HANDLING REQUESTS WITH THE MIDDLEWARE PIPELINE

Kamal Beydoun

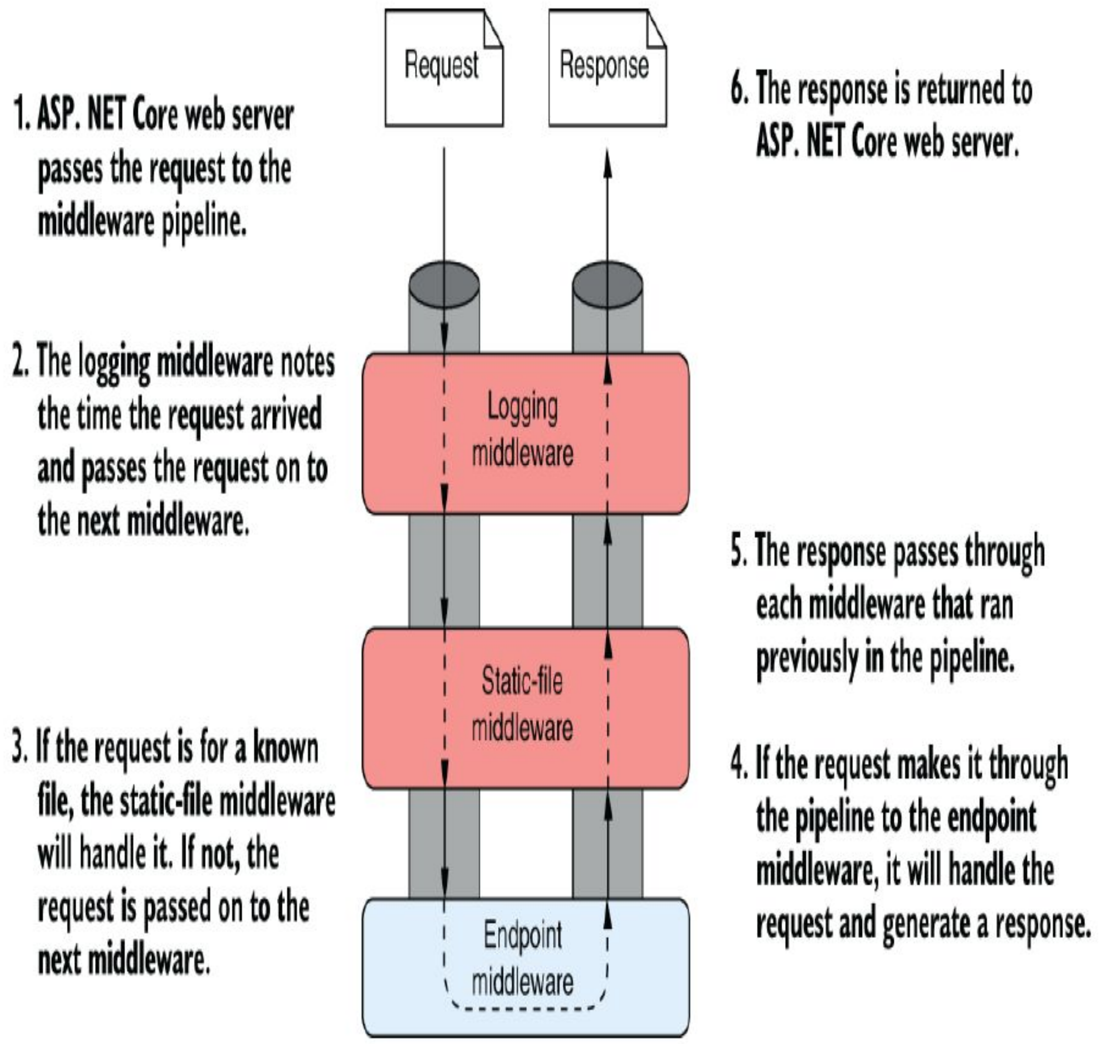
Lebanese University – Faculty of Sciences I

