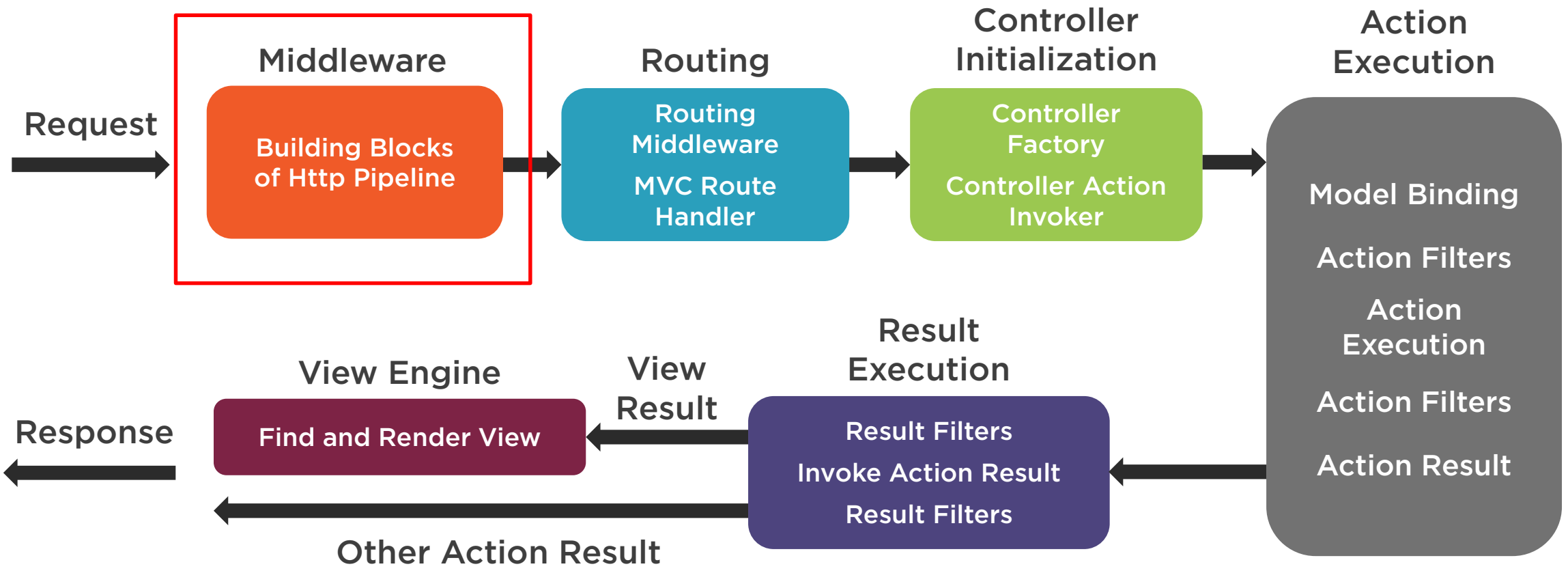
Understanding the Middleware Pipeline

Alex Wolf

www.alexwolfthoughts.com

The MVC Request Life Cycle



To-Do List

- What Is Middleware?
- Demo – Touring the Program and Startup Classes
- Demo – Building a Simple Middleware Pipeline
- Demo – Writing a Reusable Middleware Component
- Demo – MVC Middleware Pipeline Internals
- Visualizing the MVC Middleware Pipeline
- Comparing Middleware with HttpModules and HttpHandlers

What is Middleware?

Middleware is the series of components that form the application request pipeline.

