Introductionto Asp.Net Core

Jim Fawcett

CSE686 – Internet Programming

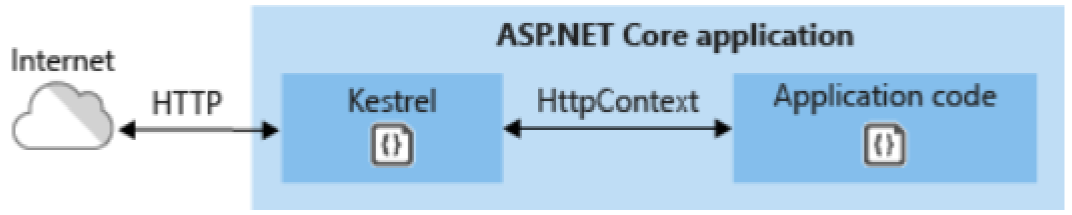
Spring 2018

Introduction

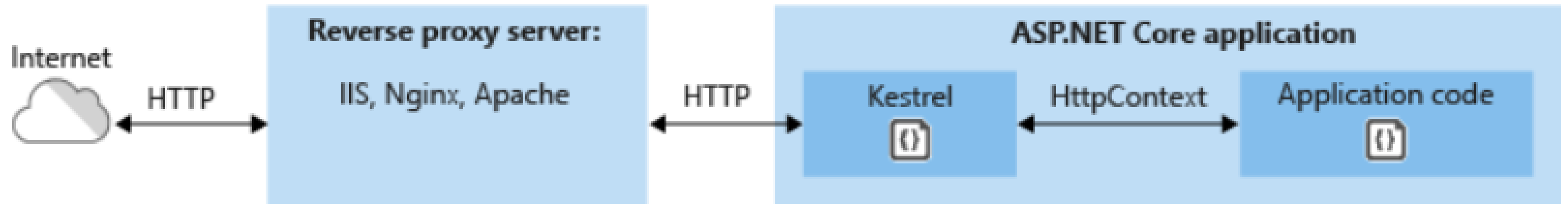
- Asp.Net Core provides a framework for building and executing both Console and Web Applications
- The 2.1 framework provides a host, responsible for startup and lifetime management.
  - Generic Host – host non-web apps
    - Windows services and executables
  - Web Host – suitable for hosting web applications
    - Create instance with IWebHostBuilder
  - Primary focus is web applications
  - It provides a pluggable hosting environment that supports:
    - Kestral, IIS, Apache, Nginx

Web Application Hosting Options

- Kestral



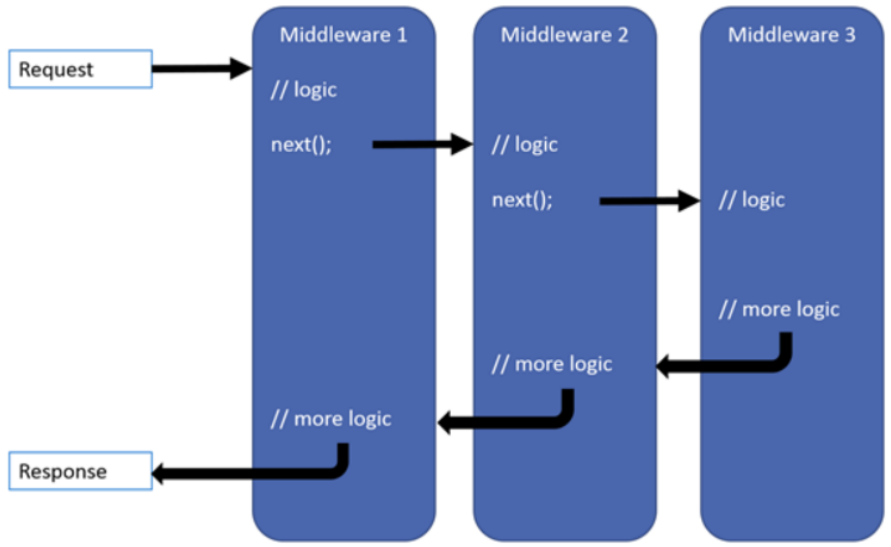
- IIS, Apache, Nginx



Diagrams from https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/?view=aspnetcore-2.2&tabs=windows

Asp.Net Core Pipeline

- Provides an application pipeline that supports pluggable services
- Pipeline services are delivered via a Dependency Injection Container
- The pipeline is configured with one or more components.



```
Middleware components pass Requests
to next component via Request
Delegates.

Each component configures a lambda
that binds to a Request Delegate,
defining its processing and invoking
a next() function.

When a Request arrives the middleware
delegate sequence is invoked.
```

Diagram from https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-2.2

Middleware

- Middleware is software that's assembled into an app pipeline to handle requests and responses. Each component:
  - Chooses whether to pass the request to the next component in the pipeline.
  - Can perform work before and after the next component in the pipeline.
- Request delegates are used to build the request pipeline. The request delegates handle each HTTP request.
- The points, above, are taken from:

  https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-2.2

Configuring Middleware

- You configure pipeline middleware using the Configure method, provided by the Startup class.
  - Services include:
    - Serving static files
    - MVC routing and operations
    - Custom services
  - Service lifetime:
    - AddSingletonService<IService, Service>()
      Singleton service used for the lifetime of the Application
    - AddScopedService<IService, Service>()
      Singleton service used for the duration of one HTTP request
    - AddTransientService<IService, Service>()
      Created with each request for service, possibly many times per HTTP request

Middleware provided by the framework

- partial list

- Authenication
- Cookie Policy
- CORS
- Diagnostics
- HTTPS Redirection
- MVC
- Routing
- Session
- Static Files
- URL Rewriting
- WebSockets

Startup

- Startup Class
  - ConfigureServices method
    - Registers a service interface and implementing class for dependency injection using one of the AddService methods, described in the previous slide
    - Each AddService adds a service to the Dependency Injection Services container.
  - Configure method
    - Creates the application’s pipeline with app.UseXXX() invocations.
    - app.Run(some write method)
    - An app.UseXXX invocation need not pass a message down the pipeline.
    - App.Run executes only if all app.UseXXX() middleware pass along the request message.
    - Essentially, the pipeline is the sequence of app.UseXXX() methods in StartupConfigure()

Building Web Host

- IWebHostBuilder CreateWebHostBuilder(string[] args)
  - Creates a host and defines the Startup Assembly
- IWebHostBuilder methods:
  - Build()
  - …
- Extension methods:
  - Start(IWebHostBuilder, String[])
  - UseConfiguration(IWebHostBuilder, IConfiguration)
  - UseServer(IWebHostBuilder, IServer)
  - UseStartup(IWebHostBuilder, String)
  - …

app.Run

- app.Run(…) is similar to app.UseXXX(), but does not send on a request. It is the pipeline terminus.

That’s All Folks