Kamal.beydoun@ul.edu.lb

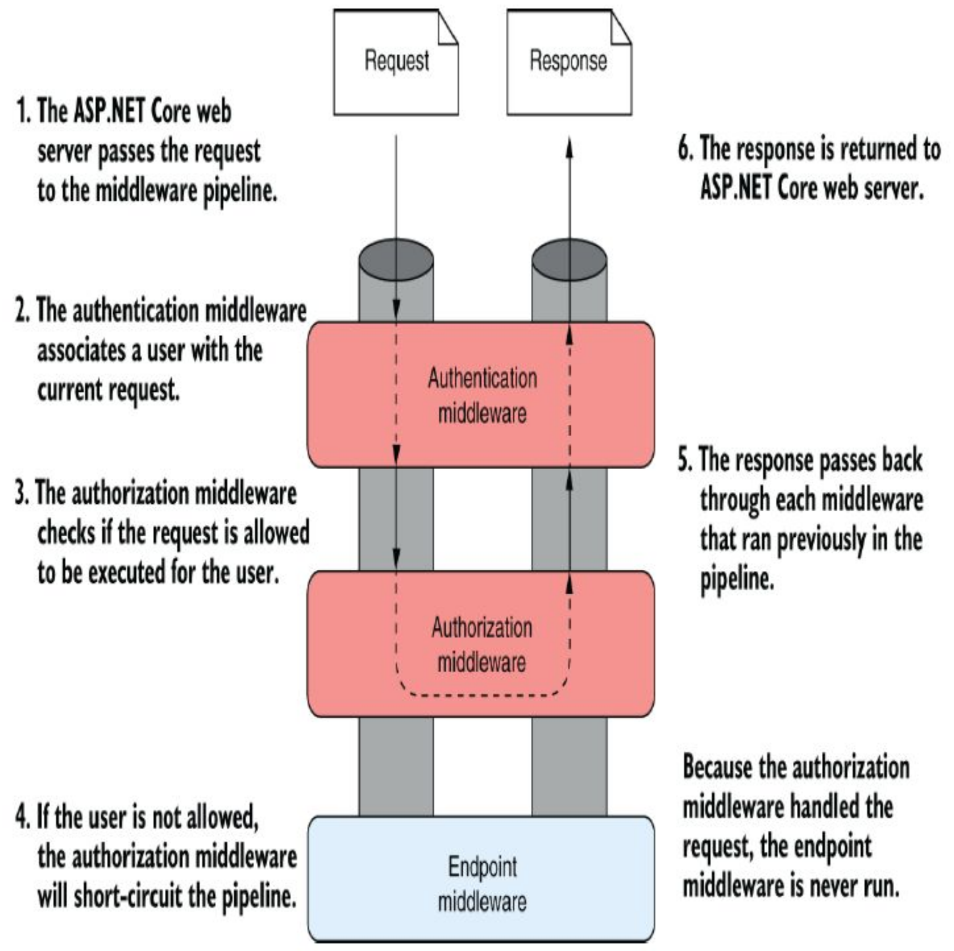
DEFINING MIDDLEWARE

- **ASP.NET Core Middleware:**
  - Comprises C# classes or functions managing HTTP requests or responses.
  - Chained together, creating a pipeline.
  - **Output** of one middleware serves as the **input** for the next in the sequence.
- **ASP.NET Core Middleware can**:
  - Handle an incoming HTTP request by generating an HTTP response
  - Process an incoming HTTP request, modify it, and pass it on to another piece of middleware
  - Process an outgoing HTTP response, modify it, and pass it on to another piece of middleware or to the ASP.NET Core web server

DEFINING MIDDLEWARE


Request
Response
1. ASP. NET Core web server passes the request to the middleware pipeline.
2. The logging middleware notes the time the request arrived and passes the request on to the next middleware.
3. If the request is for a known file, the static-file middleware will handle it. If not, the request is passed on to the next middleware.
Logging middleware
Static-file middleware
Endpoint middleware
6. The response is returned to ASP. NET Core web server.
5. The response passes through each middleware that ran previously in the pipeline.
4. If the request makes it through the pipeline to the endpoint middleware, it will handle the request and generate a response.

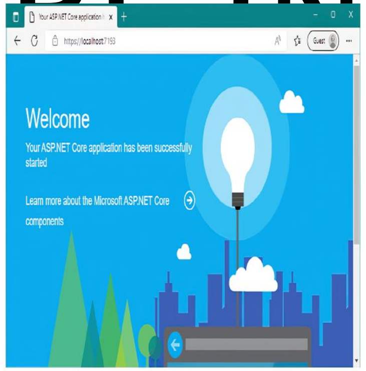
DEFINING MIDDLEWARE


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

HTTPCONTEXT OBJECT

- **Construction by Web Server:**
  - For each **request**, ASP.NET Core web server builds an **HttpContext**.
  - Serves as a storage box for individual requests.
- **Contents of HttpContext:**
  - Properties of the request, request-specific services, loaded data, and errors.
- **Middleware Access to HttpContext:**
  - All middleware has access to the HttpContext for a request.
  - Utilizes this object to examine user credentials, identify requested pages, and retrieve posted data.
- **Processing and Updating HttpContext:**
  - Application processes the request and updates the HttpContext with an appropriate response.
  - Returned through the middleware pipeline to the web server.
- **Response Handling:**
  - ASP.NET Core web server converts the updated HttpContext to a raw HTTP response.
  - Sent back to the reverse proxy, which forwards it to the user's browser.

