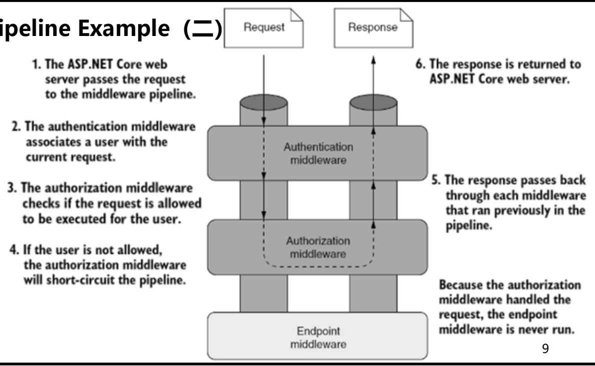
## Pipeline Example (二)


Request
Response
1. The ASP.NET Core web server passes the request to the middleware pipeline.
2. The authentication middleware associates a user with the current request.
3. The authorization middleware checks if the request is allowed to be executed for the user.
4. If the user is not allowed, the authorization middleware will short-circuit the pipeline.
Authentication middleware
Authorization middleware
Endpoint middleware
6. The response is returned to ASP.NET Core web server.
5. The response passes back through each middleware that ran previously in the pipeline.
Because the authorization middleware handled the request, the endpoint middleware is never run.

## Understanding Middleware Pipeline (一)

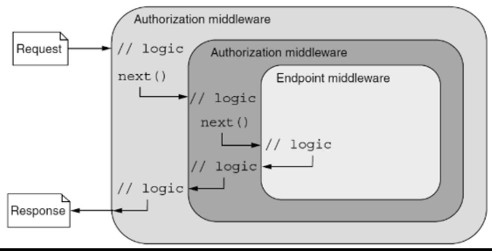
■One of the most common use cases for middleware is for the **cross-cutting concerns** of your application.
■These aspects need to occur for every request:
- ■Logging each request
- ■Adding standard security headers to the response
- ■Associating a request with the relevant user
- ■Setting the language for the current request

## Understanding Middleware Pipeline (二)

■The middleware pipeline implements the **chain-of-responsibility** design pattern.
■The pipeline is **bidirectional**.
■When a middleware component short-circuits the pipeline and returns a response, it's called **terminal middleware**.
■Requests are passed to the middleware pipeline as **HttpContext** objects.

## Understanding Middleware Pipeline (三)

■You can think of the pipeline as being a series of nested middleware.



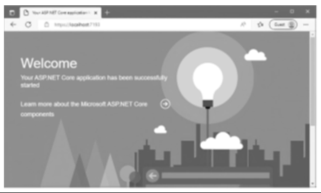
## How to Combine Middleware in a Pipeline?

■To build a complete application, compose multiple middleware components into a pipeline.
■Microsoft ships many standard middleware components with ASP.NET Core.
■Call Use* methods to add middlewares to the pipeline.

## Example: A Holding Page (一)

CodeLab 2.1

■Provide a sample HTML page no matter what the request is.
■It is useful occasionally when you're setting up a application to ensure that it's processing requests without errors.


Welcome

Your ASP.NET Core application has been successfully started

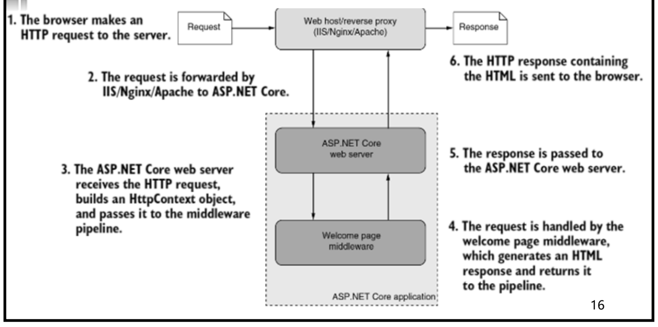
Learn more about the Microsoft ASP.NET Core components

## Example: A Holding Page（二）

- **WelcomePageMiddleware** is designed to provide such an HTML page.
- You will create one of the simplest middleware pipelines, consisting only one middleware.

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);  ┐ Uses the default
WebApplication app = builder.Build();                                ┘ WebApplication
                                                                       configuration
app.UseWelcomePage();  ◁—— The only custom middleware in the pipeline