Features Provided by Middleware

Routing

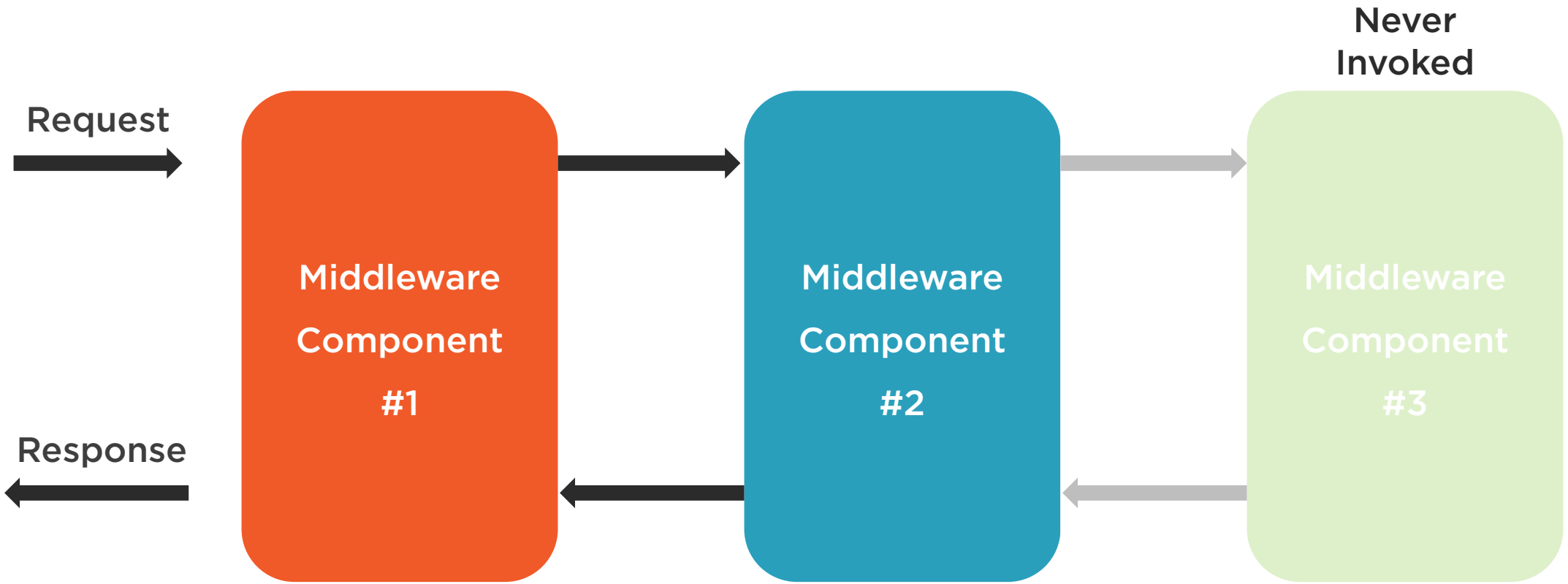
Session

CORS

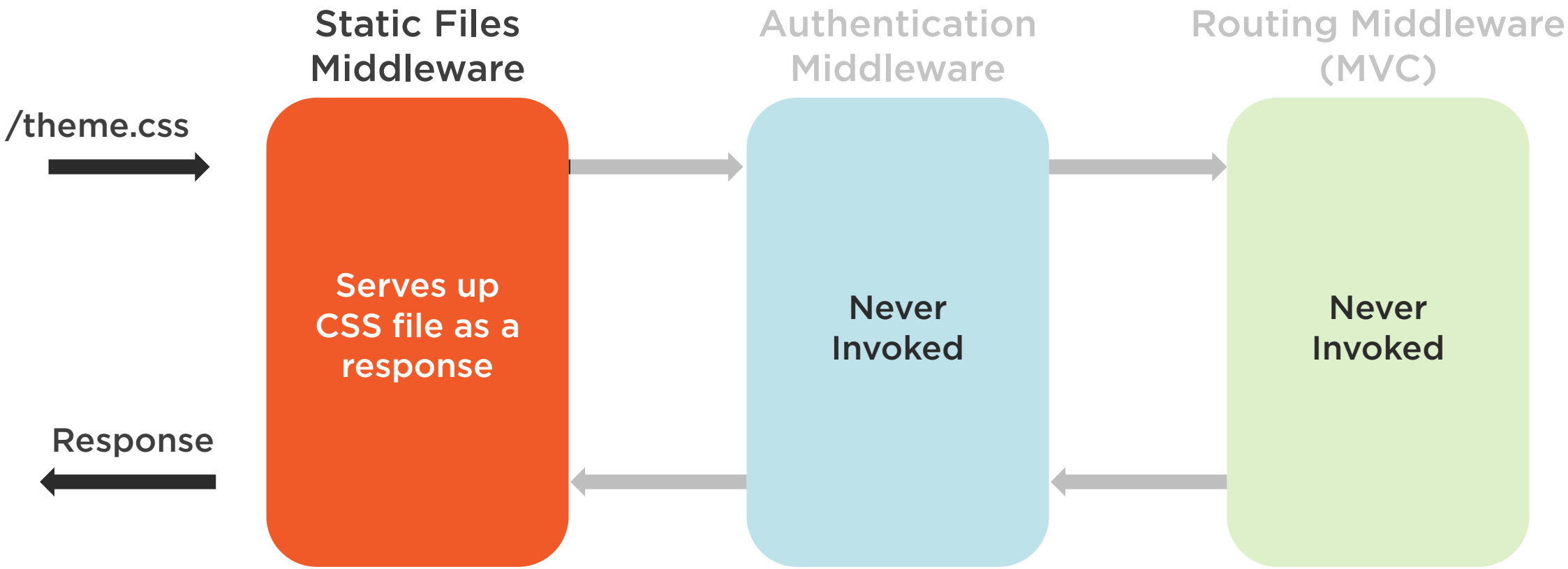
Authentication