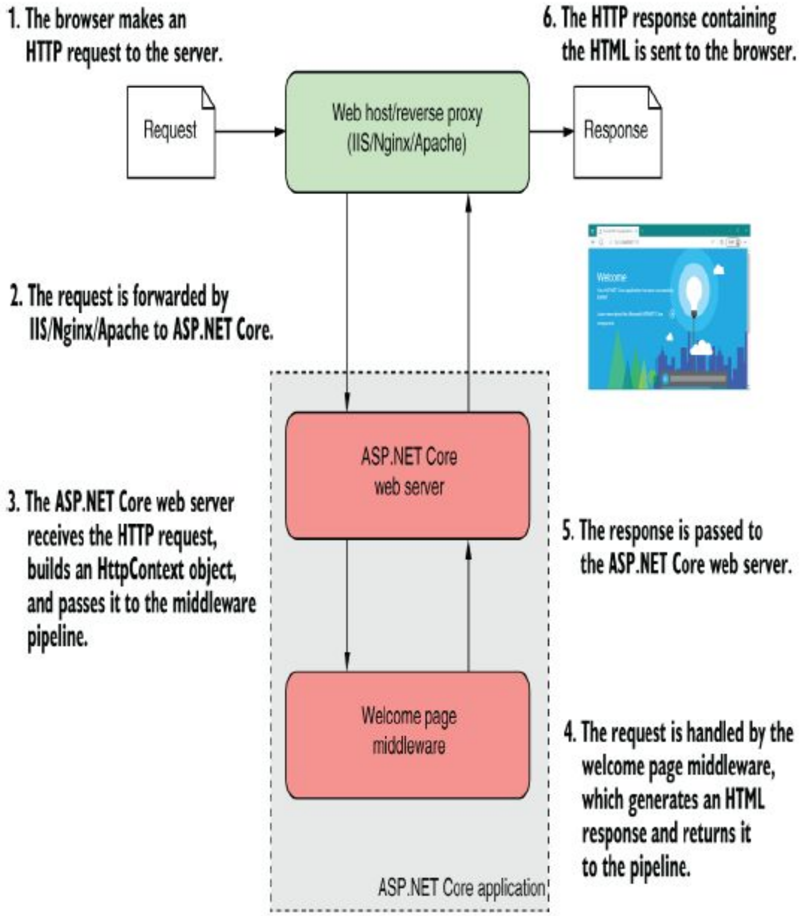
COMBINING MIDDLEWARE IN A PIPELINE – A HOLDING



Welcome

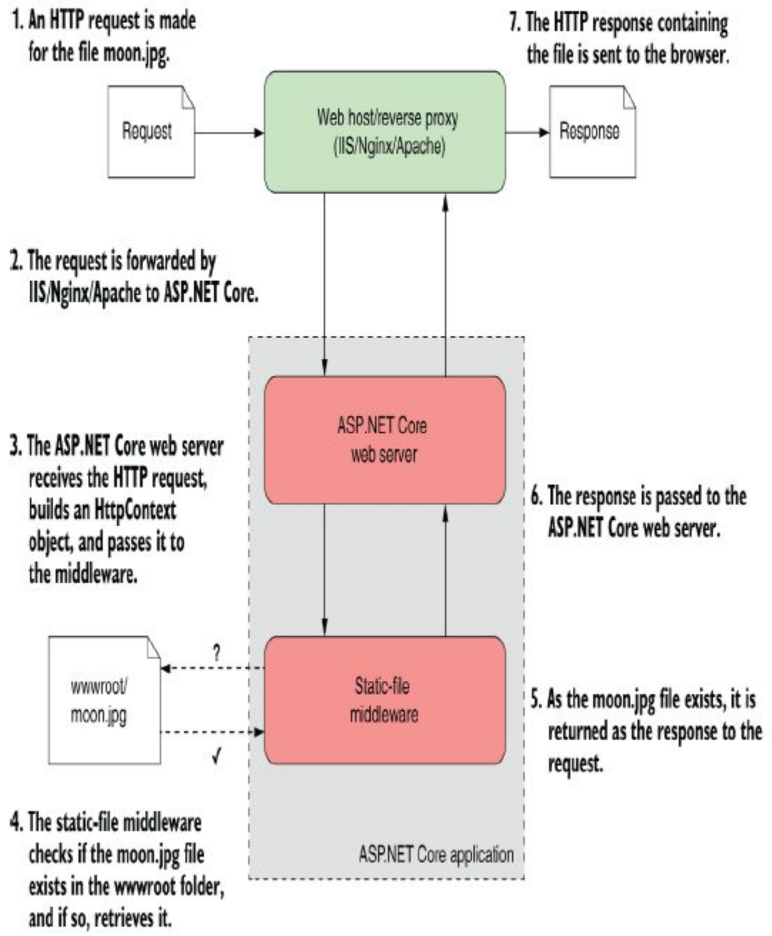
Your ASP.NET Core application has been successfully started

Learn more about the Microsoft ASP.NET Core components


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

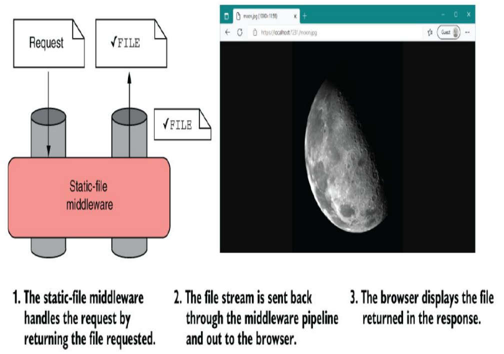
```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);   ❶
WebApplication app = builder.Build();                                  ❶

app.UseWelcomePage();                                                  ❷

app.Run();                                                             ❸
```

