Kestrel Server and Reverse Proxy Servers

Kestrel Server

Overview:

- Kestrel is the cross-platform web server for ASP.NET Core.
- It is lightweight and suitable for serving dynamic content.

Responsibilities:

- **HTTP Requests Handling:** Handles incoming HTTP requests and responses.
- **Hosting:** Hosts the ASP.NET Core application.
- **Configuration:** Supports various configurations such as HTTP/2, HTTPS, etc.,

Use Case:

- Ideal for development and internal networks.
- Typically used in conjunction with a reverse proxy for production environment.

Reverse Proxy Servers

Overview:

- A reverse proxy server forwards client requests to backend servers and returns the responses to the clients.
- Common reverse proxy servers include Nginx, Apache, and IIS.

Responsibilities:

- **Load Balancing:** Distributes incoming requests across multiple servers.
- **SSL Termination:** Handles SSL/TLS encryption and decryption.
- **Caching:** Catches responses to improve performance.
- **Security:** Provides additional security features like request filtering, IP whitelisting, and rate limiting.

Use Case:

- Used in front of Kestrel to enhance security, load balancing, and other enterprise-level requirements.

Responsibilities of Kestrel and Reverse Proxy Servers

Kestrel:

- Serves HTTP requests directly.
- Provides efficient request processing.
- Should be used behind a reverse proxy for additional security and stability.

Reverse proxy:

- Acts as an intermediary between clients and Kestrel.
- Provides SSL termination, load balancing and security features.
- Enhances the overall performance and security of the application.

Summary of launchsettings.json:

- launchsettings.json configures how an ASP.NET Core application is launched during development.
- It can define multiple profiles, each with its own settings for URLs, environment variables, and launch options.
- The iisSettings section configures IIS Express settings, while the profiles section defines different launch profiles for the application.

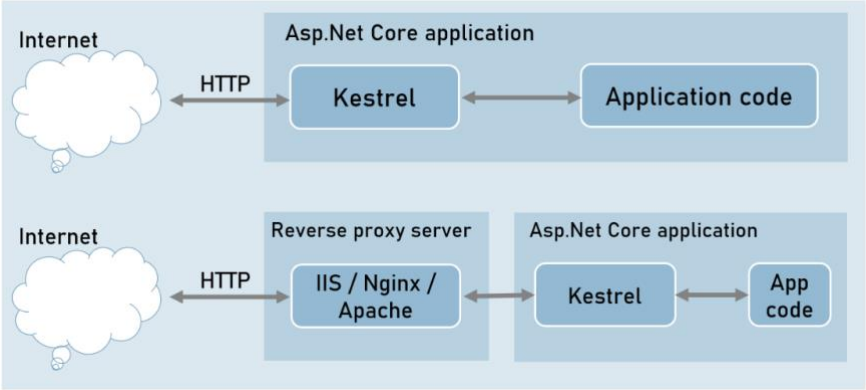
Kestrel and Other Servers

Application Servers

- Kestrel

Reverse Proxy Servers

- IIS
- Nginx
- Apache



Benefits of Reverse proxy Servers

- Load Balancing
- Caching
- URL Rewriting
- Decompressing the requests
- Authentication
- Decryption of SSL certificates

IIS Express

- HTTP access logs
- Post sharing
- Windows authentication
- Management console
- Process activation
- Configuration API
- Request filters
- HTTP redirect rules

