



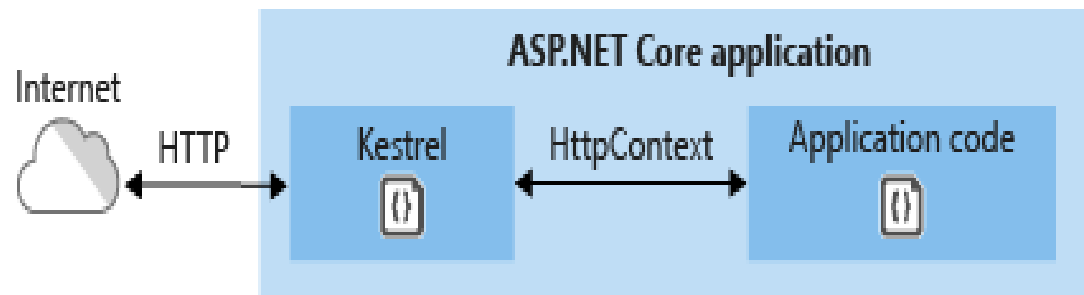
LẬP TRÌNH C# 4

BÀI 2: CONFIGURATION FILE VÀ STATIC FILE

- ⊙ Kestrel Webserver
- ⊙ appsettings.json File
- ⊙ MiddleWare / Pipeline
- ⊙ Static file



- ❑ Kestrel là web server dành cho ASP.NET Core
- ❑ Kestrel là một HTTP web server mã nguồn mở (open source - <https://github.com/aspnet/KestrelHttpServer>)
- ❑ Đa nền tảng (cross-platform-Windows, LINUX and Mac), hướng sự kiện (event-driven) và bất đồng bộ (asynchronous I/O)
- ❑ Được thêm vào mặc định trong ứng dụng ASP.NET Core

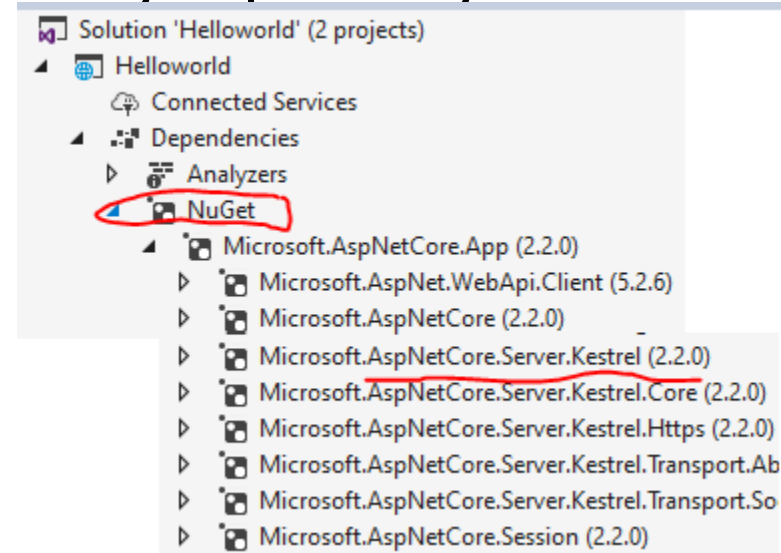


❑ Kestrel mặc định tự có trong Nuget package khi tạo dự án Asp.net core

❑ Phương thức Main gọi đến CreateDefaultBuilder tạo một host cho ứng dụng.

CreateDefaultBuilder đăng ký

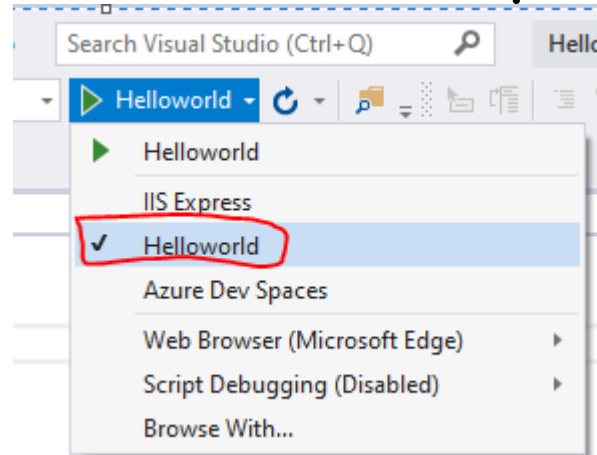
Kestrel như là server sẽ sử dụng trong ứng dụng