Caching

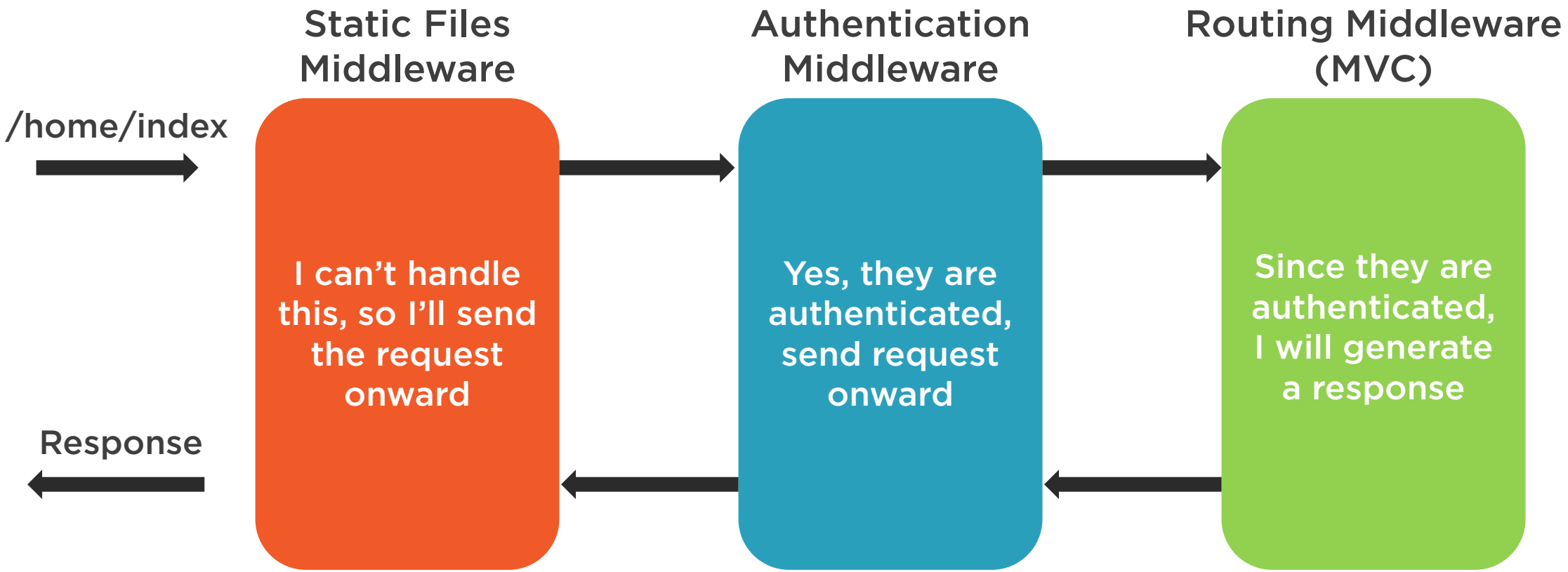
The Middleware Pipeline



A Sample Middleware Pipeline



A Sample Middleware Pipeline (cont.)



