TEKTUTORIALSHUB

Kestrel: Web Server for ASP.NET Core

1 Comment

← Application Startup

Middleware & Request Pipeline →

In this tutorial, let us look at the `Kestrel` Web Server for ASP.NET Core. The way we host our application in ASP.NET Core has gone through some drastic change compared to the previous version of ASP.NET. The `Kestrel` is the new default web server that is included in the ASP.NET Core project templates. It runs within the application process making it completely self-contained.

Table of Content

What is Kestrel?
Why Kestrel
Using Kestrel
There are two ways you can use the Kestel
Self Hosting
Behind another Webserver

Benefits of Kestrel Web Server
Alternatives to Kestrel
Summary

What is Kestrel?

The `Kestrel` is open-source, cross-platform, event-driven, asynchronous I/O based `HTTP server`. It is developed to host ASP.NET Core applications on any platform. It is included by default in the ASP.NET Core applications.

It is based on libuv

`Kestel` is an Open source library and is available at GitHub

Why Kestrel

The older ASP.NET applications are tightly coupled to the Internet Information Services or `IIS`.

The IIS is a complete web server with all the features that you re Web Server. Over the period it has grown into a matured web server and along the way, it has a become one of the best Web servers around and at the same tim

ASP.NET being tightly coupled with `IIS` carried the burden of the

The newly designed ASP.NET Core applications are now complet makes ASP.NET Core run on any platform making it truly cross-p

Valueimpression

But, it still needs to have the ability to listen to `HTTP` requests and send the response back to the Client. That is where `Kestrel` comes in

Using Kestrel

The `Kestrel` runs in in-process in the ASP.NET Core Applications. Hence, It runs independently of the environment in which it lives. The kestrel Web server is available in the namespace `Microsoft.AspNetCore.Server.Kestrel`

We looked at the `program` and `startup` class in the tutorial Application startup in ASP.NET Core

The `Program` class contains a `static void Main` function, which is the entry point to our Application.

```

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

    public static IWebHost BuildWebHost(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>()
        .Build();
}
```

Valueimpression

The `Main` method invokes `CreateDefaultBuilder`, which is responsible to create the web application host.

The `CreateDefaultBuilder` is a helper method (click here for the source code)

It calls the `UseKestrel` method to registers the `Kestrel` as the server that will be used to host our application

There are two ways you can use the Kestel

1. By Itself (Self Hosting)
2. Behind another Webserver

Self Hosting

Under Sef Hosting model the ASP.NET Core applications directly listen to the `HTTP` Requests from the internet as shown in the image below.

Valueimpression

The diagram shows how Kestrel Web server is used to self-host an ASP.NET Core Application

The following image shows how to use dotnet run cli command t
need to do is to go to the project folder, where .csproj file is four

```
dotnet run
```

Valueimpression

The kestrel starts and listens on port 5000/5001.

```
Command Prompt - dotnet run

D:\TekTutorialsHub\ASPNetCore\GettingStarted\HelloWolrd\HelloWolrd>dotnet run
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[0]
      User profile is available. Using 'C:\Users\BHOOMI\AppData\Local\ASP.NET\DataProtection-Keys' as key repository and
 Windows DPAPI to encrypt keys at rest.
Hosting environment: Development
Content root path: D:\TekTutorialsHub\ASPNetCore\GettingStarted\HelloWolrd\HelloWolrd
Now listening on: https://localhost:5001
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Another way is to go to the published folder and run the following command, which will fire up the kestrel server.

```
dotnet helloWorld.dll
```

Behind another Webserver

`Kestrel` is not a fully-featured web server. But that is what makes i

It is not advisable to run `Kestrel` as a standalone web server in the
recommended to run it behind a Fully Featured Web server like I
Web server acts as a `reverse proxy server`

Valueimpression

The reverse proxy server takes the HTTP request from the internet and passes it to the `kestrel` server just the way it is received.

The `IIS` can take the HTTP request and perform some useful processing like `logging`, `request filtering`, `URL rewrites` before passing the request to `Kestrel`.

The following diagram shows how it is implemented

Kestrel Web server running behind another W

Apache, etc

Valueimpression

There are many reasons why you should use this model in the production

1. Security
2. It can limit your exposed surface area. It provides an optional additional layer of configuration and defense.
3. Simplifies Load Balancing
4. SSL Setup
   Only your reverse proxy server requires an SSL certificate, and that server can communicate with your application servers on the internal network using plain HTTP.
5. Sharing Single IP with Multiple addresses
6. Request FIltering, logging & URL Rewrites, etc.
7. It can make sure that the application restarts if it is crashes

The `CreateDefaultBuilder` method calls the `UseIISIntegration`, which tells ASP.NET that the application will be using the `IIS` as a reverse proxy in front of `Kestrel`

Benefits of Kestrel Web Server

1. Kestrel is fast. It is not a fully-featured website and does n
   from a standard web server. Hence it makes it lightweight
2. Is supports all the versions of .NET Core
3. It is cross-platform. You can run it on Windows/Linux or M
   always

Valueimpression

4. It is very simple to configure and run. In fact, it is already configured, when you create a new ASP.NET Core project in Visual Studio
5. Supports HTTPS
6. Supports HTTP/2 (except on macOS. But will be supported in a future release).

Alternatives to Kestrel

The `Kestrel` is not the only way to host ASP.NET Core applications. There is another webserver implementation available in Windows known as `HTTP.SYS`

The HTTP.sys is a Windows-only HTTP server based on the Http.Sys kernel driver.

Here is a nice tutorial on How to use HTTP sys.

Summary

In this tutorial, we learned about the `Kestrel` Web server. Kestrel is an In-Process web server built to run ASP.NET Core applications on any Platform.

← Application Startup

Valueimpression

ought on “Kestrel: Web Server for ASP.NET Core”

ADEL

MAY 8, 2018 AT 7:05 PM

Thanks for this tutorials, very interesting,

just to ask if you can put the date for your posts please.

thanks

Reply

Leave a Comment

Your email address will not be published.

Valueimpression

Type here..

Name*

Email*

Website

Post Comment »

Valueimpression

[redacted]

T site uses Akismet to reduce spam. Learn how your comment

Valueimpression

Copyright © 2020 TekTutorialsHub |

Valueimpression