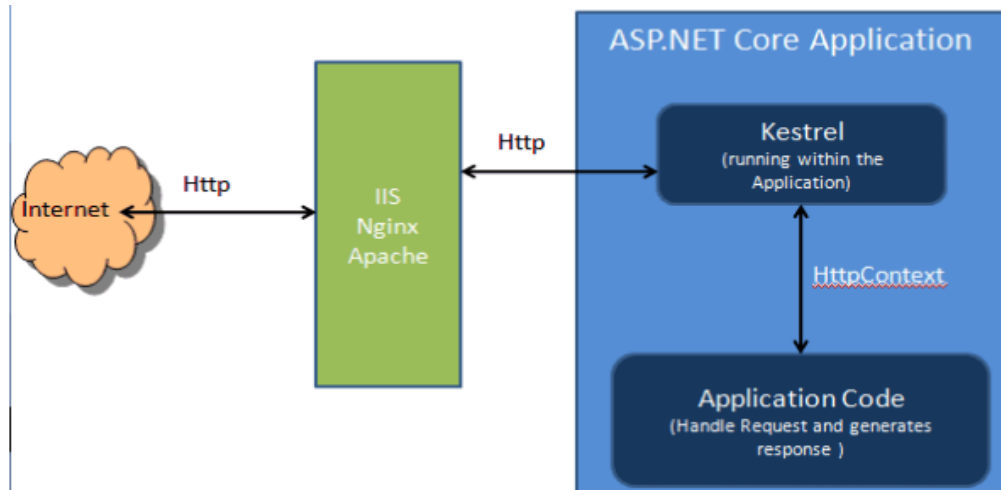
```
public static void Main(string[] args)
{
    BuildWebHost(args).Run();
}

public static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>()
        .Build();
```

- ❑ Mặc định ứng dụng Asp.net core chạy với server IIS, để chạy với Kestrel cần chọn lại server



- ❑ Có thể dùng Kestrel kết hợp với các server khác tăng bảo mật

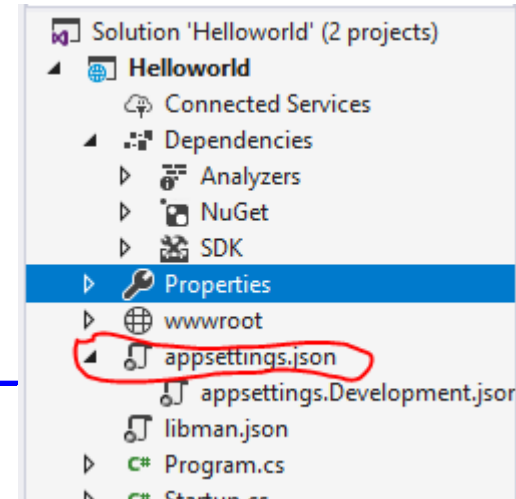


- ❑ Cấu hình là các tham số hoặc các cài đặt cụ thể cung cấp cho ứng dụng khi nó chạy
- ❑ Các cài đặt này được lưu trữ riêng biệt trong code và trong các file độc lập
- ❑ Giúp các developer và quản trị kiểm soát và dễ dàng thay đổi cách mà ứng dụng chạy
- ❑ Ví dụ: Connection Strings cần để kết nối đến cơ sở dữ liệu được lưu trong một file cấu hình. Bằng cách thay đổi chuỗi connection mà bạn có thể thay đổi tên cơ sở dữ liệu, vị trí... mà không cần thay đổi mã nguồn.

- ❑ Ứng dụng ASP.NET core có các file cấu hình có thể lưu ở các định dạng XML, JSON, INI...
- ❑ Mặc định ứng dụng ASP.NET core sẽ nạp file cấu hình JSON lưu ở appsettings.json
- ❑ Sinh viên cần tìm hiểu định dạng của Json

(https://www.w3schools.com/js/js_json)
ví dụ:

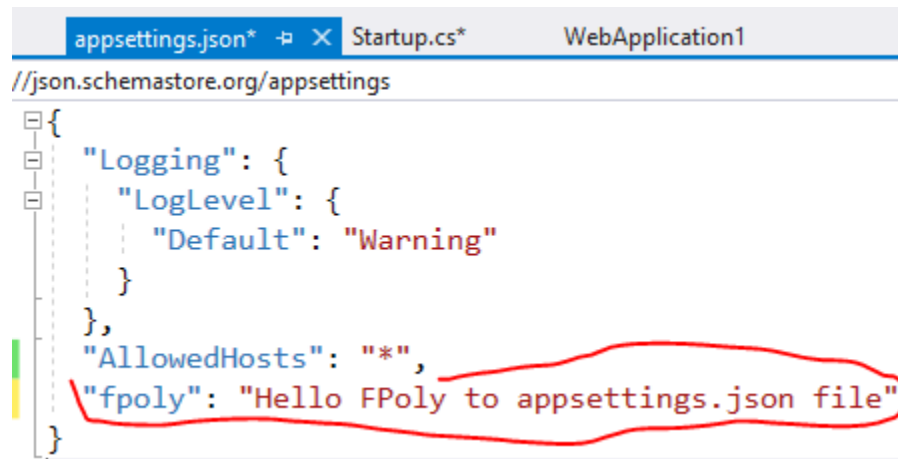
```
{  
  "subsection":  
  {  
    "suboption1": "subvalue1",  
    "suboption2": "subvalue2"  
  }  
}
```



- ❑ ASP.NET Core hỗ trợ đọc cấu hình từ các nguồn khác nhau và các định dạng khác nhau:
 - ❖ Từ file (JSON, INI hoặc XML)
 - ❖ Command line Arguments (tham số dòng lệnh)
 - ❖ Environment variables (biến môi trường)
 - ❖ Custom Provider (cái này là tự tạo ra provider riêng theo ý muốn)
 - ❖ User secrets

❑ Ví dụ đọc nội dung từ appsettings.json

- ❖ Mở appsettings.json và thêm nội dung với key: fpoly và value: "Hello FPoly to appsettings.json file"



```
//json.schemastore.org/appsettings
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "fpoly": "Hello FPoly to appsettings.json file"
}
```

- ❖ Trong Startup.cs tạo một thể hiện từ interface IConfiguration để đọc nội dung từ appsettings.json

```
public class Startup
{
    private IConfiguration _config;
```

❑ Ví dụ đọc nội dung từ appsettings.json

- ❖ Sử dụng thể hiện của interface IConfiguration bên trong constructor lớp Startup.cs

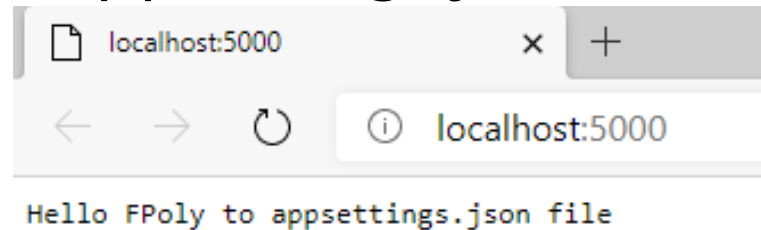
```
public class Startup
{
    private IConfiguration _config;
    // Here we are using Dependency Injection to inject the Configuration object
    0 references | 0 exceptions
    public Startup(IConfiguration config)
    {
        _config = config;
    }
}
```

- ❖ Bên trong phương thức Configure truy xuất đến key:fpoly và dùng context.Response.WriteAsync hiển thị nội dung

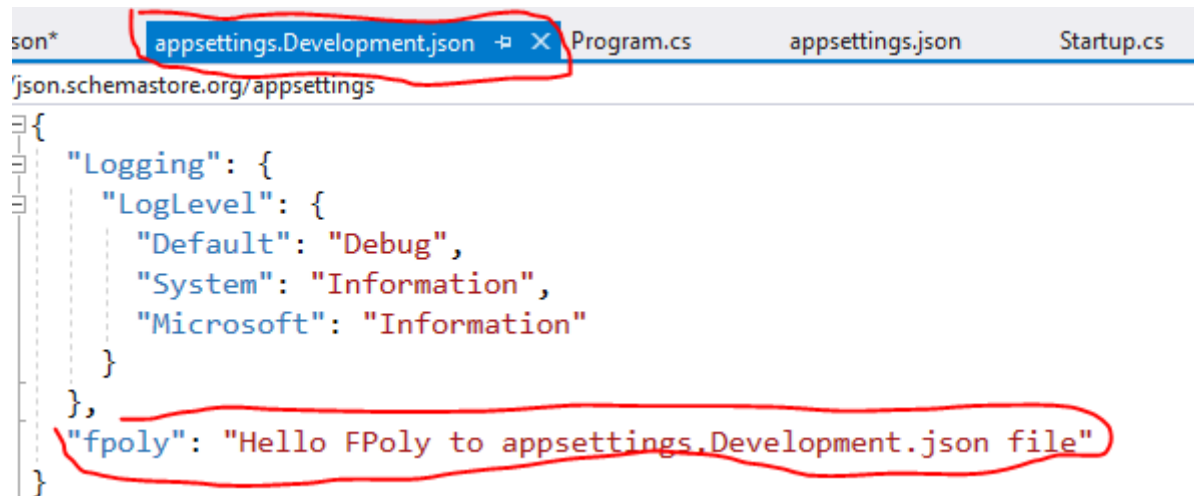
```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync(_config["fpoly"]);
    });
}
```

