Setting up an ASP.NET Core 6 MVC Application

Gill Cleeren
CTO Xpirit Belgium

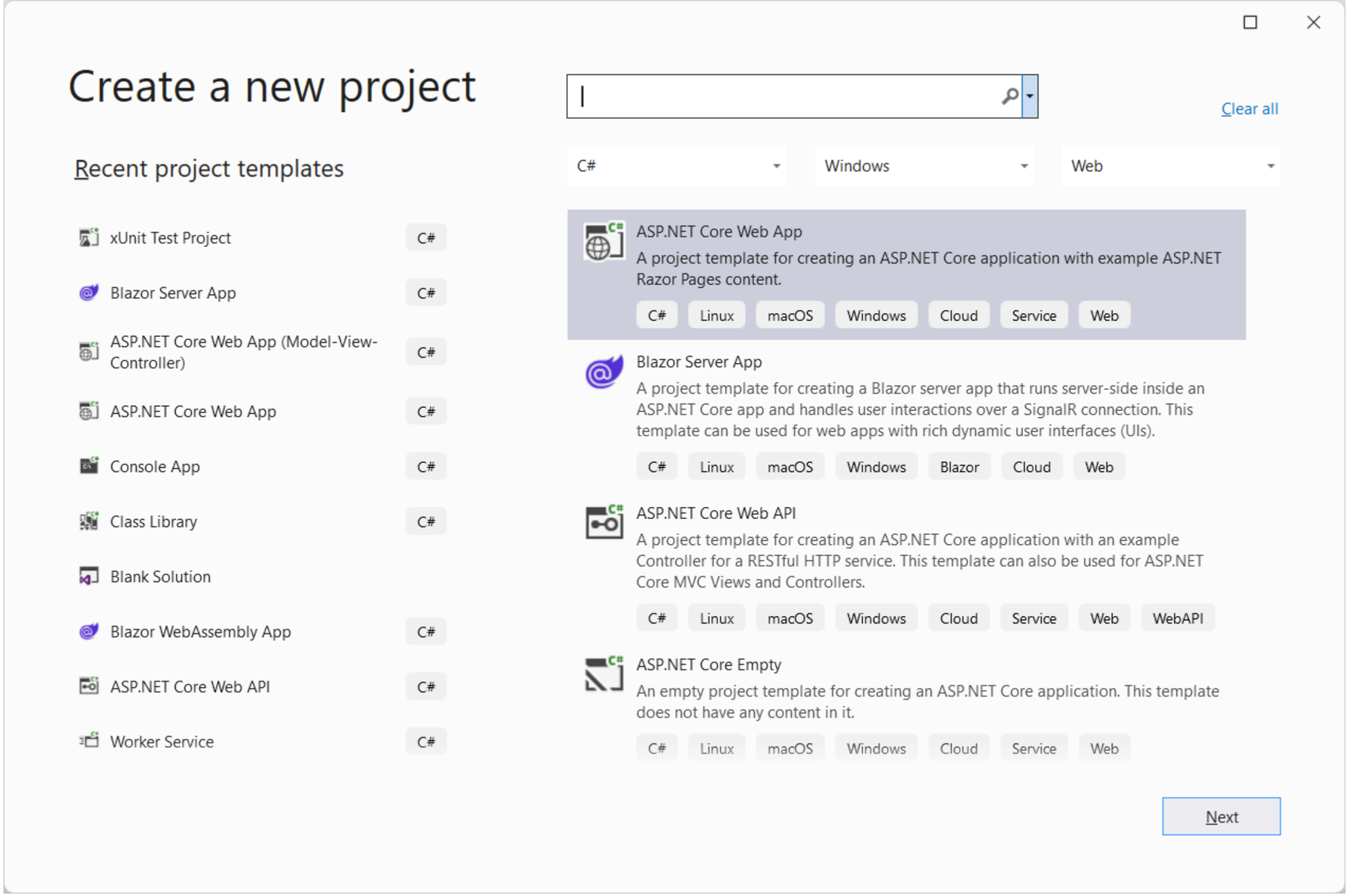
@gillcleeren – xpirit.com/gill

Module overview

- Creating a new project
- Exploring the generated files
- Configuring the site
- How ASP.NET Core handles a request

Creating a New Project

Templates


Create a new project
Clear all
Recent project templates
xUnit Test Project C#
Blazor Server App C#
ASP.NET Core Web App (Model-View-Controller) C#
ASP.NET Core Web App C#
Console App C#
Class Library C#
Blank Solution
Blazor WebAssembly App C#
ASP.NET Core Web API C#
Worker Service C#
C#
Windows
Web
ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web
Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Blazor Cloud Web
ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web WebAPI
ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# Linux macOS Windows Cloud Service Web
Next

Templates in the .NET CLI

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\gill> dotnet new
The 'dotnet new' command creates a .NET project based on a template.