Configuring the Middleware Pipeline

Run

Generate a response and short circuit the request

Use

Perform logic and send request to next component

Map

Conditionally send the request to other Middleware

```
app.Use(async context => {
        await context.Response.WriteAsync("Hello World, from Middleware!")
    });
}
```

Coding Middleware

Can vary widely in complexity

Can be written inline or in a separate class

```
app.Use(async (context, next) =>
    // Before logic here
    await next.Invoke();
    // After logic here
    });
}
app.Run(async context => {
    await context.Response.WriteAsync
    ("I will generate the response")
    });
}
```

- Middleware implemented with "Use" method can forward the request onto the next delegate

- Second Middleware component implemented with "Run" generates the response

```
app.Map("/hello-world", SayHello);
```

- If incoming requests ends in “hello-world” execute the SayHello method

```
private static void SayHello
(IApplicationBuilder app)

{

    app.Run(async context => {

    await context.Response.WriteAsync
    ("Hi, the request was mapped here.")

    });

}
```

- SayHello method generates response using the Run method

```
public class HelloMiddleware {

private RequestDelegate _next;

public HelloMiddleware(RequestDelegate next)

{

     _next = next;

}

public Task Invoke(HttpContext context)

{

     // Logic here

     await _next.Invoke(context)

}

}
```

- Custom Middleware class
- Constructor accepting the “next” Middleware delegate
- Invoke method to execute logic
- Forward request on to next Middleware component

The Program and Startup Classes

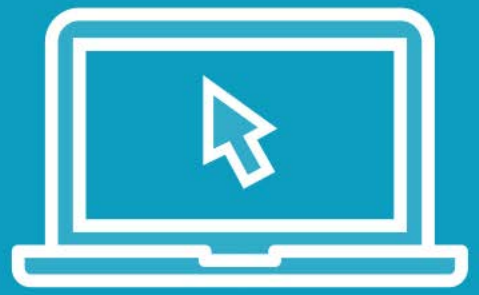
Program Class

Defines a Main() method used as the entry point into an application

Startup Class

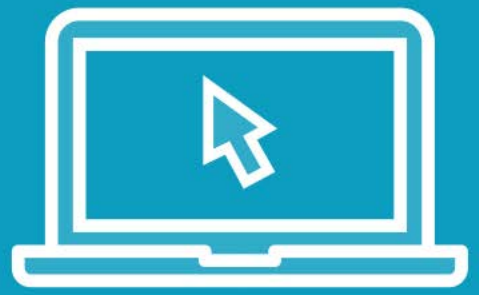
Defines a Configure() method used to register Middleware components

Demo



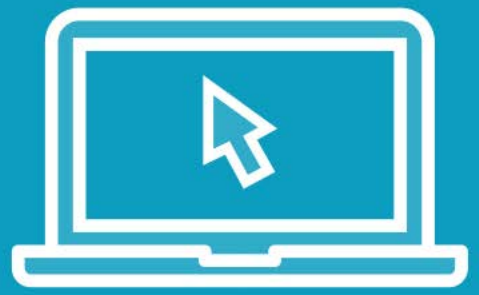
Understanding the MVC Middleware Pipeline

