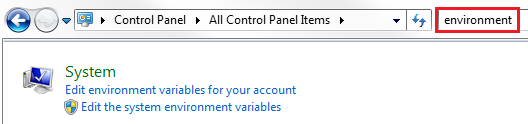
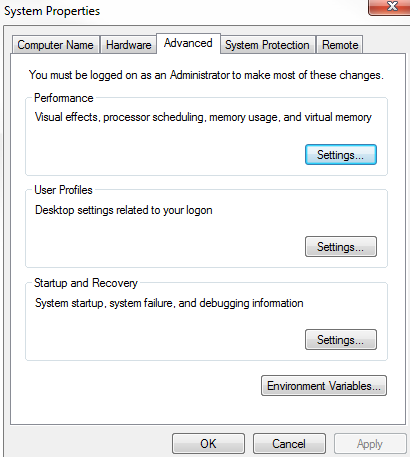
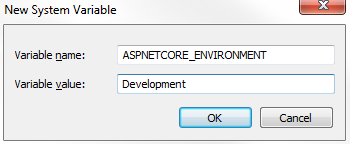
**ASP.NET Core environment variables**

**Software Development Environments**   
  
In most software development organisations we typically have the following development environments.

* Development
* Staging
* Production

**Why do we need different Development Environments like Development, Staging, Production etc.**   
  
**Development Environment :**We software developers typically use this environment for our day to day development work. We want non-minified JavaScript and CSS files to be loaded on a development environment for ease of debugging. Similarly we want developer exception page if there is an unhandled exception so we can understand the root cause of the exception and fix it if required.   
  
**Staging Environment :**Many organisations, try to keep their staging environment as identical as possible to the actual production environment. The primary reason for this environment is to identify any deployment related issues. Also, if you are developing a B2B (Business to Business) application, you may be interfacing with other service provider systems. Many organisations, usually setup their staging environment to interface with the service providers as well, for complete end to end testing. We usually do not troubleshoot and debug on a staging environment, so for better performance we want minified JavaScript and CSS files to be loaded. If there is an unhandled exception, display user friendly error page instead of the developer exception page. A user friendly error page will not contain any technical details. It contains a generic message like the following    
  
"Something has gone wrong, please email, chat or call our application support using the contact details below"   
  
**Production Environment :**The actual live environment, that we use for day to day business. Production environment should be configured for maximum security and performance. So load minified JavaScript and CSS files to improve the performance. For better security, display a User Friendly Error Page instead of the Developer Exception Page. The technical details on the Developer Exception Page does not make sense to the end user and they can be used by malicious user to break into your application.   
  
**Configuring ASPNETCORE\_ENVIRONMENT variable**  
-We use this variable to set the environment for our application. On our local development machine we usually set this environment variable in **launchsettings.json**file.  
  
We can also set it in the operating system if we want to. To set it on a Windows Operating system

1. Open Windows Control Panel
2. In the Control Panel window, type **"environment"** in the **"Search Control Panel"**text box on the top right hand corner
3. Click on **"Edit the system environment variables"** link   
   
4. In the **"System Properties"** window that pops up, click on **"Environment Variables"** button   
   
5. On the **"Environment Variables"** window that pops up, click on **"New"** button under **"System variables"** section
6. In the **"New System Variable"** window that pops up, type
   * ASPNETCORE\_ENVIRONMENT in "Variable name" text box
   * Development in "Variable value" text box   
     
7. Click **"OK"** to close all the pop up windows

On a **staging**or **production**environment we typically set this environment variable in the operating system.   
  
We usually set this variable to one of the following values depending on the environment in which our application is hosted and running.

* Development
* Staging
* Production

**Accessing ASPNETCORE\_ENVIRONMENT variable value**  
  
Out of the box, ASP.NET core provides IHostingEnvironment service which we can use to access ASPNETCORE\_ENVIRONMENT variable value. Take a look at the sample application we have been working with, notice the Configure() method in Startup.cs file. IHostingEnvironment service is already injected into this method.  
  
Now modify the code in Configure() method as shown below. Notice, we are using EnvironmentName property of IHostingEnvironment service instance to access the environment name.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)  
{  
    if (env.IsDevelopment())  
    {  
        app.UseDeveloperExceptionPage();  
    }  
  
    app.UseStaticFiles();  
  
    app.Run(async (context) =>  
    {  
        await context.Response.WriteAsync("Hosting Environment: " + env.EnvironmentName);  
    });  
}

If you have the environment variable set in both the places i.e launchsettings.json file and in the operating system, then the values in launchsettings.json file overrides the value specified at the operating system level.   
  
**Please note :**Please restart Visual Studio, if you do not see the Environment Variable value set at the operating system level   
  
If we have not explicitly set ASPNETCORE\_ENVIRONMENT variable, it defaults to **Production**. This is done on purpose for better security and performance.    
  
Imagine, on a production server, we have forgot to set the ASPNETCORE\_ENVIRONMENT variable to Production.If the default value is Development, the application may display Developer Exception Page and it could be exploited by a malicious user to hack into your application.    
  
Also, instead of loading minified JavaScript and CSS files, the non-minified files may be loaded. So for better performance and security it defaults to a value of Production if we have not explicitly set the ASPNETCORE\_ENVIRONMENT variable.   
  
**Useful methods of IHostingEnvironment service**   
  
Use the following methods of IHostingEnvironment service to identify the environment in which our application is running.

* IsDevelopment()
* IsStaging()
* IsProduction()

What if you have a custom environment like UAT (User Acceptance Testing) or QA (Quality Assurance) environment. Well, custom environments like these are also supported in ASP.NET core. For example, to check if the environment is UAT, use **IsEnvironment()**method as shown below.   
  
env.IsEnvironment("UAT")

// If the environment is Development serve Developer Exception Page  
if (env.IsDevelopment())  
{  
    app.UseDeveloperExceptionPage();  
}  
// else serve User Friendly Error Page with Application Support Contact Info  
else if (env.IsStaging() || env.IsProduction() || env.IsEnvironment("UAT"))  
{  
    app.UseExceptionHandler("/Error");  
}

**Tag Helpers are new in ASP.NET Core.** In a razor view ie. in a .CSHTML page we can use, something called an Environment Tag Helper. This environment tag helper supports rendering different content depending on the value of the ASPNETCORE\_ENVIRONMENT variable. We will discuss Tag Helpers including the environment tag helper in detail as we progress through this course and create **Models**, **Views**and **Controllers**for our application.