Common templates are:
Template Name         Short Name    Language    Tags
--------------------  ------------  ----------  -------------------
ASP.NET Core Web App  webapp,razor  [C#]        Web/MVC/Razor Pages
Blazor Server App     blazorserver  [C#]        Web/Blazor
Class Library         classlib      [C#],F#,VB  Common/Library
Console App           console       [C#],F#,VB  Common/Console
Windows Forms App     winforms      [C#],VB     Common/WinForms
WPF Application       wpf           [C#],VB     Common/WPF

An example would be:
   dotnet new console

Display template options with:
   dotnet new console -h
Display all installed templates with:
   dotnet new --list
Display templates available on NuGet.org with:
   dotnet new web --search

PS C:\Users\gill>
```

Demo

Creating a new project using a template

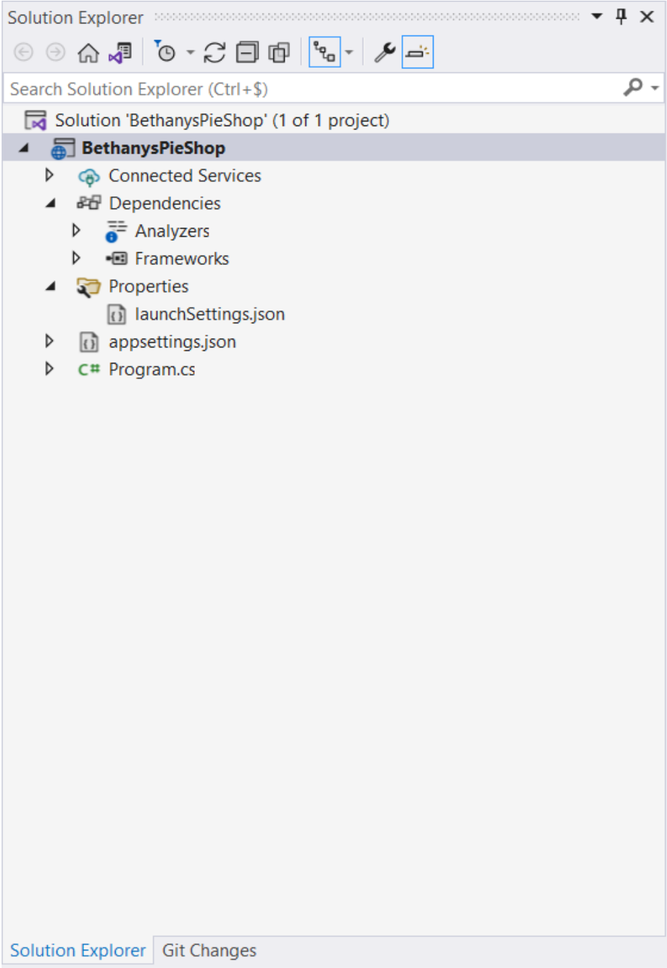
Building and running the application

Demo

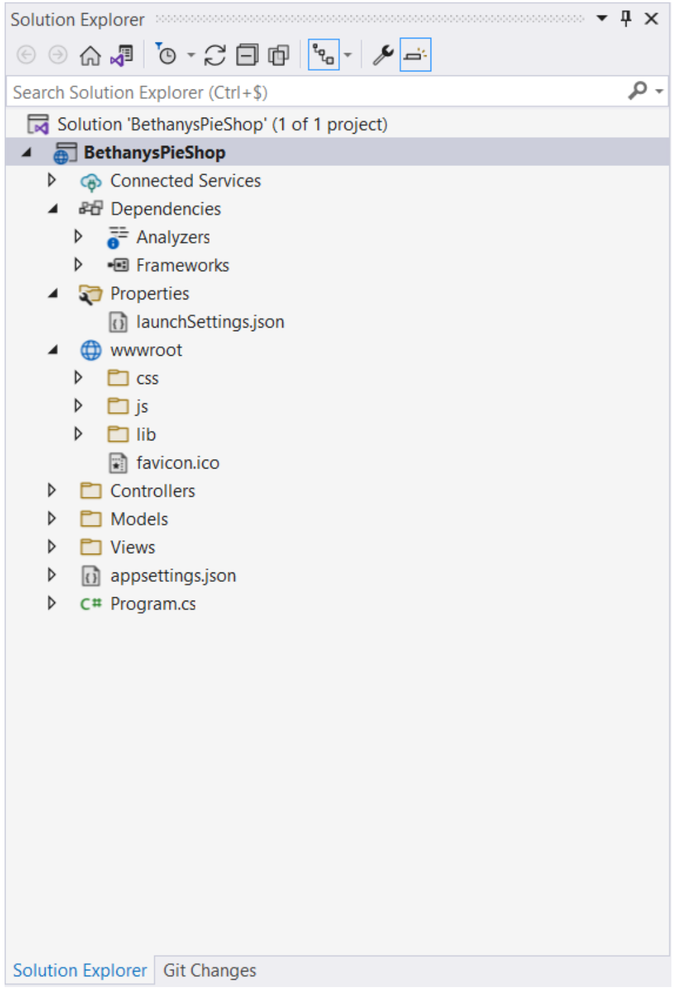
Using the CLI to create a new web application

Exploring a New Project

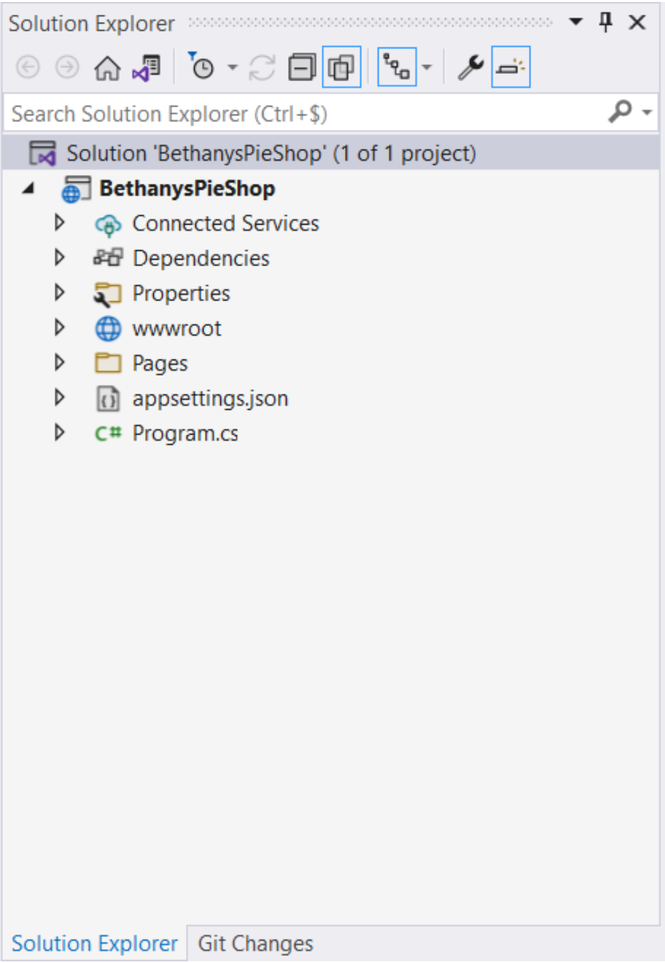
Project Structure (Empty Web Application)



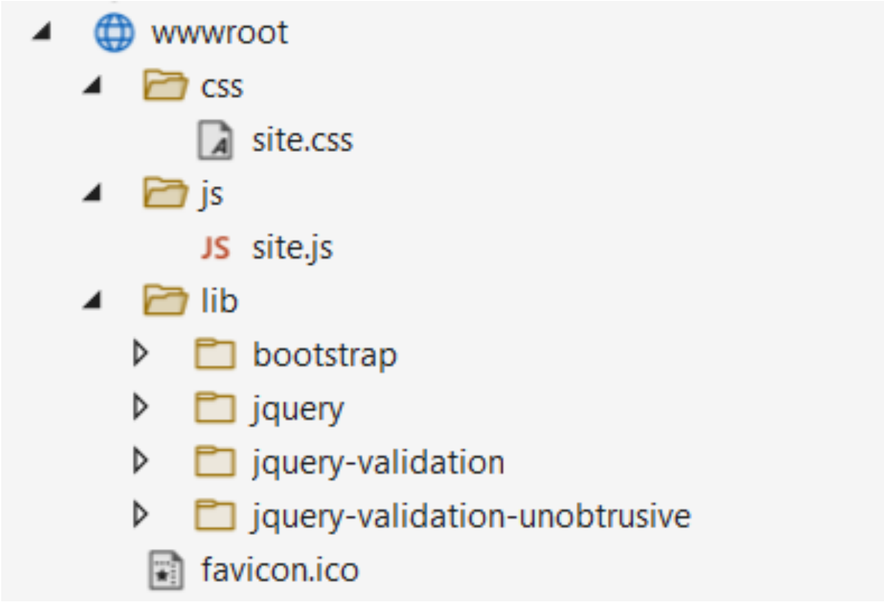
Project Structure (Web Application MVC)



Project Structure (Razor Pages)



The wwwroot Folder



wwwroot/image1.jpg

http://bethanyspieshop.com/image1.jpg

The csproj File

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

</Project>
```