❑ Ví dụ đọc nội dung từ appsettings.json

❖ Run ứng dụng



❖ Nếu làm việc ở môi trường Development có thể cấu hình trực tiếp trong appsettings.Development.json



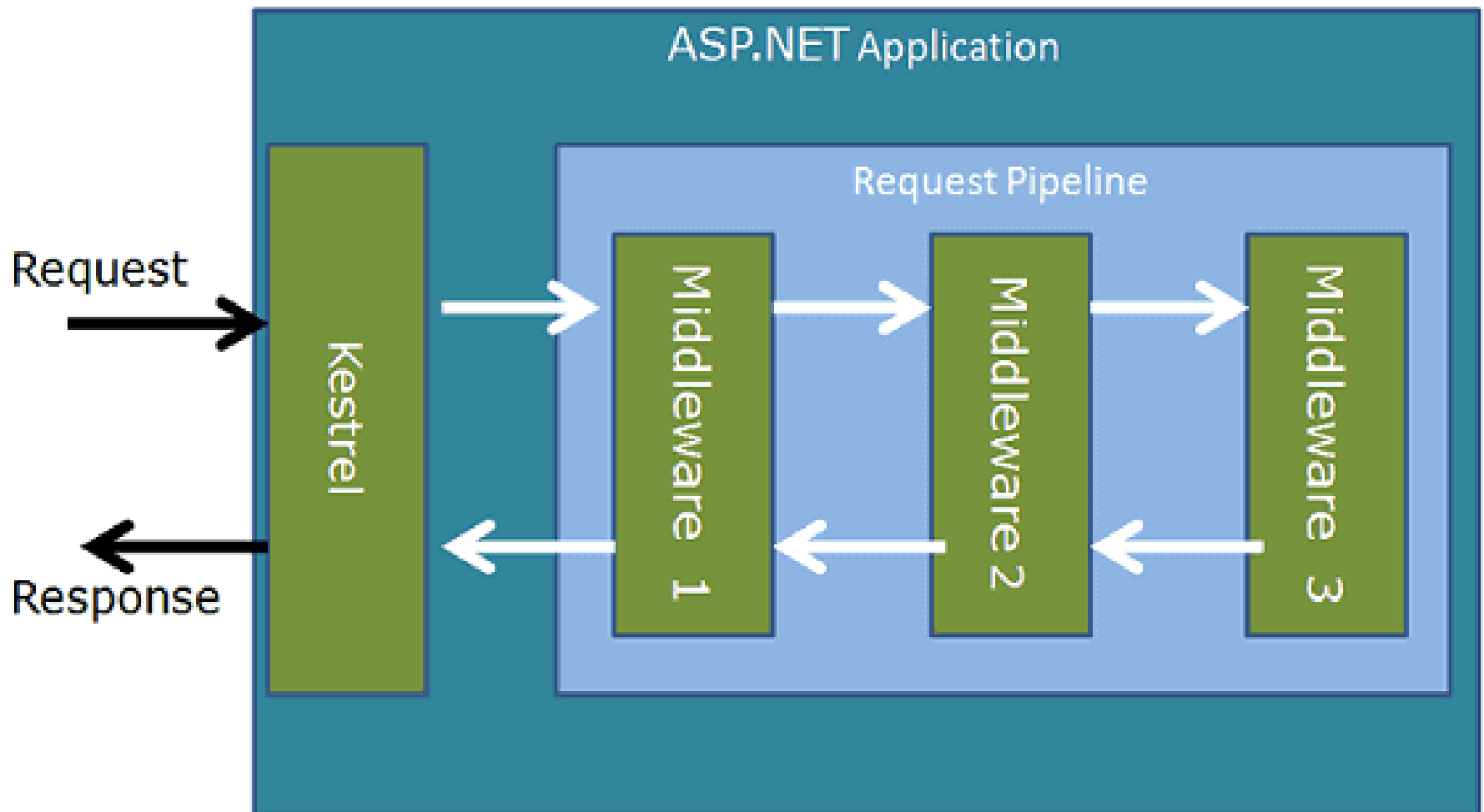
- ❑ Ví dụ đọc đọc mảng từ file cấu hình

```
"wizards": [  
  {  
    "Name": "Gandalf",  
    "Age": "1000"  
  },  
  {  
    "Name": "Harry",  
    "Age": "17"  
  }  
]
```

