ASP.NET WEB API

Terms

- Controller: handles multiple requests/actions in a specific domain e.g. WeatherForecastController
- Action: is a single endpoint that can handle requests
- Typically a Controller includes multiple actions

Attributes

- ASP.NET uses attributes for different purposes
- Makes defining the WEB API easier in code
- Limits the need of code written

Attributes examples

- `[HttpPost]` configures a method to be handled as a Post request
- `[Route(...)]` defines URL pattern for the controller or its action
- `[Consumes(...)]` defines which content types can be configured from a specific action e.g. `application/xml`, `application/x-www-form-urlencoded`

Controllers

- end with keyword `Controller` e.g. `WeatherController`
- must derive from `ControllerBase`
- include different HTTP-endpoints
- usually created to handle requests for a specific domain

Controllers example

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

- `Route("[controller]")` defines that the controller is accessible via its name (without the Controller keyword)
  e.g. localhost:1234/WeatherForecast
- `[ApiController]` enables API-specific behaviors
  - Attribute Routing - Routes have to be defined using attributes
  - Automatic HTTP 400 responses for model violations
  - Binding source parameter inference
    - `[FromBody]`, `[FromForm]`, `[FromHeader]`, `[FromServices]` etc.
    - Attributes for each parameter define where the values should come from
  - etc.

Dependency injection

Dependency injection

- ASP.NET uses services to inject dependencies into its application
- When launching an ASP.NET application the `builder` is used to add different services
- Services may include: Controllers, DbContext (EF Core)

Custom Dependencies

- Custom Dependencies need an Interface and a class implementing the interface
- e.g. `builder.Services.AddScoped<IMyDependency, MyDependency>();`
- Different options to add the dependencies:
  - transient: are always different
  - scoped: during lifetime of a single request
  - singleton: are the same for every request

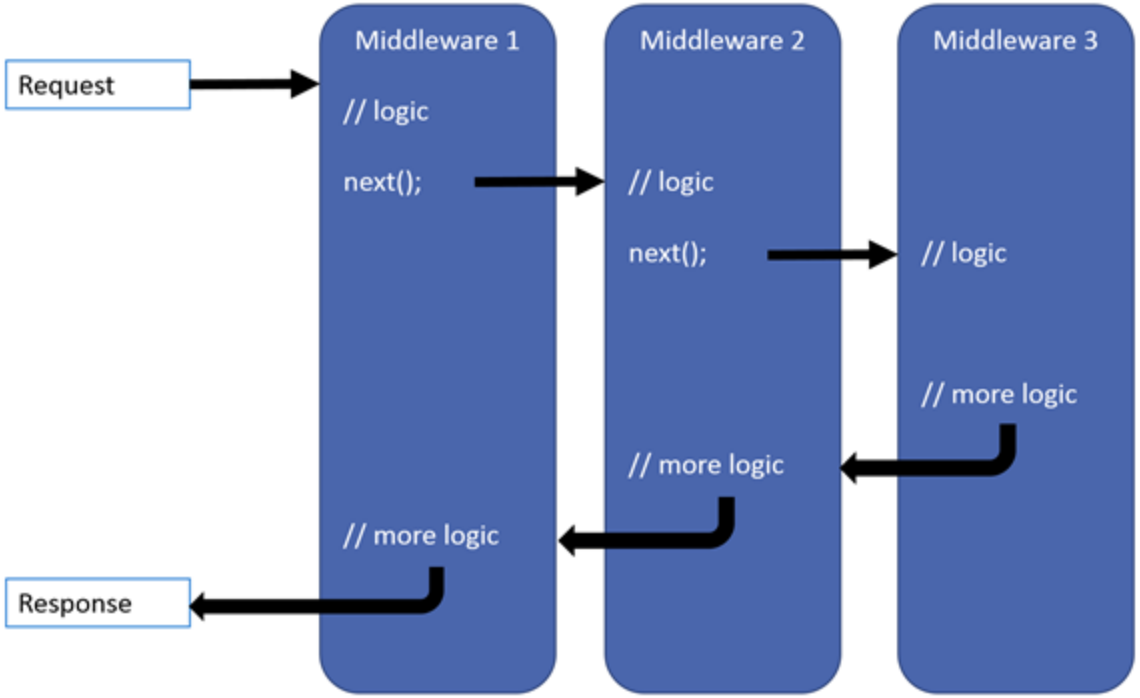
More: Dependency injection in ASP.NET Core | Microsoft Learn