COMBINING MIDDLEWARE IN A PIPELINE - HANDLING STATIC

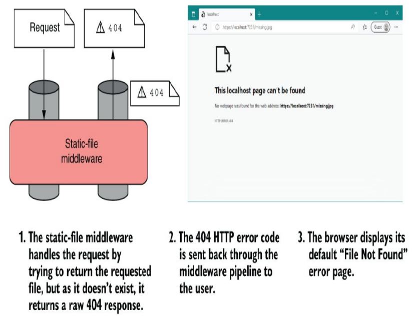


1. An HTTP request is made for the file moon.jpg.
Request
Web host/reverse proxy (IIS/Nginx/Apache)
Response
7. The HTTP response containing the file is sent to the browser.
2. The request is forwarded by IIS/Nginx/Apache to ASP.NET Core.
ASP.NET Core web server
3. The ASP.NET Core web server receives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

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

app.UseStaticFiles();    ❶

app.Run();
```



Request
√FILE
√FILE
Static-file middleware
1. The static-file middleware handles the request by returning the file requested.
2. The file stream is sent back through the middleware pipeline and out to the browser.
3. The browser displays the file returned in the response.

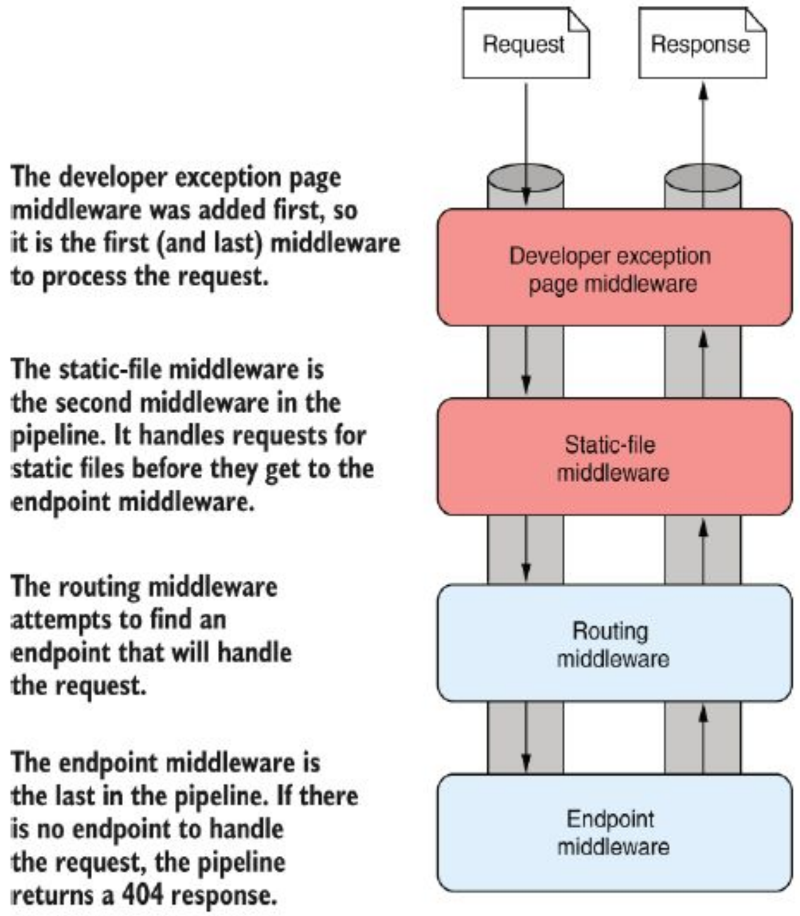


Request
⚠ 404
⚠ 404
Static-file middleware
This localhost page can't be found
1. The static-file middleware handles the request by trying to return the requested file, but as it doesn't exist, it returns a raw 404 response.
2. The 404 HTTP error code is sent back through the middleware pipeline to the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

HTTP RESPONSE STATUS CODES

- Every HTTP response contains a ***status code*** and, **optionally**, a ***reason phrase*** describing the status code.
  - **1xx—Information**. This code is not often used; it provides a general acknowledgment.
  - **2xx—Success**. The request was successfully **handled** and **processed**.
  - **3xx—Redirection**. The browser must follow the provided link to allow the user to log in, for example.
  - **4xx—Client error**. A problem occurred with the request. The request sent invalid data, for example, or the user isn't authorized to perform the request.
  - **5xx**—Server error. A problem on the server caused the request to fail.

COMBINING MIDDLEWARE IN A PIPELINE – A MINIMAL API



The developer exception page middleware was added first, so it is the first (and last) middleware to process the request.

The static-file middleware is the second middleware in the pipeline. It handles requests for static files before they get to the endpoint middleware.

The routing middleware attempts to find an endpoint that will handle the request.

The endpoint middleware is the last in the pipeline. If there is no endpoint to handle the request, the pipeline returns a 404 response.

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

UseDeveloperExceptionPage();          ❶
app.UseStaticFiles();                   ❷
app.UseRouting();                       ❸

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

app.Run();
```