- ❑ Sử dụng index như là một phần được chia tách trong string bởi dấu hai chấm

```
await context.Response.WriteAsync("<div>" + Configuration.GetSection("wizards:0:Name").Value + "</div>");
```

Middleware in ASP.NET Request Pipeline



- ❑ Một Middleware là một module code nó nhận yêu cầu gửi đến từ Request và trả về Response. Cụ thể trong ASP.NET Core, middlewarre có thể:
 - ❖ Nhận một HTTP Request gửi đến và phát sinh ra HTTP Response để trả về
 - ❖ Nhận một HTTP Request gửi đến, thi hành một số tác vụ (có thể là sửa đổi HTTP Request), sau đó chuyển đến một middleware khác
 - ❖ Nhận HTTP Response, sửa nó và chuyển đến một Middleware khác

- ❑ Pipeline: Trong ứng dụng ASP.NET Core, các middleware kết nối lại với nhau thành một xích, middleware đầu tiên nhận HTTP Request, xử lý nó và có thể chuyển cho middleware tiếp theo hoặc trả về ngay HTTP Response. Chuỗi các middleware theo thứ tự như vậy gọi là pipeline
- ❑ Mỗi Middleware trong pipeline sẽ tuần tự có cơ hội thứ hai để kiểm tra lại request và chỉnh response trước khi được trả lại.
- ❑ Bất cứ middleware nào trong request pipeline đều có thể ngắt request pipeline tại chỗ đó với chỉ một bước đơn giản là không gán request đó đi tiếp

- ❑ Ví dụ trong Startup.cs thì
 - ❖ UseDeveloperExceptionPage() là Middleware
 - ❖ Run() là Middleware

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

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Hello world!");
    });
}
```


- ❑ Để bắt đầu sử dụng Middleware cần thêm nó vào Request Pipeline thông qua phương thức `Configure` trong `Startup.cs`
- ❑ Phương thức **`Configure`** sẽ nhận các thể hiện của **`IApplicationBuilder`**, sử dụng chúng để đăng ký các middleware

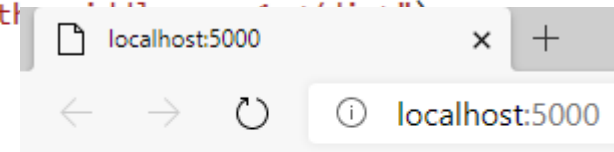
```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 1 </div>");
    });
}
```

- ❖ sử dụng phương thức `app.Run` để đăng ký middleware đầu tiên và hiển thị dòng chữ "Hello world from Middleware 1" khi nó thực thi.

- ❑ Phương thức Run thêm vào một middleware ngắt.
- ❑ Phương thức Use thêm vào middleware, nó sẽ gọi middleware tiếp theo trong pipeline.
- ❑ Chúng ta có thể gọi middleware tiếp theo bằng cách gọi phương thức **Invoke** của middleware tiếp theo

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<div> Hello FPoly, from the middleware 1 </div> ");
        await next.Invoke();//gọi middleware tiếp theo
    });

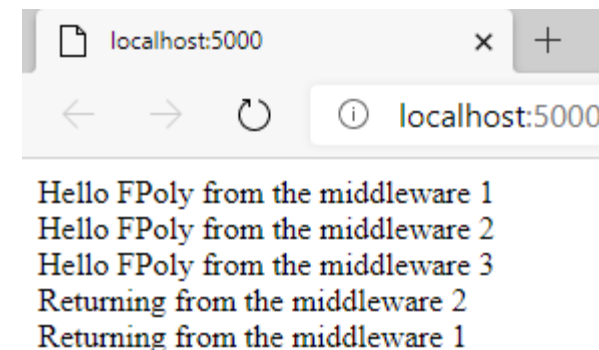
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("<div> Hello FPoly, from the middleware 2 </div> ");
    });
}
```