Adding Dependencies

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>

</Project>
```

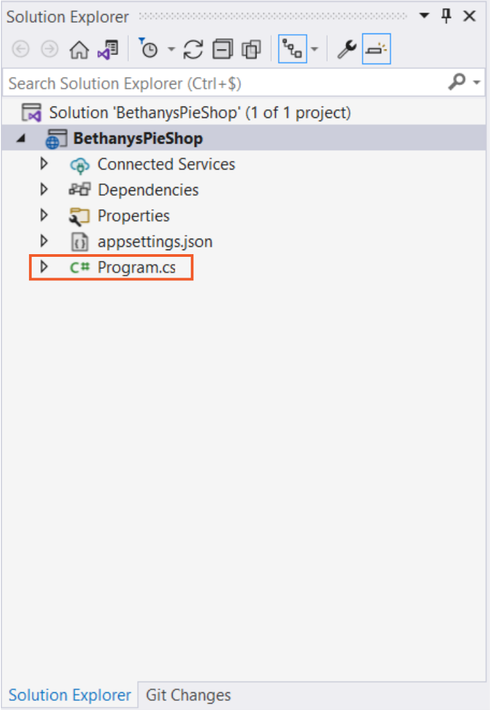
Demo

Exploring the generated files

Looking at launchSettings.json

Configuring the Site

The Program Class



ASP.NET Core applications start like console applications

- static void main
- “Replaced” with .NET 6 & C# 10 with top-level statements

Contains logic to start the server and listen for requests as well as configuration of the application

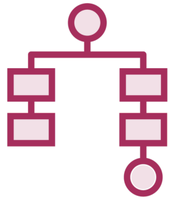
```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

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

app.Run();
```

The Default Program.cs

The CreateBuilder Method

Set up Kestrel server

Configure IIS integration



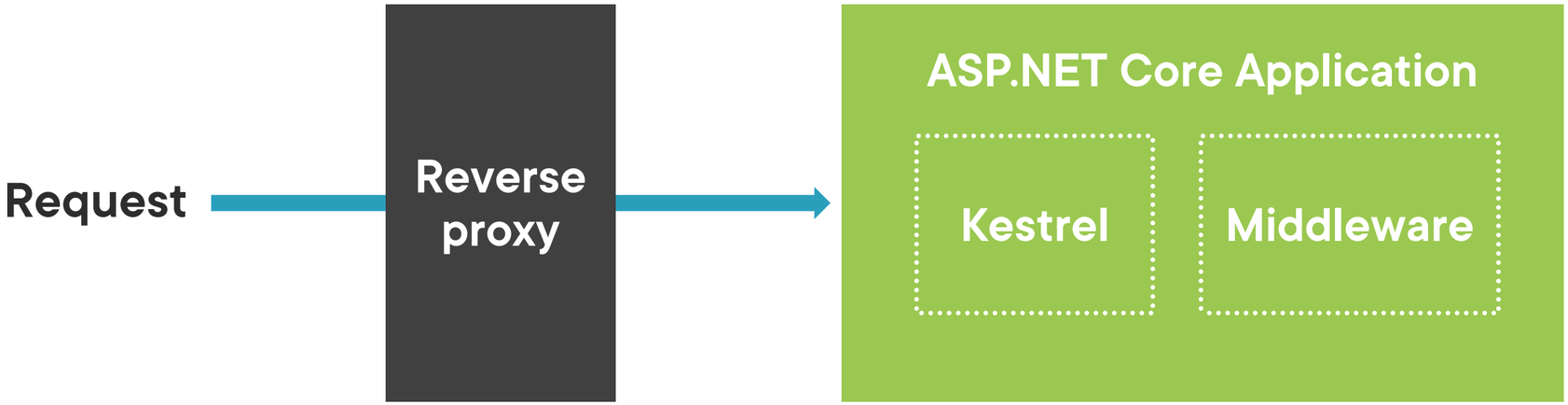
Specify content root

Read application settings

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

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

app.Run();
```