Demo



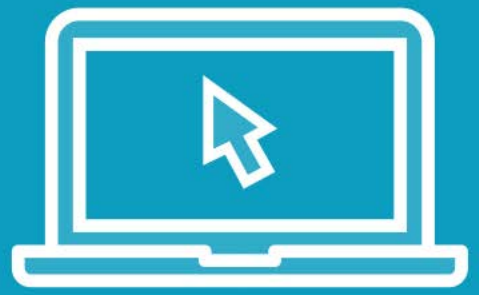
Building a Simple Middleware Pipeline

Demo



Writing a Reusable Middleware Component

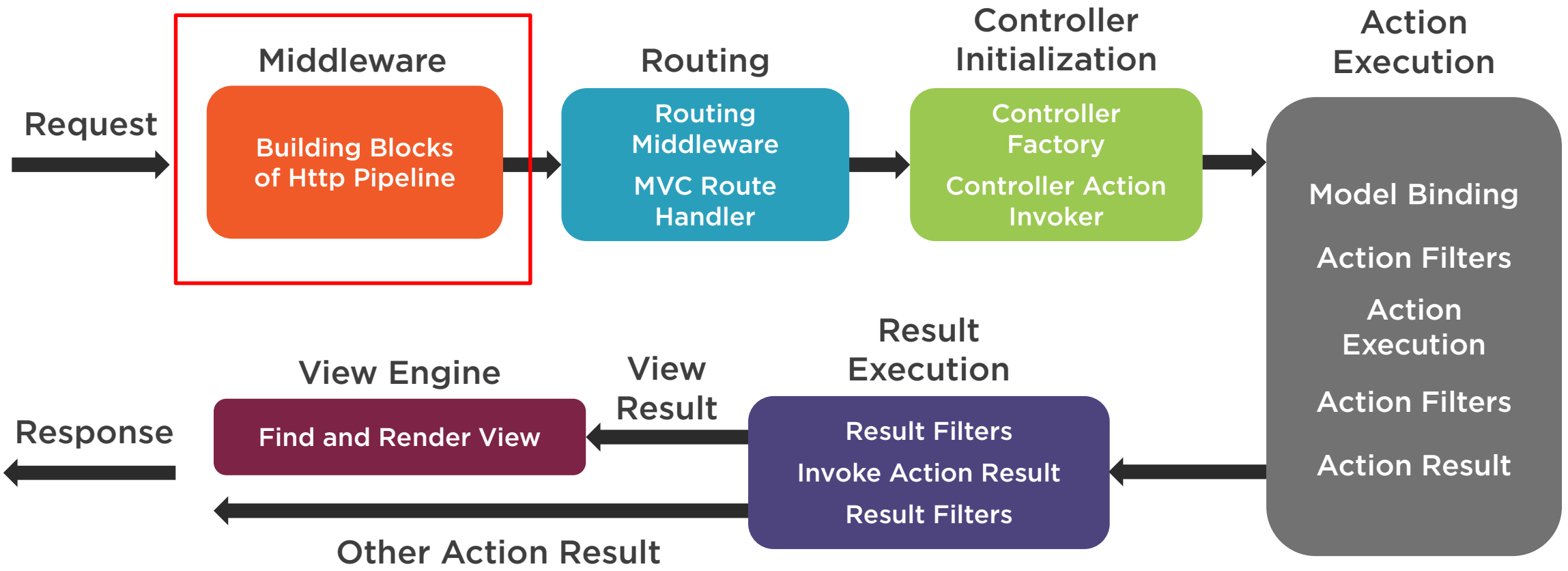
Demo



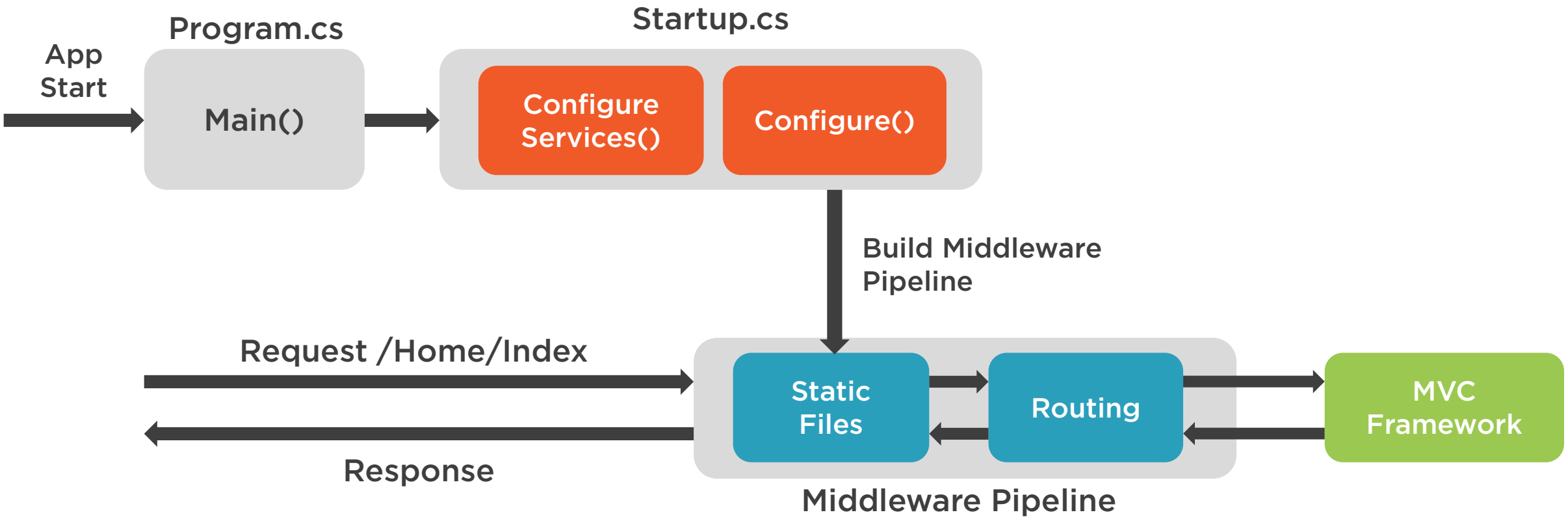
MVC Middleware Pipeline Internals

Visualizing the MVC Middleware Pipeline

The MVC Request Life Cycle

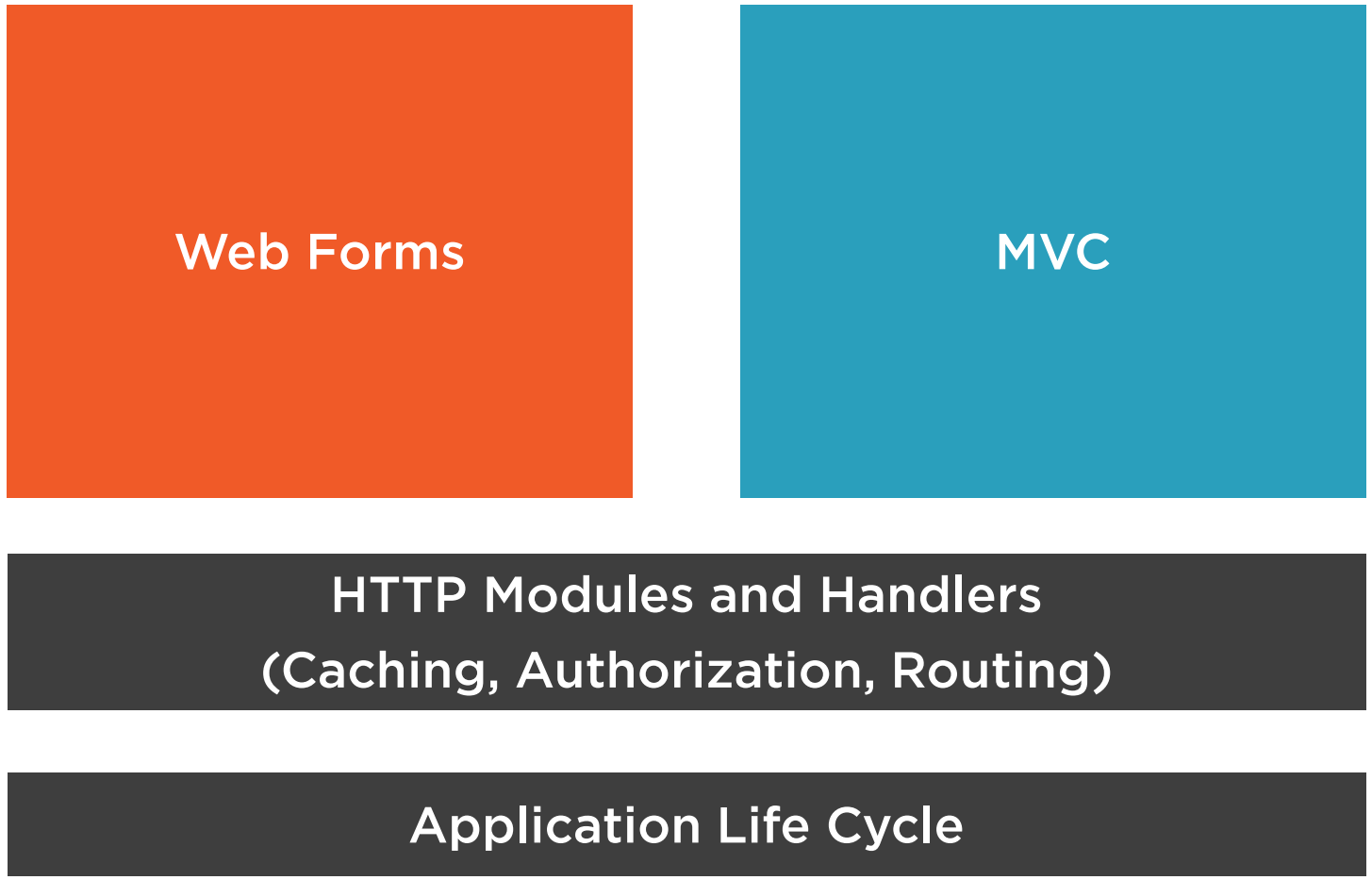


Understanding the MVC Middleware Pipeline



Comparing Middleware with HTTP Modules and Handlers