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

❷ Adds the StaticFileMiddleware to the pipeline

❸ Adds the RoutingMiddleware to the pipeline

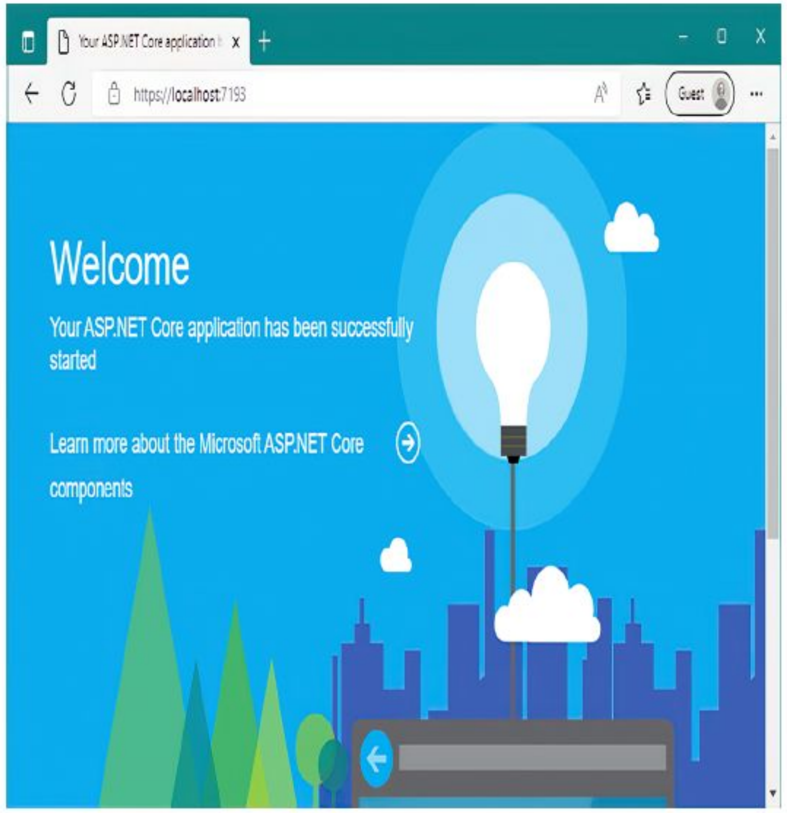
❹ Defines an endpoint for the application

COMBINING MIDDLEWARE IN A PIPELINE – A MINIMAL API APPLICATION

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

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

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

app.Run();
```

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

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



Your ASP.NET Core application h

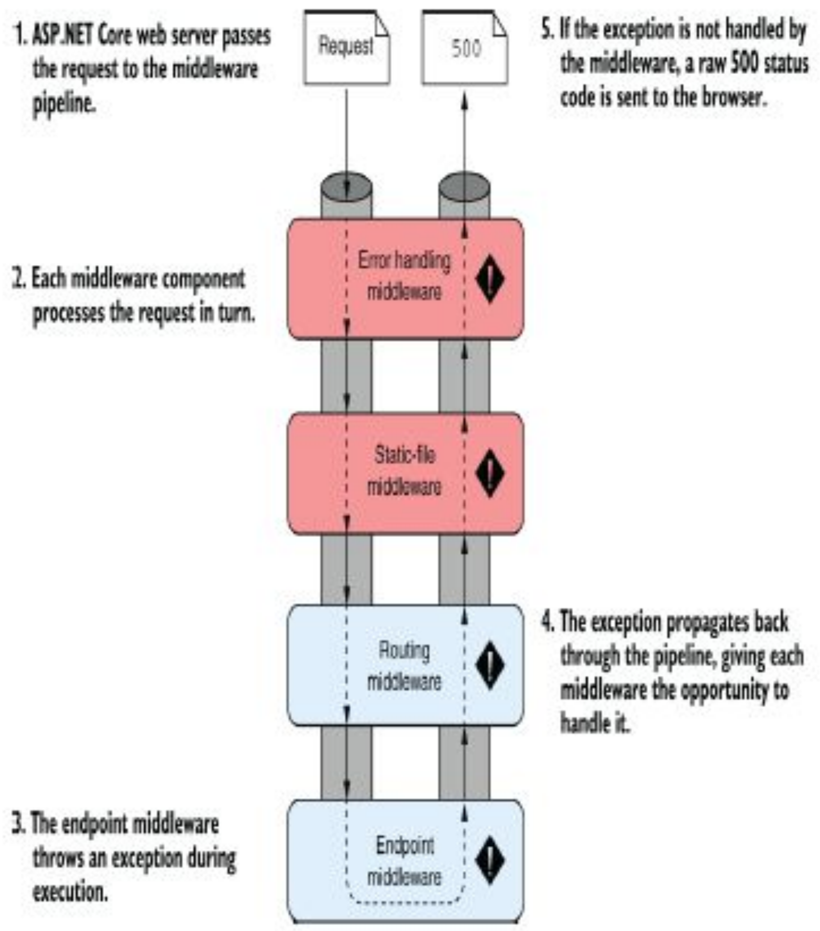
https://localhost:7193

Welcome

Your ASP.NET Core application has been successfully started

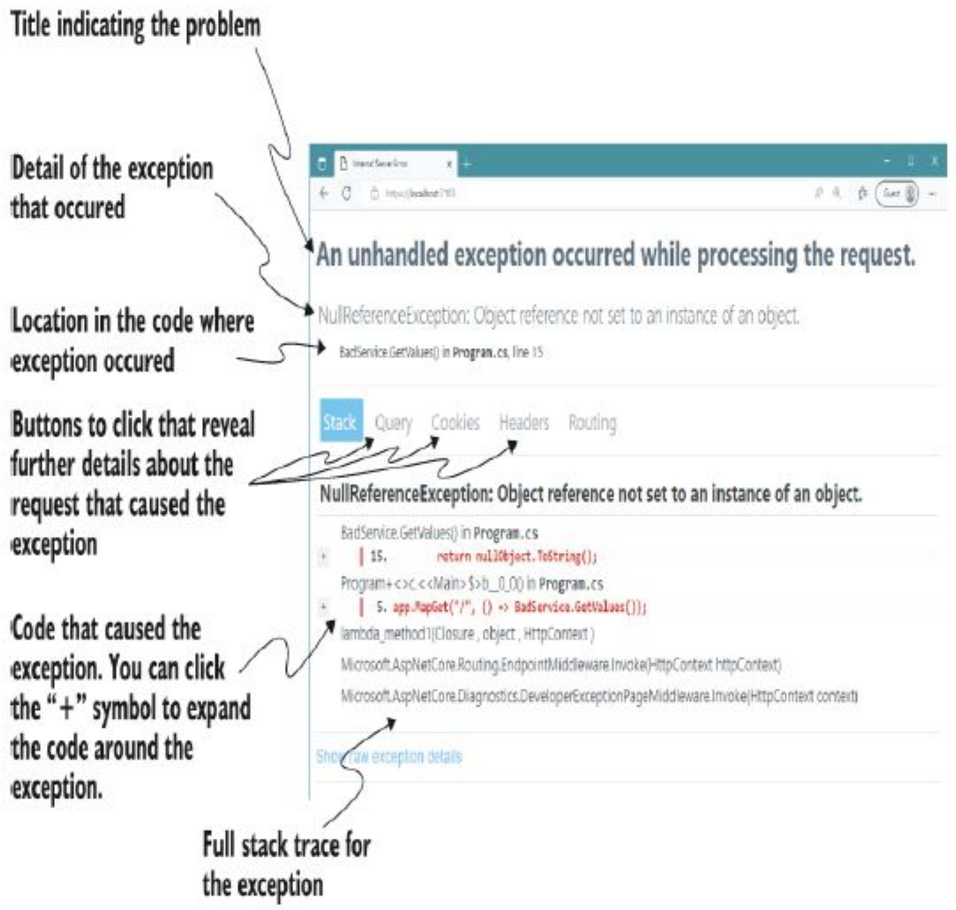
Learn more about the Microsoft ASP.NET Core components

HANDLING ERRORS USING MIDDLEWARE


1. ASP.NET Core web server passes the request to the middleware pipeline.
Request
500
5. If the exception is not handled by the middleware, a raw 500 status code is sent to the browser.
2. Each middleware component processes the request in turn.
Error handling middleware
Static-file middleware
Routing middleware
4. The exception propagates back through the pipeline, giving each middleware the opportunity to handle it.
3. The endpoint middleware throws an exception during execution.
Endpoint middleware

VIEWING EXCEPTIONS IN DEVELOPMENT: DEVELOPEREXCEPTIONPAGE