- Middleware cuối cùng sẽ trả request ngược lại cho middleware trước đó, và sẽ ngắt quá trình trong request pipeline(chạy ngược lại)

```
app.Use(async (context, next) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 1 </div>");
    await next.Invoke();
    await context.Response.WriteAsync("<div> Returning from the middleware 1 </div>");
});
app.Use(async (context, next) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 2 </div>");
    await next.Invoke();
    await context.Response.WriteAsync("<div> Returning from the middleware 2 </div>");
});
app.Run(async (context) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 3 </div>");
});
```

- Thứ tự chạy các middleware 1,2,3 và quay lại middleware 2,1

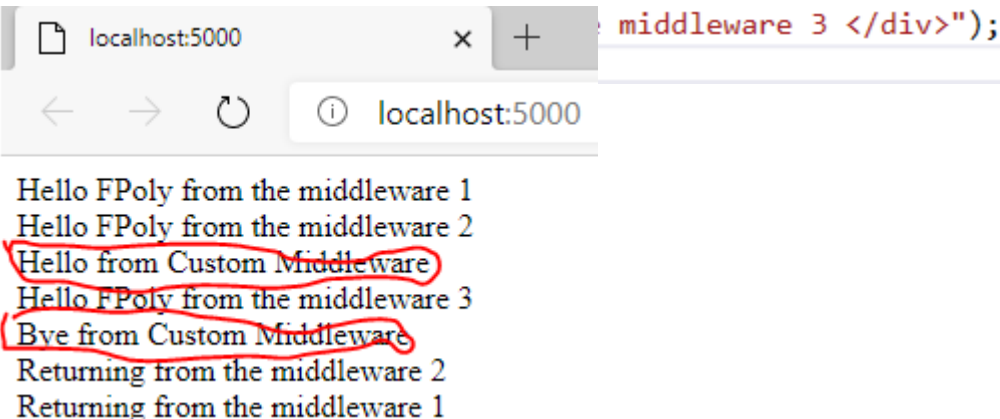


- ❑ Khai báo class chứa các middleware, dùng con trỏ hàm RequestDelegate tham chiếu đến middleware và dùng phương thức Invoke gọi tới một middleware tiếp theo

```
public class CustomMiddleware
{
    private readonly RequestDelegate _next;
    0 references | 0 exceptions
    public CustomMiddleware(RequestDelegate next)
    {
        _next = next; //lấy tham chiếu của middleware tiếp theo trong pipeline lưu vào biến local tên là _next
    }
    0 references | 0 exceptions
    public async Task Invoke(HttpContext context) // phương thức Invoke nhận tham chiếu đến HttpContext
    {
        await context.Response.WriteAsync("<div> Hello from Custom Middleware </div>");
        await _next(context); //gọi middleware tiếp theo
        await context.Response.WriteAsync("<div> Bye from Custom Middleware </div>");
    }
}
```

❑ Đăng ký middleware trong request pipeline sử dụng phương thức UseMiddleware

```
app.Use(async (context, next) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 1 </div>");
    await next.Invoke();
    await context.Response.WriteAsync("<div> Returning from the middleware 1 </div>");
});
app.Use(async (context, next) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 2 </div>");
    await next.Invoke();
    await context.Response.WriteAsync("<div> Returning from the middleware 2 </div>");
});
app.UseMiddleware<CustomMiddleware>();
app.Run(async (context) =>
{
    await context.Response.WriteAsync("<div> Hello FPoly from the middleware 3 </div>");
});
```





DEMO

- Demo cấu hình và dùng file appsettings.json
- Demo Middleware





LẬP TRÌNH C# 4

BÀI 2: CONFIGURATION FILE VÀ STATIC FILE (P2)

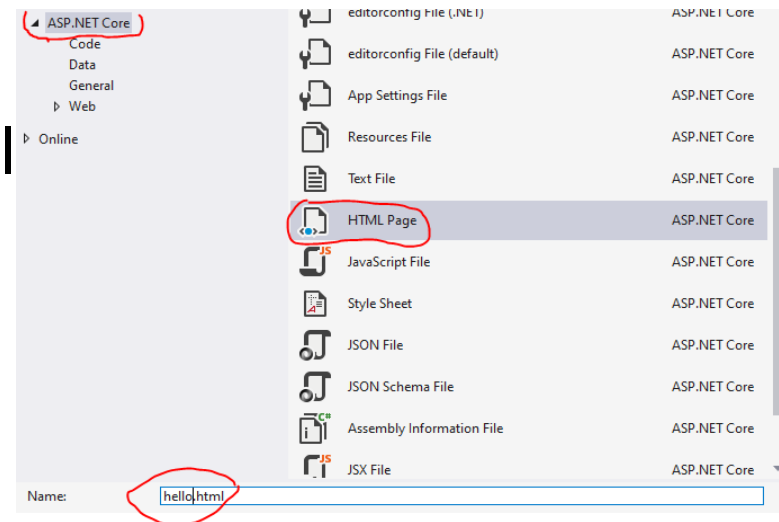
- ❑ Các file như HTML, CSS, ảnh, JavaScript được gọi là file tĩnh hay static files
- ❑ Các file này thường chứa trong thư mục wwwroot của dự án
- ❑ Có hai cách để duyệt các file tĩnh này trong ASP.NET Core
 - ❖ Dùng `app.UseStaticFiles()` bên trong phương thức `Configure` của `Startup.cs`