The Purpose of HTTP Modules and Handlers



Understanding HTTP Modules and Handlers

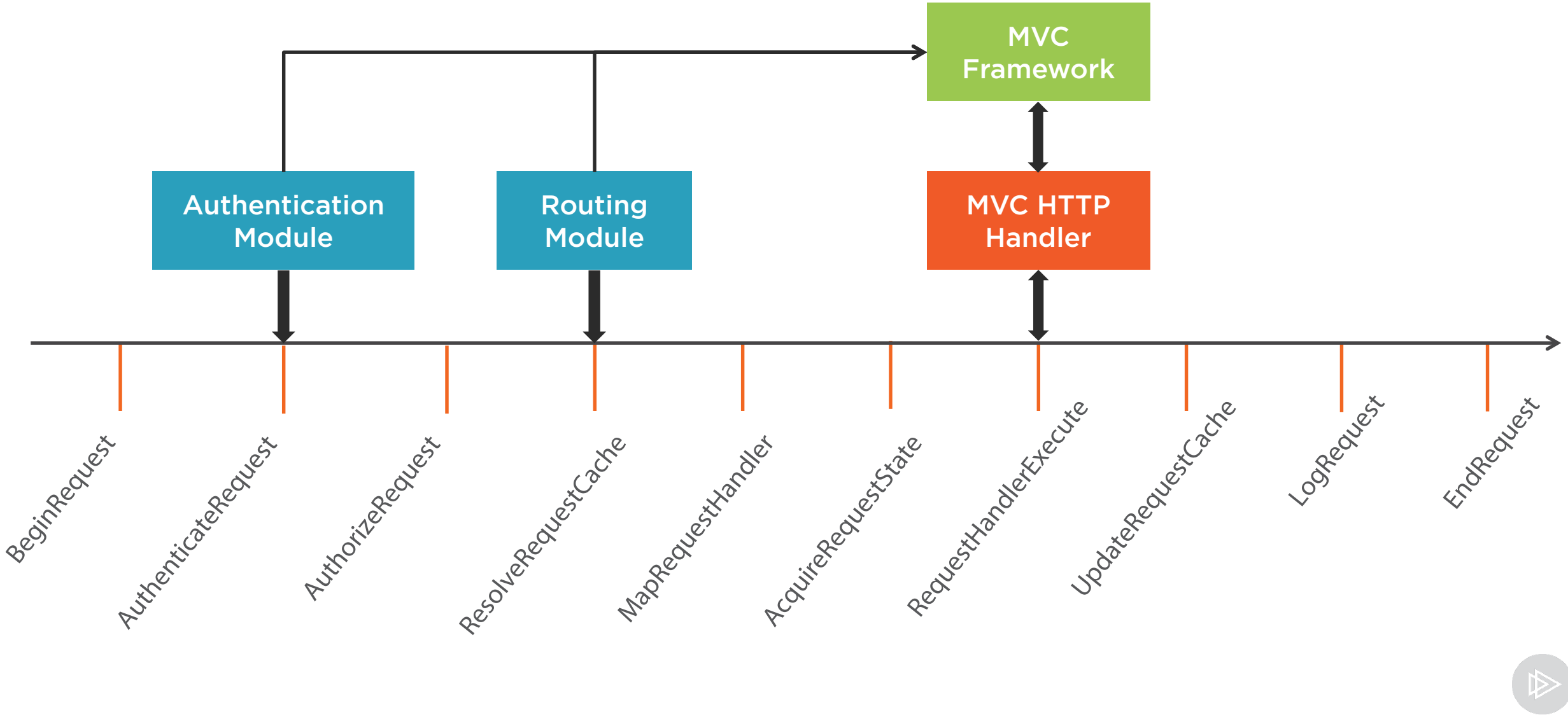
HTTP Modules

Hook into Application Life Cycle Events to provide reusable services

HTTP Handlers

Ultimately responsible for generating a response back to the client

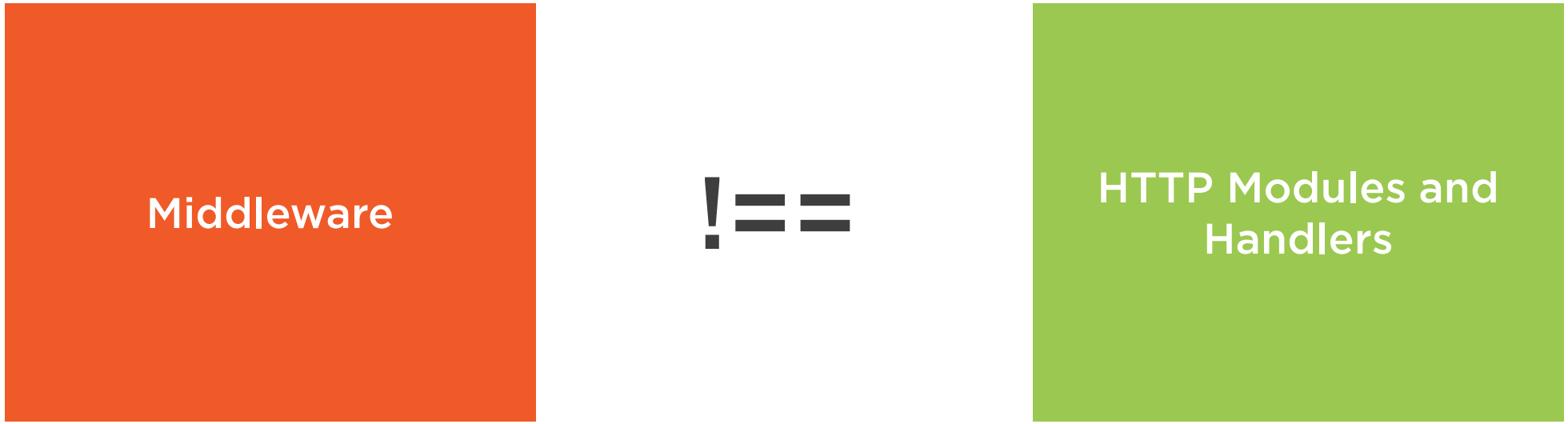
The Legacy Application Life Cycle



Comparing Middleware, Modules, and Handlers

Middleware	HTTP Handlers	HTTP Modules
Generates a response for the request Provides application level services and features	Generates a response for the request	Provides application level services and features

Middleware vs Modules and Handlers



Contrasting Middleware, Modules, and Handlers

Middleware	HTTP Handlers	HTTP Modules
Executes in the order and reverse order that components are inserted into the pipeline	Executes code for every associated application event	Executes once to generate the response

Benefits of Middleware

Lightweight and Highly Modular

Cross Platform Compatibile

Free From Legacy Dependencies

Summary

Middleware forms the request handling pipeline

The Program and Startup classes provide essential application configuration

Middleware can exist as inline methods or as reusable classes

MVC relies on Middleware to serve static files and receive routed requests

Middleware replaces the functionality of HTTP Modules and Handlers

Middleware fulfills modular and cross compatible design goals of .NET Core