with Token

▶ 3. Call service with token after login

Comments 0 Examples 0

GET https://localhost:5001/api/user Send Save

Params Authorization ● Headers (9) Body Pre-request Script Tests Settings Cookies Code

TYPE Token {{data}}

Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body Cookies Headers (4) Test Results Status: 200 OK Time: 7 ms Size: 177 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "message": "Pass authorized"  
3 }
```



# OAuth 2

## Abstract Protocol Flow