The Default Program.cs

Sidestep: Running a Server with Kestrel

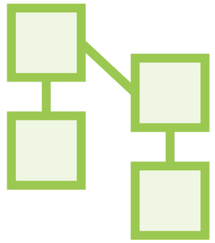


Configuration of the Application

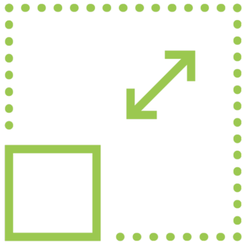
Service registration

Middleware

Service Registration



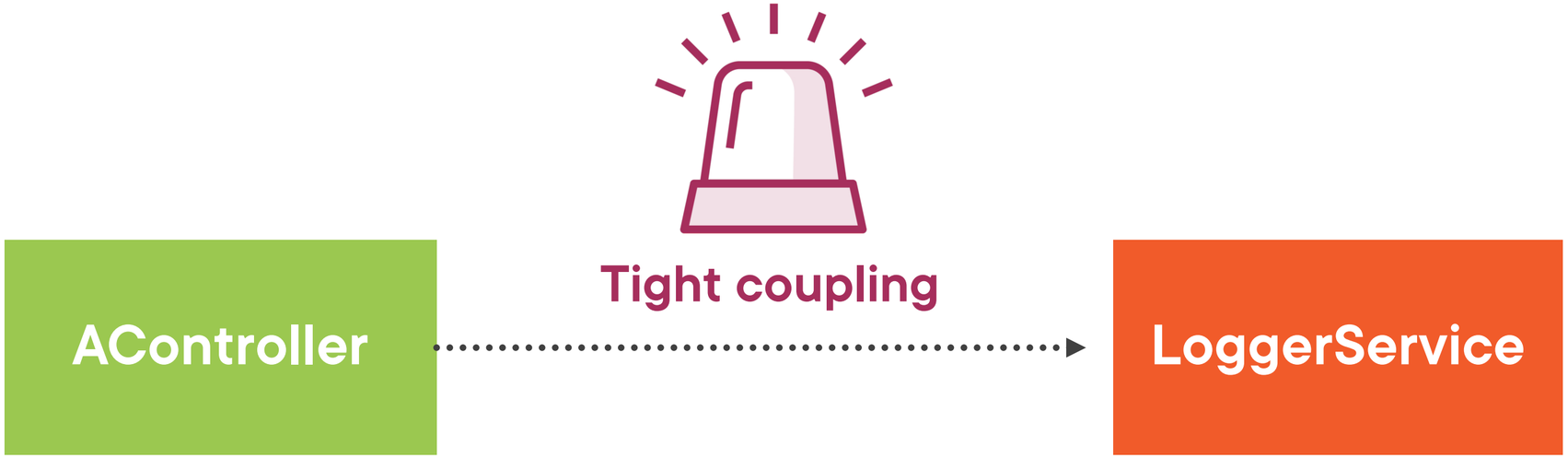
All classes used by the application



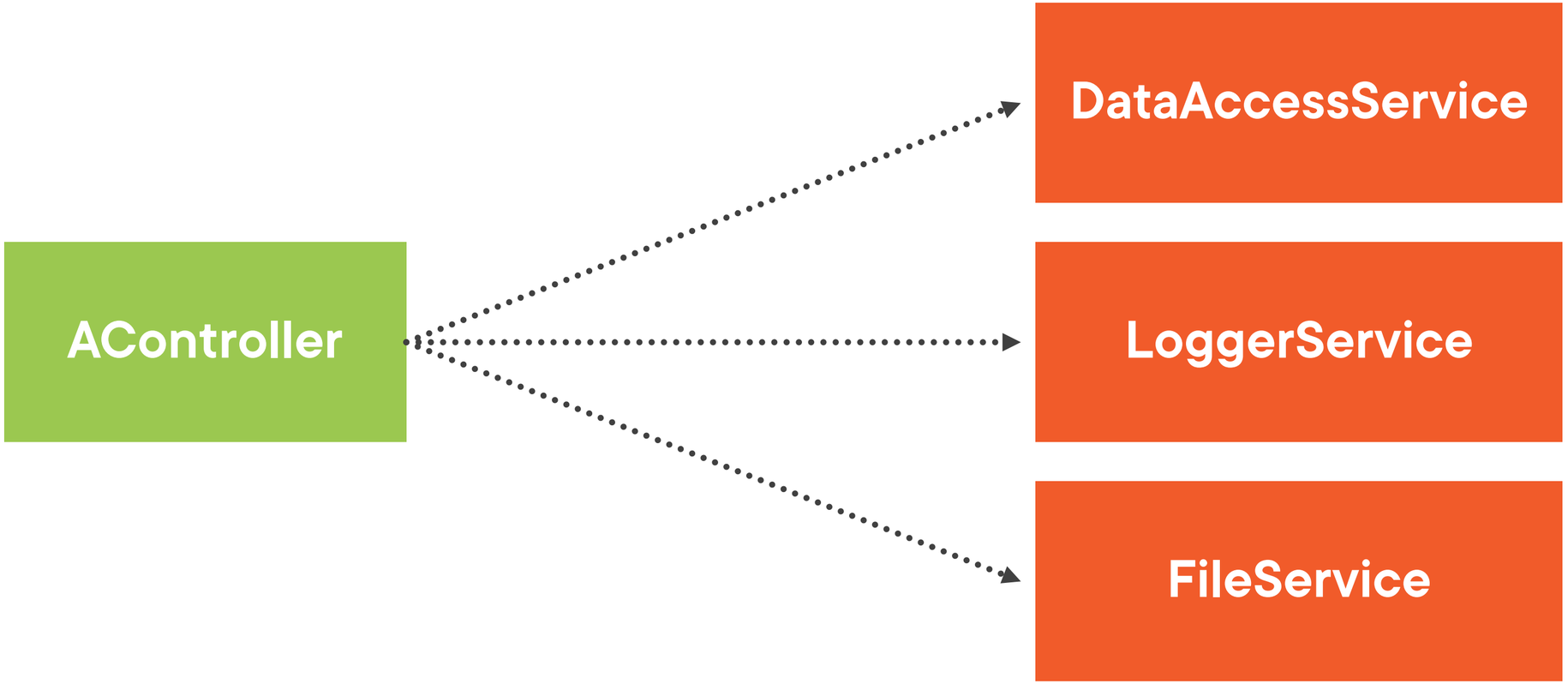
