

Conditional compiling using C# preprocessor directives, Build Events and Batch Build

I was playing around VS2008 finding a way to speed up my job and following a request of Jared I needed to split my monolithic plugin in two parts.

I evaluated various possibilities and then i dediced to try to use the **C# preprocessor directives** to be able to build two different version of my plugin without splitting the project in two.

I'd like to share my experience here

In my project I had the need to have a DEBUG mode that instead of creating the XML and the log file in the location expected by the Healhcheck viewer creates them in the same location from the where the EXE is run.

The second need was to write in the output XML different sections depending on the plugin type.

To achieve the two goals I defined two Conditional compilation symbols: PLATFORM and ALPHATEST and then I started to use the C# preprocessor directives `#if`, `#elif` in this way:

```
#if (LDMS)
    private const string logName = @"\myldms.log";
    private const string xmlName = @"\myldms.xml";
    private const string dirOutName = @"CoreInfo";
#elif (PLATFORM)
    private const string logName = @"\myos.log";
    private const string xmlName = @"\myos.xml";
    private const string dirOutName = @"PlatformInfo";
#endif
```

So if LDMS is defined (in the Build tab of the project properties) some part the code are compiled, if PLATFORM is defined other parts are compiled and so on...

But what happens if no symbols are defined or both of them are??

We need some control here that can be achieved with the **#error** directive

```
#if (!LDMS && !PLATFORM)
    #error Defining LDMS or PLATFORM compiler directive is mandatory
#endif

#if (LDMS && PLATFORM)
    #error Defining LDMS and PLATFORM as the same time is not permitted
#endif
```

Seems quite nice but it was a bit annoying to define manually the symbols, build the project, rename the executables and so on for 4 times...

So I started to use the **Batch Build, Configuration Manager** and **Build Events** feature of VS2008

I had then two goals in mind:

1. Avoid to manually define the symbols and build every time.
2. Avoid to manually rename the output file (the EXE) every time.

Let's start with the **Configuration Manager**

1. From the Build menu' select the option **Configuration Manager**.
2. From the **Configuration** dropdown select the option **New**.
3. Copy the setting from the current configuration and give it a name (example: LDMS Plugin). Keep the checkbox "**Create new solution configurations**" checked and press OK.
4. Repeat the process for all the configurations you need (I needed four).

Now that we have many different configurations we need to 'configure' them with different **Conditional compilation symbols**

1. Go to the project properties in the **Build** tab.
2. Select from the dropdown the configuration you want to set and then set the **Conditional compilation symbols** that you need.
3. Press **SAVE** and repeat the process for all your newly created configurations.

Now is the moment when **Build Events** kicks in

To be able to have the name of the EXE different for every build you need to play with the post-build event command line and its macros.

Do not worry about the macros syntax: in the "Edit Post-build" GUI a Macros button is available with all the macros you need (they are variables at the end) and their current value.

I used this command line in the post-build command:

```
rename "$(TargetPath)" "$(ConfigurationName)$$(TargetExt)"
```

The result is that the EXE will be renamed after the build phase with the configuration name and the .exe extension (es: *CoreInfoOS.exe*)

But how we can benefit from this doing all the four (or more) compilation at the same time obtaining then the full automation?

Select the option **Batch build** in the **Build** menu and select all the configuration that you want to build and then press the button **Build**

This article is quite a basic and fast introduction to this interesting topic. Please feel free to send me suggestion on how to enhance it or any question you have in mind.

This document was generated from the following discussion: [Conditional compiling using C# preprocessor directives, Build Events and Batch Build \(first part\)](#)