Middleware

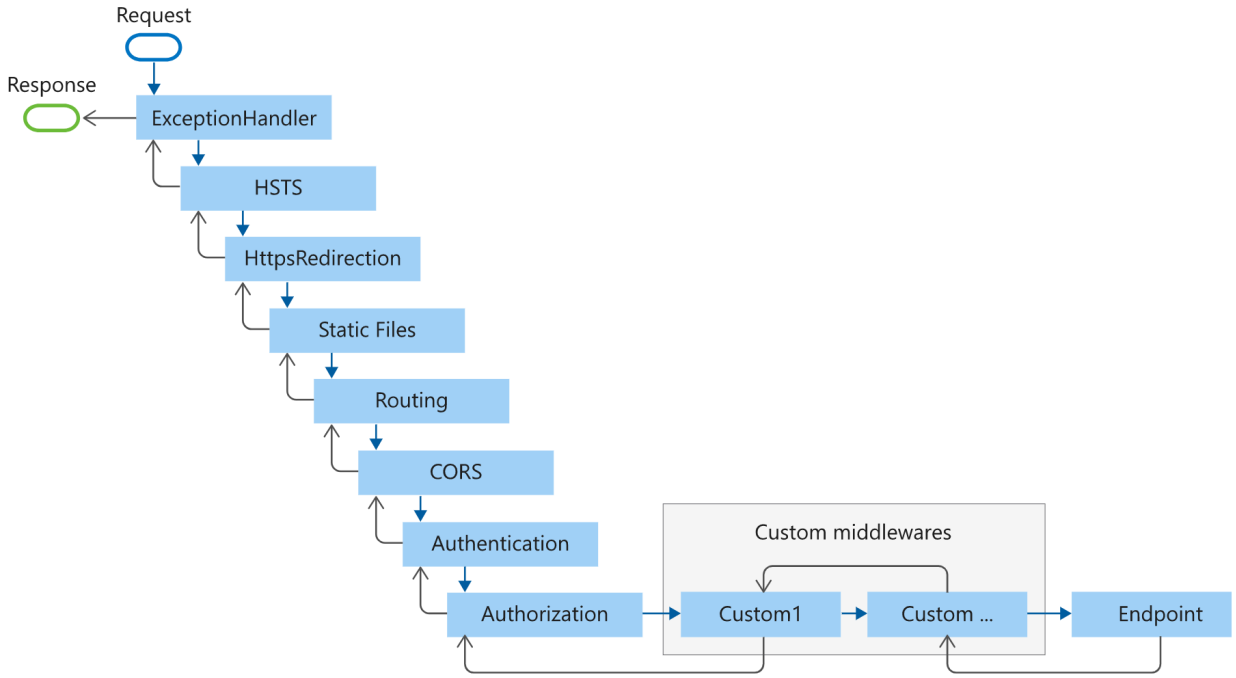
The handling of requests is executed by a series of 'middleware' components.

- Each middleware is either invoking the next middleware or terminating the request
- To enable different middleware components the built ASP.NET app includes `Use[Feature]` methods
- e.g. `UseAuthorization`, `UseHttpsRedirection`

Middleware Basics



Middleware Order



The middleware order is defined by the `Use[Feature]` method execution order when creating the ASP.NET app.

Host

The host includes all of the mentioned resources.

- HTTP server implementation
- Middleware
- Logging
- Dependency Injection services
- Configuration

Most commonly used: `WebApplication`

Host `WebApplication` example

```
using Microsoft.AspNetCore.Mvc;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

var app = builder.Build();

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

`WebApplication`

- Uses Kestrel + IIS integration
- Loads configuration from (examples):
  - appsettings.json
  - environment variables
  - command line arguments

Logging

- ASP.NET has an included logging API
- Works with different logging providers e.g. Console, Debug, Azure, Windows
- Configured using the Builder

```
builder.Logging.AddConsole();
```

Log from Model

- Use dependency injection

```
public class AboutModel : PageModel
{
    private readonly ILogger _logger;

    public AboutModel(ILogger<AboutModel> logger)
    {
        _logger = logger;
    }

    public void OnGet()
    {
        _logger.LogInformation("About page visited at {DT}",
        DateTime.UtcNow.ToLongTimeString());
    }
}
```

Sources

- Create web APIs with ASP.NET Core | Microsoft Learn
- ASP.NET Core fundamentals overview | Microsoft Learn
- Logging in .NET Core and ASP.NET Core | Microsoft Learn
- ASP.NET Core Middleware | Microsoft Learn
- Dependency injection in ASP.NET Core | Microsoft Learn