More modular approach

More dependencies will need to be injected

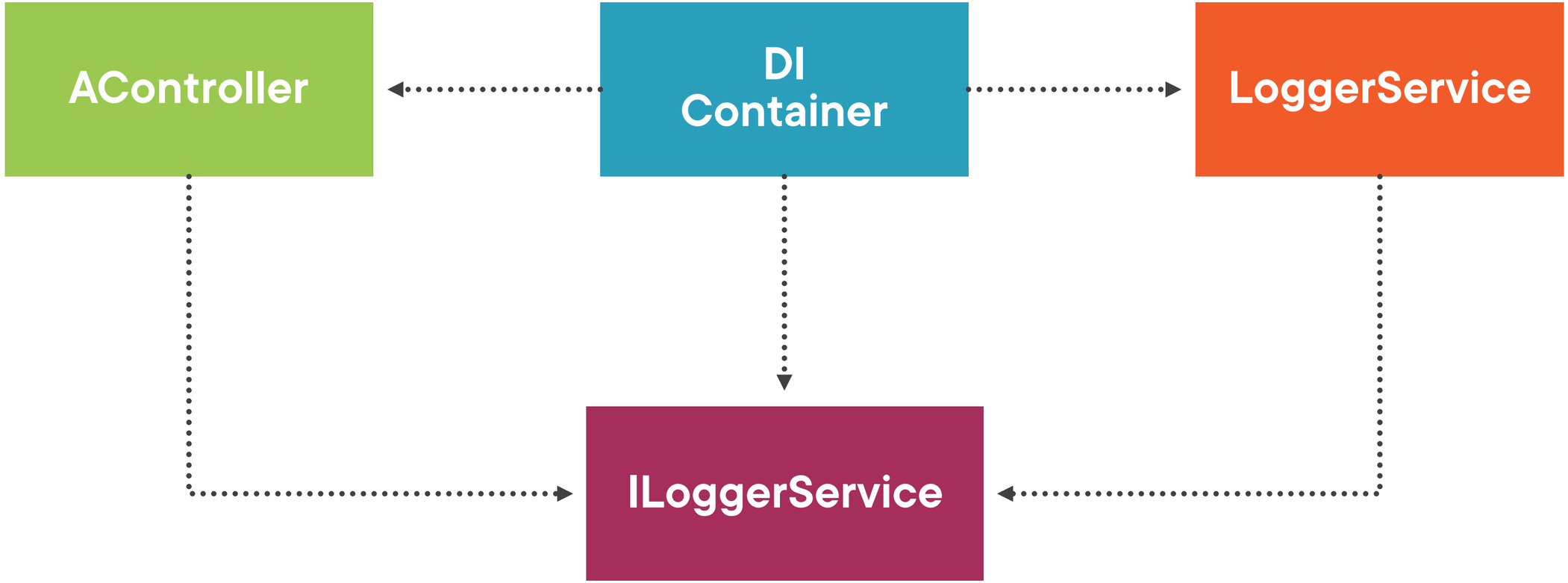
Using Services



Initializing Dependencies



Introducing Dependency Injection (DI)



Registering Services

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddScoped<ILoggerService, LoggerService>();

var app = builder.Build();

...