```
app.UseDeveloperExceptionPage();
```


Title indicating the problem
Detail of the exception that occured
Location in the code where exception occured
Buttons to click that reveal further details about the request that caused the exception
Code that caused the exception. You can click the "+" symbol to expand the code around the exception.
Full stack trace for the exception
Stack
Query
Cookies
Headers
Routing
Show raw exception details

An unhandled exception occurred while processing the request.

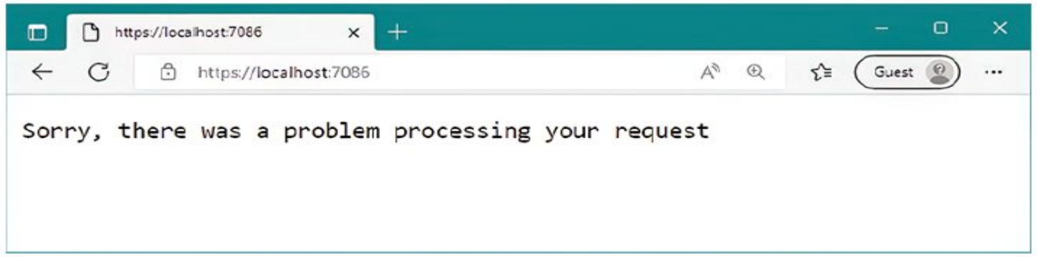
NullReferenceException: Object reference not set to an instance of an object.

BadService.GetValues() in **Program.cs**, line 15

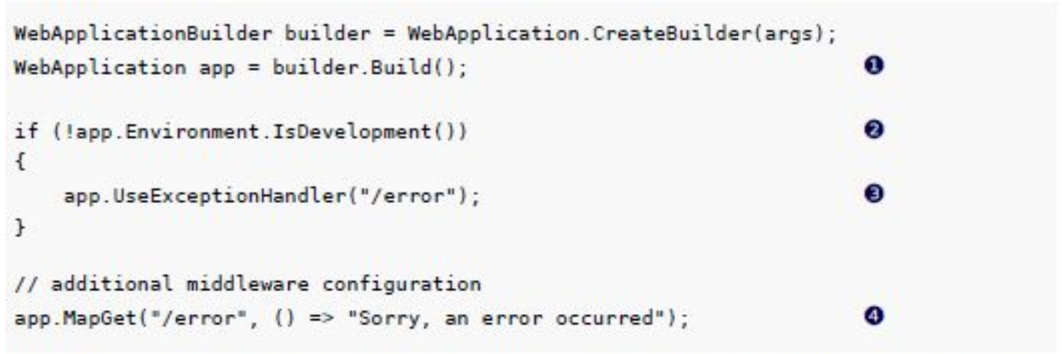
NullReferenceException: Object reference not set to an instance of an object.