app.Run();  ◁—— Runs the application to handle requests
```

## Example: A Holding Page（三）


1. The browser makes an HTTP request to the server.
Request
Web host/reverse proxy (IIS/Nginx/Apache)
Response
2. The request is forwarded by IIS/Nginx/Apache to ASP.NET Core.
6. The HTTP response containing the HTML is sent to the browser.
3. The ASP.NET Core web server receives the HTTP request, builds an HttpContext object, and passes it to the middleware pipeline.
ASP.NET Core web server
5. The response is passed to the ASP.NET Core web server.
Welcome page middleware
4. The request is handled by the welcome page middleware, which generates an HTML response and returns it to the pipeline.
ASP.NET Core application

## Example: Handling Static File（一）

CodeLab 2.1

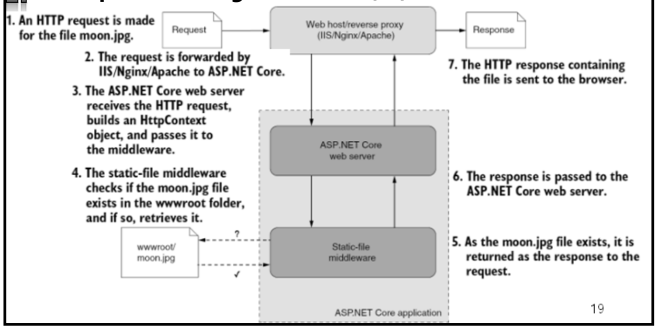
- Most web applications, including those with dynamic content, serve some pages by using static files.
- Images, JavaScript, and CSS stylesheets are normally saved to disk during development and are served up when requested from the special wwwroot folder of your project, normally as part of a full HTML page request.
- By default, the **wwwroot** folder is the only folder in your application that ASP.NET Core will serve files from.

## Example: Handling Static File（二）

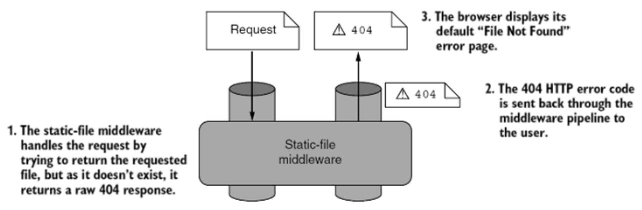
- Use StaticFileMiddleware to serve static files from the wwwroot folder.

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);
WebApplication app = builder.Build();

app.UseStaticFiles();  ◁—— Adds the StaticFileMiddleware to the pipeline

app.Run();
```

## Example: Handling Static File（三）


1. An HTTP request is made for the file moon.jpg.
Request
Web host/reverse proxy (IIS/Nginx/Apache)
Response
2. The request is forwarded by IIS/Nginx/Apache to ASP.NET Core.
7. The HTTP response containing the file is sent to the browser.
3. The ASP.NET Core web server receives the HTTP request, builds an HttpContext object, and passes it to the middleware.
ASP.NET Core web server
4. The static-file middleware checks if the moon.jpg file exists in the wwwroot folder, and if so, retrieves it.
6. The response is passed to the ASP.NET Core web server.
wwwroot/ moon.jpg
?
✓
Static-file middleware
5. As the moon.jpg file exists, it is returned as the response to the request.
ASP.NET Core application

## Example: Handling Static File（四）

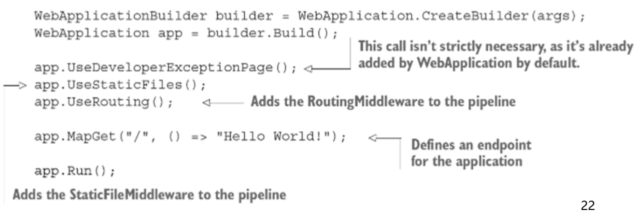

Request
⚠ 404
3. The browser displays its default "File Not Found" error page.
⚠ 404
2. The 404 HTTP error code is sent back through the middleware pipeline to the user.
1. The static-file middleware handles the request by trying to return the requested file, but as it doesn't exist, it returns a raw 404 response.
Static-file middleware

## Example: A Minimal API Application（一）

- Create an application with four pieces of middleware:
  - routing middleware: choose a minimal API endpoint to execute,
  - endpoint middleware: generate the response,
  - static-file middleware: serve any image files from the wwwroot folder,
  - exception-handler middleware: handle any errors that might occur.

## Example: A Minimal API Application（二）

- WebApplication automatically adds some middleware to the pipeline, such as the EndpointMiddleware.



```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);
WebApplication app = builder.Build();

app.UseDeveloperExceptionPage();
app.UseStaticFiles();
app.UseRouting();

app.MapGet("/", () => "Hello World!");

app.Run();
```

This call isn't strictly necessary, as it's already added by WebApplication by default.

Adds the RoutingMiddleware to the pipeline

Defines an endpoint for the application

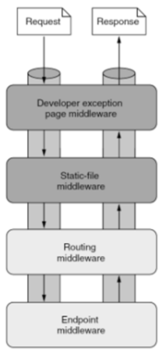
Adds the StaticFileMiddleware to the pipeline

## Example: A Minimal API Application（三）

- MapGet defines an **endpoint**, not middleware.
- It defines the endpoints that the routing and endpoint middleware can use.
- These endpoints are used by the routing and endpoint middleware.

```
app.MapGet("/", () => "Hello World!");
```

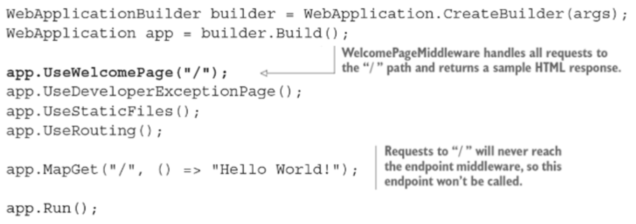
## Example: A Minimal API Application（四）



## Example: A Minimal API Application（五）

Exercise

- What is the behavior of the following pipeline?



```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);
WebApplication app = builder.Build();

app.UseWelcomePage("/");
app.UseDeveloperExceptionPage();
app.UseStaticFiles();
app.UseRouting();

app.MapGet("/", () => "Hello World!");

app.Run();
```

WelcomePageMiddleware handles all requests to the "/" path and returns a sample HTML response.

Requests to "/" will never reach the endpoint middleware, so this endpoint won't be called.

## Example: A Minimal API Application（六）

- You should always consider the order of middleware when adding it to WebApplication.
- Middleware added earlier in the pipeline will run (and potentially return a response) before middleware added later.

## Summary

- Middleware consists of small components that execute in sequence when the application receives an HTTP request.
- They can perform:
  - logging
  - identifying the current user for a request
  - serving static files
  - handling errors
- Add middleware to the pipeline.