**ASP.NET Core appsettings.json file**

**Configuration Sources in ASP.NET Core**  
  
In previous versions of ASP.NET, we store application configuration settings, like database connection strings for example, in **web.config**file. In ASP.NET Core application configuration settings can come from the following **different configurations sources.**

* Files (appsettings.json, appsettings.{Environment}.json, where {Environment} is the app's current hosting environment)
* User secrets
* Environment variables
* Command-line arguments

**appsettings.json file :**In the asp.net core project that is generated by the **"Empty"** project template we already have a file with name **appsettings.json**. I have modified this file to include a new setting with the key - **MyKey**.

{

  "Logging": {

    "LogLevel": {

      "Default": "Warning"

    }

  },

  "AllowedHosts": "\*",

  "MyKey": "Value of MyKey from appsettings.json"

}

**Accessing configuration information**  
  
To access configuration information in the **Startup**class, inject the IConfiguration service provided by the Framework. Startup class is in Startup.cs file.

public class Startup

{

    private IConfiguration \_configuration;

    // Notice we are using Dependency Injection here

    public Startup(IConfiguration configuration)

    {

        \_configuration = configuration;

    }

    public void ConfigureServices(IServiceCollection services)

    {

    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)

    {

        if (env.IsDevelopment())

        {

            app.UseDeveloperExceptionPage();

        }

        app.Run(async (context) =>

        {

            await context.Response.WriteAsync(\_configuration["MyKey"]);

        });

    }

}

**Dependency Injection**  
  
In previous versions of ASP.NET Dependency Injection was optional and to configure it we have to use frameworks like Ninject, StructureMap etc.**In ASP.NET Core Dependency Injection is an integral part.** Dependency Injection allow us to create systems that are loosely coupled, extensible and easily testable.

**ASP.NET Core IConfiguration service**

* IConfiguration service is setup to read configuration information from all the various configuration sources in asp.net core
* If you have a configuration setting with the**same key in multiple configuration sources**, the later configuration sources override the earlier configuration sources
* CreateDefaultBuilder() method of the WebHost class which is automatically invoked when the application starts, reads the configuration sources in a specific order.
* To see the order in which the configuration sources are read, please check out ConfigureAppConfiguration() method on the following link  
  <https://github.com/aspnet/MetaPackages/blob/release/2.2/src/Microsoft.AspNetCore/WebHost.cs>

Upon inspecting the file, you will see, the following is the default order in which the various configuration sources are read

1. appsettings.json,
2. appsettings.{Environment}.json
3. User secrets
4. Environment variables
5. Command-line arguments

You can change this order if you want to or even add your own custom configuration sources in addition to all the existing configuration sources.