**ASP.NET Core launchsettings.json file**

**launchsettings.json file**

* You will find this file in the **"Properties"** folder in the project root folder.
* The settings in this file are used when we run this ASP.NET core project either from Visual Studio or by using .NET Core CLI.
* This file is **only used on local development machine**. We do not need it for publishing our asp.net core application.
* If there are certain settings that you want your asp.net core application to use when you publish and deploy your app, store them in **appsettings.json file**. We usually store our application configuration settings in this file.
* We can also have **environment specific appsettings.json files**. For example, appsettings.Staging.json for the staging environment. In ASP.NET Core, in addition to appsettings.json file, we also have other configuration sources like Environment variables, User Secrets, Command Line Arguments and even our own custom configuration source.

**Lanuch Profiles in ASP.NET Core**   
  
At the moment, the following are the settings in **launchSettings.json file**

{

  "iisSettings": {

    "windowsAuthentication": false,

    "anonymousAuthentication": true,

    "iisExpress": {

      "applicationUrl": "http://localhost:48118",

      "sslPort": 0

    }

  },

  "profiles": {

    "IIS Express": {

      "commandName": "IISExpress",

      "launchBrowser": true,

      "environmentVariables": {

        "ASPNETCORE\_ENVIRONMENT": "Development"

      }

    },

    "EmployeeManagement": {

      "commandName": "Project",

      "launchBrowser": true,

      "environmentVariables": {

        "ASPNETCORE\_ENVIRONMENT": "Development"

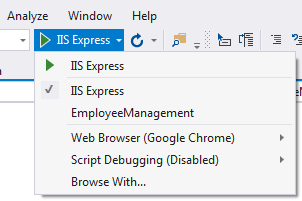
      },

      "applicationUrl": "http://localhost:5000"

    }

  }

}

Notice, we have 2 profiles - **IIS Express**and **EmployeeManagement**   
  
When we run the project from Visual Studio by pressing **CTRL + F5** or just **F5**, by default, the profile with "commandName": "IISExpress" is used. On the other hand, if we run the project using .NET Core CLI (dotnet run), the profile with the  "commandName": "Project" is used.    
  
However, we can change which profile to use by clicking on the dropdownlist in Visual Studio   
  
   
  
The value of the **commandName**property can be any one of the following.

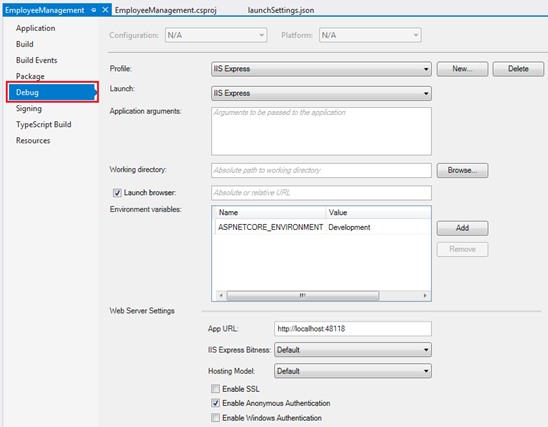
* Project
* IISExpress
* IIS

This value along with the value of **AspNetCoreHostingModel**element in the project file, specifies the internal and external web server (reverse proxy server) to launch.

|  |  |  |  |
| --- | --- | --- | --- |
| **commandName** | **AspNetCoreHostingModel** | **Internal Web Server** | **External Web Server** |
| Project | Hosting Setting Ignored | Only one web server is used - Kestrel | |
| IISExpress | InProcess | Only one web server is used - IIS Express | |
| IISExpress | OutOfProcess | Kestrel | IIS Express |
| IIS | InProcess | Only one web server is used - IIS | |
| IIS | OutOfProcess | Kestrel | IIS |

You can change the settings in **launchSettings.json**file by directly editing the file or we can also change the settings using the Graphical User Interface (GUI) provided by Visual Studio.   
  
**To access the GUI**

* Right click on the project name in Solution Explorer in Visual Studio and select **"Properties"** from the context menu.
* Click on the **"Debug"**tab on the project **"Properties"** window

   
  
Using the GUI we can change the settings in **launchSettings.json file**. Notice, the Environment Variable **"ASPNETCORE\_ENVIRONMENT"** is set to **"Development"**. We can change this value to **Staging**or **Production**depending on whether we are running this project in Staging or Production environment.    
  
We can also add new environment Variables. These environment variables are available throughout our asp.net core application and we can include code that conditionally executes depending on the value of these environment variables.   
  
**For example**, consider the following code in the **Configure()** method in **Startup.cs** file

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

    if (env.IsDevelopment())

    {

        app.UseDeveloperExceptionPage();

    }

    // Rest of the code...

}

**Developer Exception Page** is only displayed if the environment is **Development**. We will discuss environment variables and developer exception page in detail in our upcoming sessions.