```
BadService.GetValues() in Program.cs
+   | 15.        return nullObject.ToString();
Program+<>c.<<Main>$>b__0_0() in Program.cs
+   | 5. app.MapGet("/", () => BadService.GetValues());
lambda_method1(Closure , object , HttpContext )
Microsoft.AspNetCore.Routing.EndpointMiddleware.Invoke(HttpContext httpContext)
Microsoft.AspNetCore.Diagnostics.DeveloperExceptionPageMiddleware.Invoke(HttpContext context)
```

HANDLING EXCEPTIONS IN PRODUCTION: EXCEPTIONHANDLERMIDDLEWARE



```
Sorry, there was a problem processing your request
```



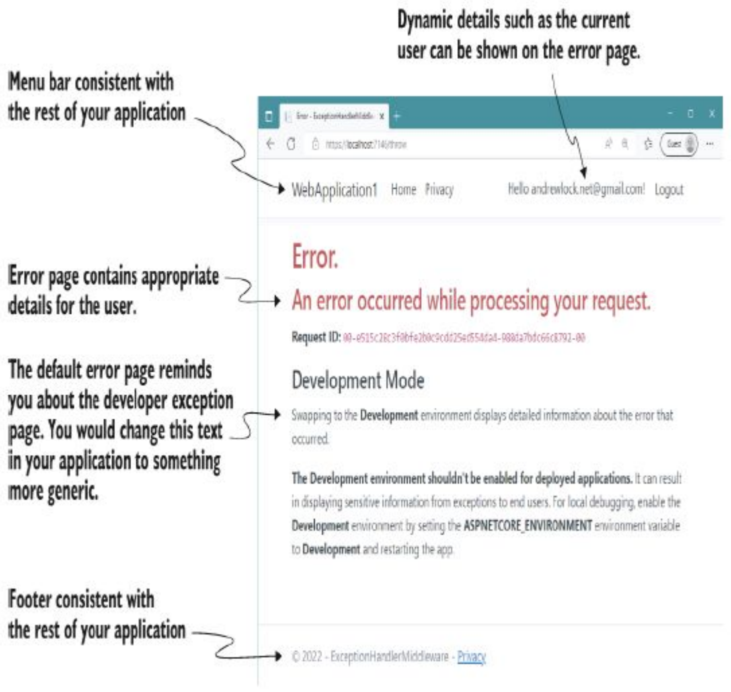
```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);
WebApplication app = builder.Build();                                  ❶

if (!app.Environment.IsDevelopment())                                  ❷
{
    app.UseExceptionHandler("/error");                                 ❸
}