Code

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

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

app.Run();
```

- The code creates and configures a minimal ASP.NET Core web application.
- WebApplication.CreateBuilder(args) sets up the application with default settings.
- builder.Build() finalizes the configuration and prepares the application.
- App.MapGet("/", () => "Hello World!"); maps a get request to the root URL and returns "Hello World!" as a response.
- App.Run() starts the web server and runs the application, ready to handle incoming requests.

What is Kestrel and what are advantages of Kestrel in ASP.NET Core?

ASP.Net Core application uses Kestrel by default.

Kestrel is an event-driven, I/O-based, open-source, cross-platform and asynchronous server which hosts ASP.Net applications. It is provided as a default server for .NET Core; therefore, it is compatible with all the platforms and their versions which .NET Core supports.

It is a listening server with a command-line interface.

It can be used to reverse proxy servers such as IIS, Nginx etc.,

Features of Kestrel are:

- Lightweight and fast.
- Cross-platform and supports all versions of .NET Core.
- Supports HTTP and HTTPS.
- Easy configuration.
- Multiple apps on same port is not supported.
- Windows authentication is not supported.

What is the difference between IIS and Kestrel? Why do we need two web servers?

The main difference between IIS and Kestrel is that Kestrel is a cross-platform server. It runs on Windows, Linux and Mac, whereas IIS runs only on Windows.

Another essential difference between the two is that Kestrel is fully open-source, whereas IIS is closed-source and deployed and maintained only be Microsoft.

IIS is very old software and comes with a considerable legacy and bloat. With Kestrel, Microsoft started with high-performance in mind. They developed it from scratch, which allowed them to ignore the legacy/compatibility issues and focus on speed and efficiency.

However, Kestrel doesn't provide all the rich functionality of a full-fledged web server such as IIS, Nginx, or Apache. Hence, we typically use it as an application server, with one of the above servers acting as a reverse proxy.

What is the purpose of launchSettings.json in asp.net core?

The launchSettings.json file is used to store the configuration information, which describes how to the ASP.NET Core application, in Visual Studio.

It mainly contains the run time profiles to configure application urls and environment.

The file is used only during the development of the application using Visual Studio. It contains only these settings that are required to run the application.

launchSettings.json is only used by Visual Studio.

You don't need launchSettings.json for publishing an app (on production server).

What is generic host or host builder in .NET Core?

.NET generic host called 'HostBuilder' helps us to manage all the below tasks:

- Dependency Injection
- Service lifetime management
- Configuration
- Logging

The generic host was previously present as 'Web Host', in .NET Core for web applications. Later, the 'Web Host' was deprecated and a generic host was introduced to cater to the web, Windows, Linux, and console applications.

What is the purpose of the .csproj file?

The project file is one of the most important files in our application. It tells .NET how to build (compile) the project.

The csproj file stores list of package dependencies of the current project, target .net version and other compilation settings.

The .csproj file also contains all the information that .NET tooling needs to build the project. It includes the type of project you are building (console, web, desktop etc.), the platform this project targets, and any dependencies on other projects or 3rd party libraries.
Here is an example of a .csproj file that lists the NuGet packages and their specific versions.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
<PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
</PropertyGroup>

<ItemGroup>
<PackageReference Include="PackageName" Version="1.0.0.0">
</ItemGroup>
</Project>
```

What is IIS?

IIS stans for Internet Information Services. It is a powerful web server developed by Microsoft. IIS can also act as a load balancer to distribute incoming HTTP requests to different application servers to allow high reliability and scalability.

It can also act as a reverse proxy, i.e. accept a client's request, forward it to an application server, and return the client's response. A reverse proxy improves the security, reliability, and performance of your application.

A limitation of IIS is that it only runs on Windows. However, it is very configurable. You can configure it to suit your application's specific needs.

What is the "Startup" class in ASP.NET core prior to to ASP.NET Core 6?

The startup class is the entry point of the ASP.NET Core application. Every ASP.Net Core (ASP.NET Core 5 and earlier) application must have this class. It contains the necessary code to bootstrap the application. This class contains two methods Configure() and ConfigureServices().

- Configure(): The Configure() method is used to essential middleware(s) to the application request pipeline.
- ConfigurationServices(): The ConfigurationServices() method is used to add services to the IoC container.

In ASP.NET Core 6 (.NET 6), Microsoft unifies startup.cs and Program.cs into a single Program.cs.

In asp.net core 6 (and up), we need to add all such as registering middleware to the application pipeline, adding services to the IoC container, configuring the 'application configuration', configuring the logger, authentication and adding DbContext in Program.cs file.

What does WebApplication.CreateBuilder() do?

This method does the following things:

- Configure the app to use Kestrel as web server.
- Specify to use the current project directory as root directory of the application.
- Setup the configuration sub-system to read settings from appsettings.json and appsetting.{env}.json to environment specific configuration.
- Set Local user secrets storage only for the development environment.
- Configure environment variables to allow for server-specific settings.
- Configure command line arguments (if any).
- Configure logging to read from the logging section of the appsettings.json file and log to the Console and Debug window.
- Configure integration with IIS.
- Configure the default service provider.