app.Run();
```

Framework services

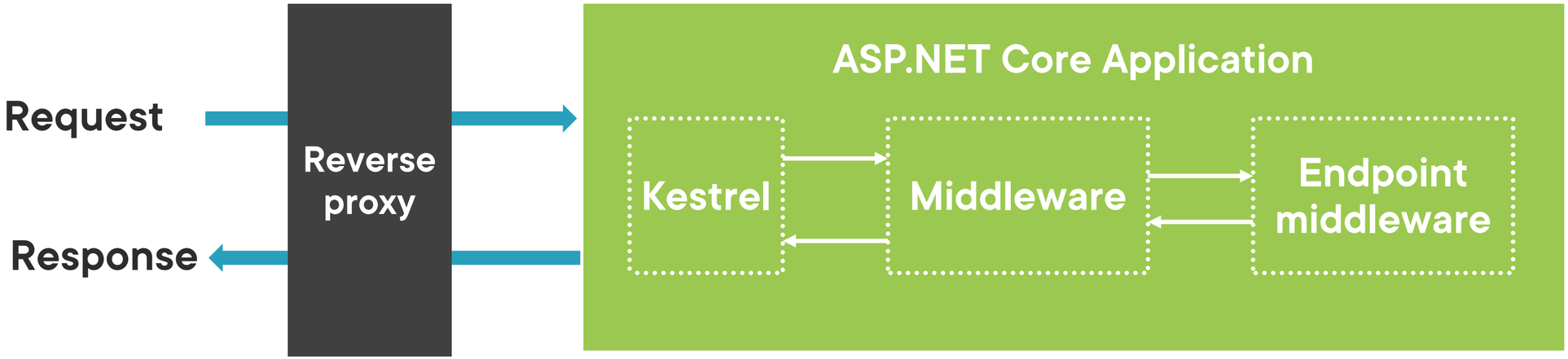
Custom services

```
public class OrderController : Controller
{
    private readonly ILoggerService _loggerService;