// additional middleware configuration
app.MapGet("/error", () => "Sorry, an error occurred");               ❹
```

❶ In development, WebApplication automatically adds the developer exception page middleware.

❷ Configures a different pipeline when not running in development

❸ The ExceptionHandlerMiddleware won't leak sensitive details when running in production.

❹ This error endpoint will be executed when an exception is handled.


Dynamic details such as the current user can be shown on the error page.
Menu bar consistent with the rest of your application
Error page contains appropriate details for the user.
The default error page reminds you about the developer exception page. You would change this text in your application to something more generic.
Footer consistent with the rest of your application
https://localhost:7146/throw
WebApplication1
Home
Privacy
Hello andrewlock.net@gmail.com!
Logout

Error.

An error occurred while processing your request.

Request ID: 00-e515c28c3f0bfe2b0c9cdd25ed554da4-988da7bdc66c8792-00

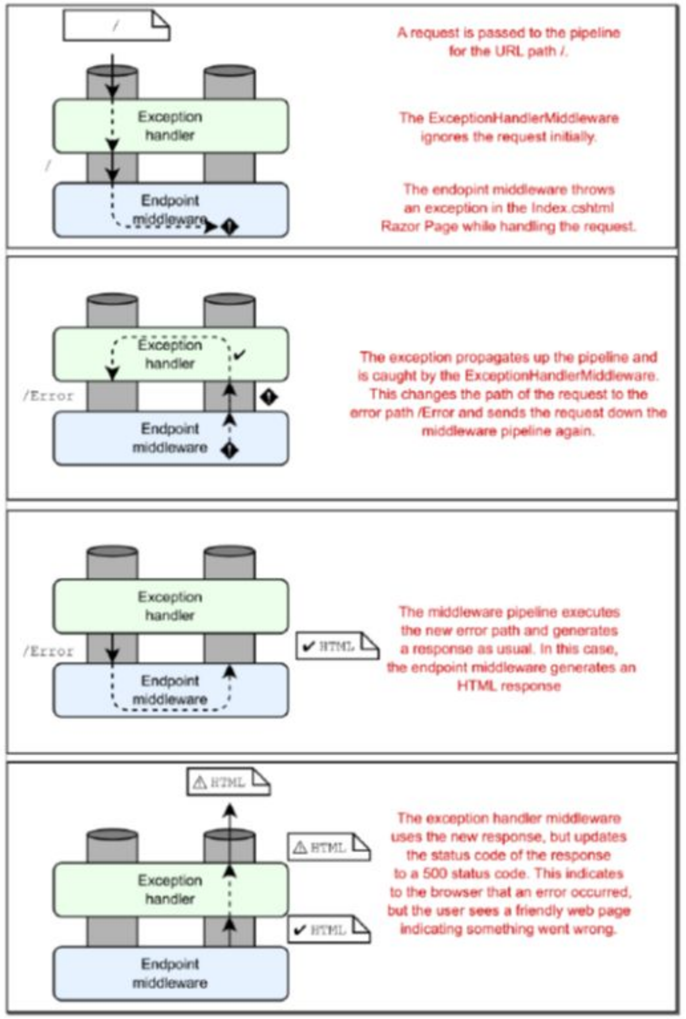
Development Mode

Swapping to the **Development** environment displays detailed information about the error that occurred.

The Development environment shouldn't be enabled for deployed applications. It can result in displaying sensitive information from exceptions to end users. For local debugging, enable the **Development** environment by setting the **ASPNETCORE_ENVIRONMENT** environment variable to **Development** and restarting the app.

© 2022 - ExceptionHandlerMiddleware - Privacy

HANDLING EXCEPTIONS IN PRODUCTION: EXCEPTIONHANDLERMIDDLE W



A request is passed to the pipeline for the URL path /.

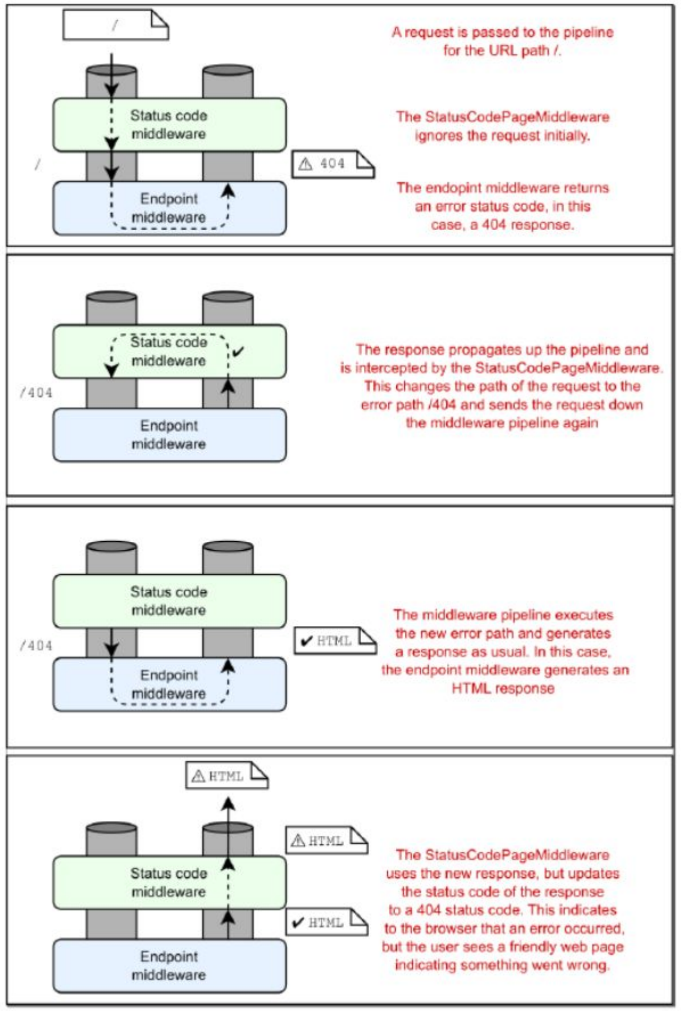
The ExceptionHandlerMiddleware ignores the request initially.

The endopint middleware throws an exception in the Index.cshtml Razor Page while handling the request.

The exception propagates up the pipeline and is caught by the ExceptionHandlerMiddleware. This changes the path of the request to the error path /Error and sends the request down the middleware pipeline again.

The middleware pipeline executes the new error path and generates a response as usual. In this case, the endpoint middleware generates an HTML response

The exception handler middleware uses the new response, but updates the status code of the response to a 500 status code. This indicates to the browser that an error occurred, but the user sees a friendly web page indicating something went wrong.



A request is passed to the pipeline for the URL path /.

The StatusCodePageMiddleware ignores the request initially.

The endopint middleware returns an error status code, in this case, a 404 response.

The response propagates up the pipeline and is intercepted by the StatusCodePageMiddleware. This changes the path of the request to the error path /404 and sends the request down the middleware pipeline again

The middleware pipeline executes the new error path and generates a response as usual. In this case, the endpoint middleware generates an HTML response

The StatusCodePageMiddleware uses the new response, but updates the status code of the response to a 404 status code. This indicates to the browser that an error occurred, but the user sees a friendly web page indicating something went wrong.