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    /...
    app.UseStaticFiles(); // dòng code thêm vào để kích hoạt static
    /...
}
```

- ❖ Dùng thông qua Controller MVC (sẽ học trong các bài sau)

❑ Ví dụ tạo trang hello.html trong wwwroot

- ❖ B1: tạo thư mục wwwroot nếu chưa tồn tại
- ❖ B2: tạo trang hello.html
- ❖ B3: Cập nhật nội dung hello.html



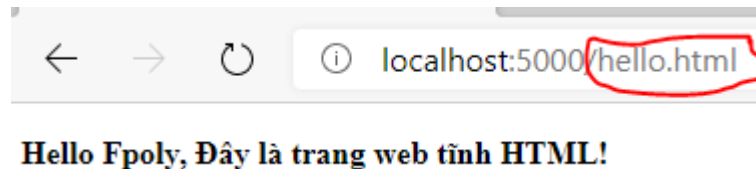
```
Hello.html  Startup.cs
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Fpoly</title>
6  </head>
7  <body>
8      <p><strong>Hello Fpoly, Đây là trang web tĩnh HTML!</strong></p>
9  </body>
10 </html>
```

❑ Ví dụ tạo trang hello.html trong wwwroot

- ❖ B4: Thêm app.UseStaticFiles() bên trong phương thức Configure của Startup.cs

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseStaticFiles();//kích hoạt static file
}
```

- ❖ Run ứng dụng và truy cập vào chính xác tên hello.html



- ❑ Có thể tổ chức thư mục dự án lưu trữ các file css, js, bootstrap trong dự án
- ❑ Ví dụ tạo file SignUp.html liên kết với SignUp.css:

The screenshot shows a web browser window with the address bar displaying 'localhost:5000/SignUp.html#'. The page content includes a red heading 'Hello, Welcome to FPoly SignUp Form', a 'Sign Up' section with the instruction 'Please fill in this form to create an account.', and three input fields for 'Email', 'Password', and 'Repeat Password'. There is a 'Remember me' checkbox and a link to 'Terms & Privacy'. At the bottom, there are two buttons: 'Cancel' (red) and 'Sign Up' (green).

localhost:5000/SignUp.html#

localhost:5000/SignUp.html#

Hello, Welcome to FPoly SignUp Form

Sign Up

Please fill in this form to create an account.

Email

Password

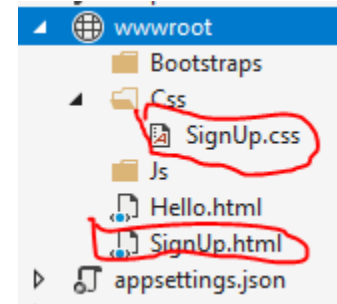
Repeat Password

☒ Remember me

By creating an account you agree to our [Terms & Privacy](#).