    public OrderController(ILoggerService loggerService)
    {
        _loggerService = loggerService;
    }
}
```

Using Services

Injected via constructor

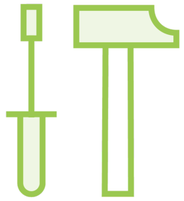
Handling Requests with Middleware



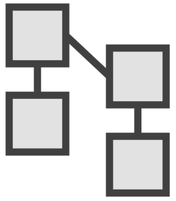
Middleware will create response based on incoming request

The Middleware Request Pipeline

Pipeline consists out of set of components



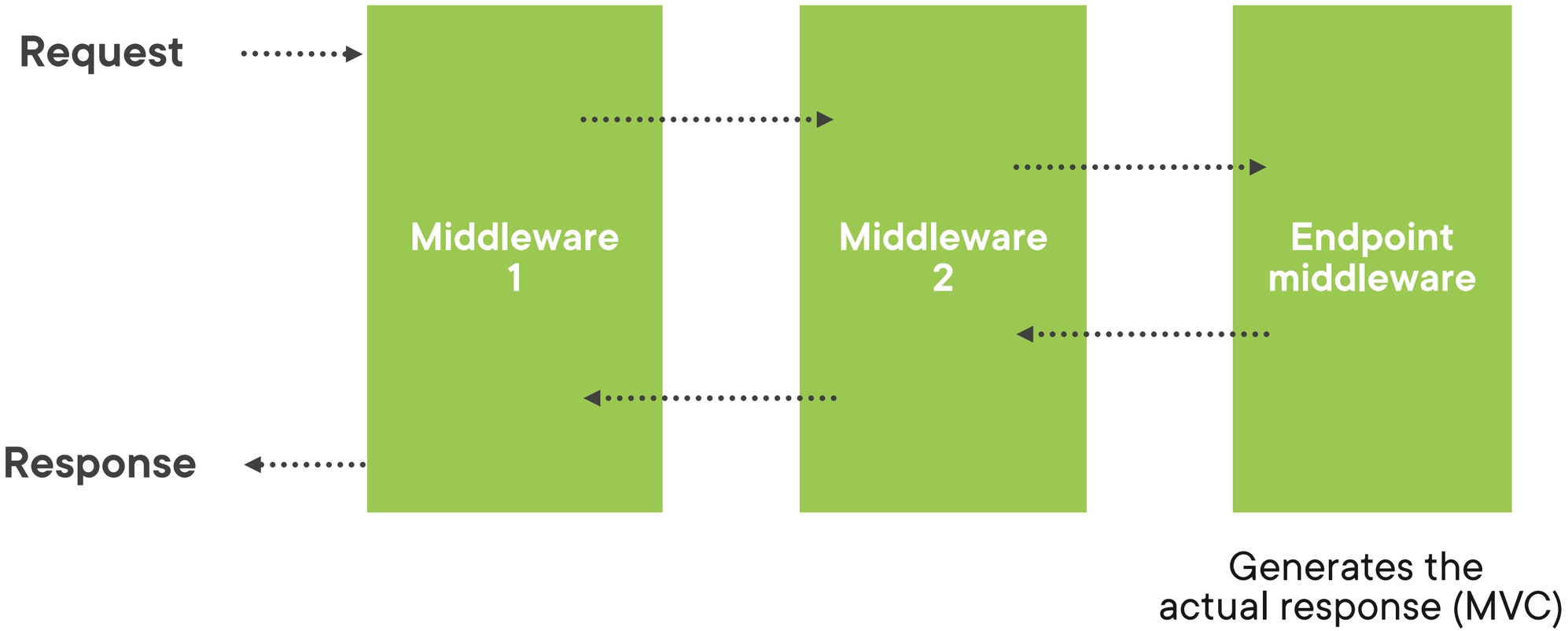
Components work on request or response



Many built-in components

Endpoint middleware sits at the end

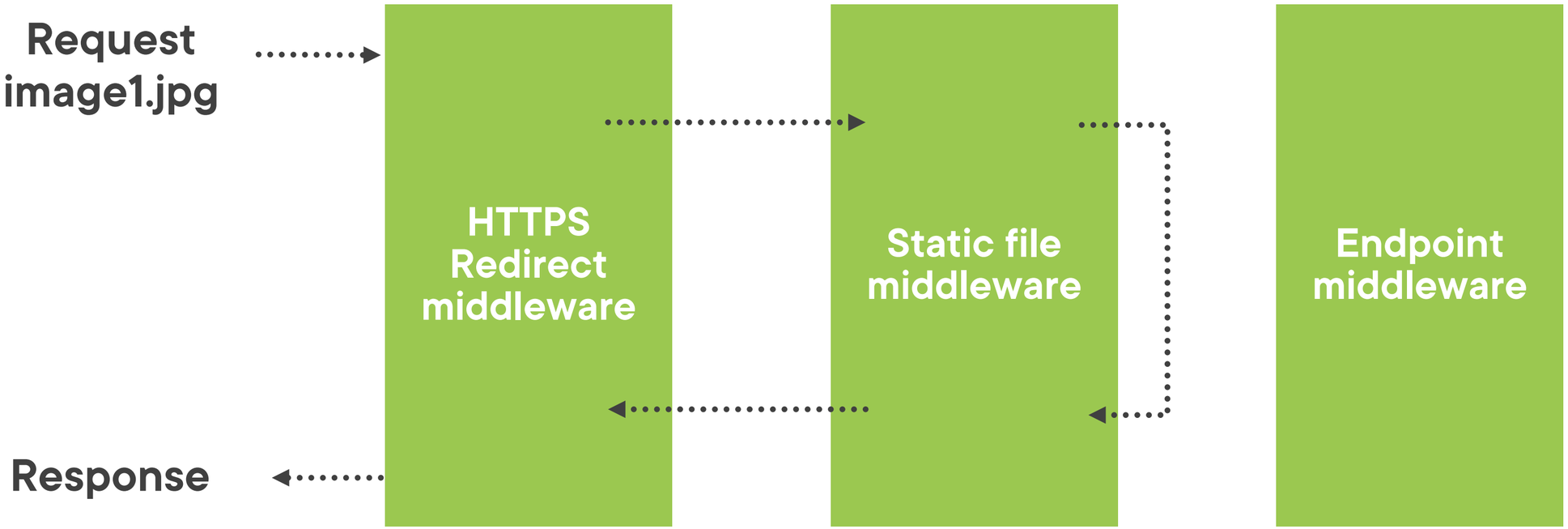
The Middleware Request Pipeline



Middleware Request Pipeline

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Exploring Static File Middleware



The order we add the components in, will be the order of the components in the pipeline!

Program.cs

Service registration

Middleware

What About the Old Model?

Program.cs

```
public class Program
{
  public static void Main(string[] args)
  {
    CreateHostBuilder(args).Build().Run();
  }

  public static IHostBuilder
    CreateHostBuilder(string[] args) =>
      Host.CreateDefaultBuilder(args)
      .ConfigureWebHostDefaults(webBuilder =>
      {
        webBuilder.UseStartup<Startup>();
      });
}
```

Startup.cs

```
public class Startup
{
  public void ConfigureServices(IServiceCollection
    services)
  {
    services.AddControllersWithViews();
  }

  public void Configure(IApplicationBuilder app,
    IWebHostEnvironment env)
  {
    ...
  }
}
```

Demo

Configuring the application

Summary

- Program class starts up application
- ASP.NET Core comes with built-in web server (Kestrel)
- Dependency injection is used by default
- Handling requests is done through middleware

Up next:
Creating your first page