Cancel Sign Up

❑ Tổ chức thư mục dự án

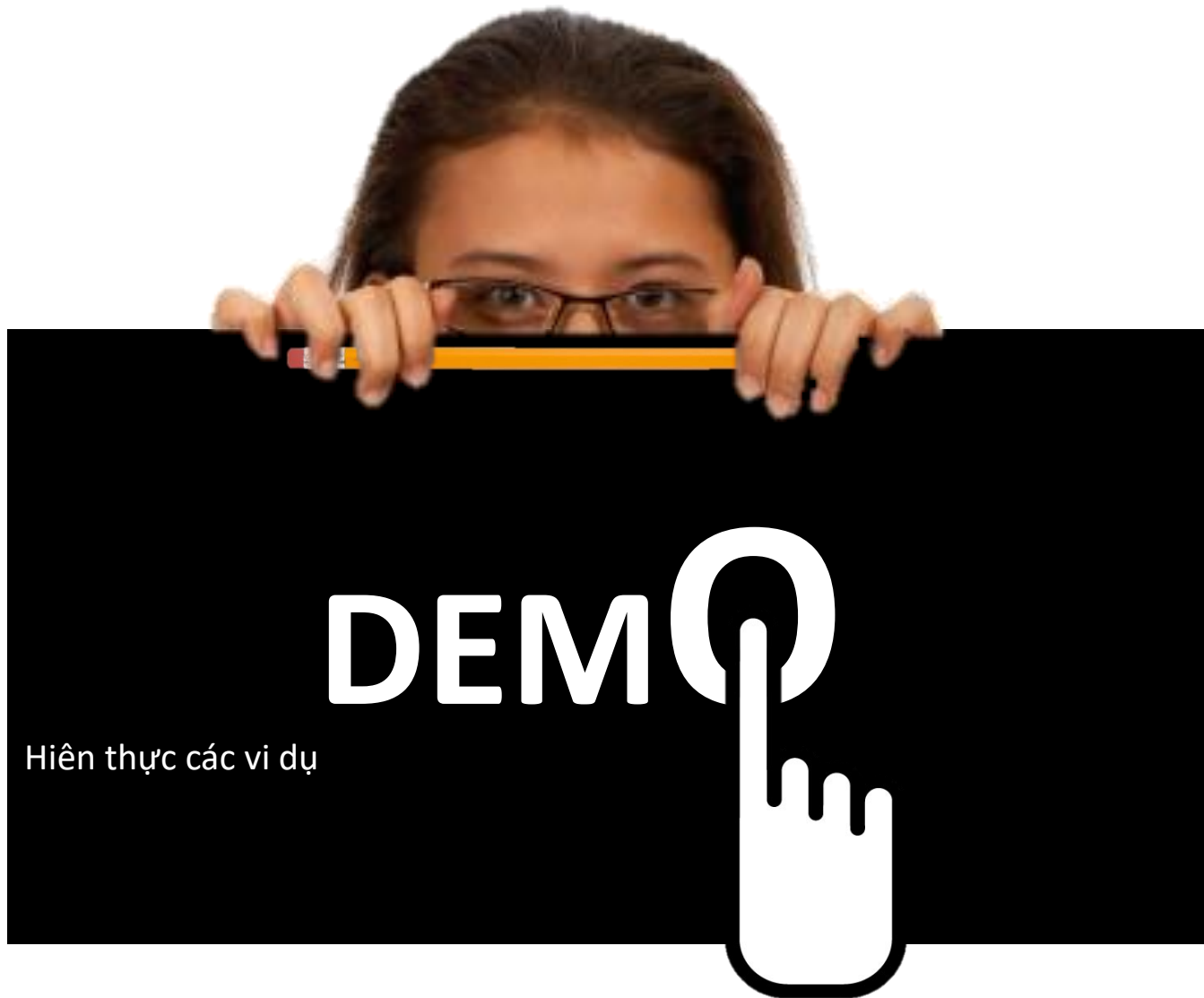


❑ Viết code file SignUp.html

```
<!DOCTYPE html>
<style>
  <!-- Viết Css tại đây -->
</style>
<html>
<head>
  <link href="Css/SignUp.css" rel="stylesheet" /> <!--Liên kết file css-->
</head>
<body>
  <h2 style="text-align:center; color:red">Hello, Welcome to FPoly SignUp Form</h2>
  <form action="/action_page.php" style="border:1px solid #ccc">
    <div class="container">
      <h1>Sign Up</h1>
      <p>Please fill in this form to create an account.</p>
      <hr>
    </div>
  </form>
  <!--Viết thêm code hoàn thiện form SignUp-->
```

Code tham khảo cho SignUp.css

```
SignUp.css*  ↵  ✕  
1  /* Full-width input fields */  
2  input[type=text], input[type=password] {  
3      width: 100%;  
4      padding: 15px;  
5      margin: 5px 0 22px 0;  
6      display: inline-block;  
7      border: none;  
8      background: #f1f1f1;  
9  }  
10  
11  input[type=text]:focus, input[type=password]:focus {  
12      background-color: #ddd;  
13      outline: none;  
14  }  
15  <!--Viết thêm code hoàn thiện-->
```



Tổng kết bài học

- ◎Kestrel Webserver
- ◎appsettings.json File
- ◎MiddleWare / Pipeline
- ◎Static file





KẾT THÚC