#ifdef in C#

Asked 12 years, 9 months ago Modified 1 year ago Viewed 141k times

122

I would like to do the below but in C# instead of C++

```
#ifdef _DEBUG
bool bypassCheck=TRUE_OR_FALSE;//i will decide depending on what i am debugging
#else
bool bypassCheck = false; //NEVER bypass it
#endif
```

12

c#

Share Improve this question Follow

asked Jun 10, 2009 at 12:49
user34537

Check **this excellent answer** as well, it shows how you can add debug symbols based on conditions via the project file (.csproj). – Matt May 12, 2017 at 13:35

4 Answers

Sorted by: Самый высокий балл (по умолчанию)

166

```
#if DEBUG
bool bypassCheck=TRUE_OR_FALSE;//i will decide depending on what i am debugging
#else
bool bypassCheck = false; //NEVER bypass it
#endif
```

Make sure you have the checkbox to define DEBUG checked in your build properties.

Share Improve this answer Follow

answered Jun 10, 2009 at 12:50
heavyd
16.8k 5 55 72

54

I would recommend you using the Conditional Attribute!

Update: 3.5 years later

You can use `#if` like this (example copied from MSDN):

```
// preprocessor_if.cs
#define DEBUG
#define VC_V7
using System;
public class MyClass
{
    static void Main()
    {
#if (DEBUG && !VC_V7)
        Console.WriteLine("DEBUG is defined");
#elif (!DEBUG && VC_V7)
        Console.WriteLine("VC_V7 is defined");
#elif (DEBUG && VC_V7)
        Console.WriteLine("DEBUG and VC_V7 are defined");
#else
        Console.WriteLine("DEBUG and VC_V7 are not defined");
#endif
    }
}
```

Если вы используете `#if` исключение какого-либо метода из компиляции, то вам придется исключить из компиляции все фрагменты кода, которые также вызывают этот метод (иногда вы можете загрузить некоторые классы во время выполнения и не можете найти вызывающего абонента с помощью "Найти все ссылки"). В противном случае будут ошибки.

С другой стороны, если вы используете условную компиляцию, вы все равно можете оставить все фрагменты кода, вызывающие этот метод. Все параметры по-прежнему будут проверены компилятором. **Метод просто не будет вызываться во время выполнения**. Я думаю, что гораздо лучше скрыть метод только один раз и не удалять весь код, который его вызывает. Вы не можете использовать условный атрибут для методов, которые возвращают значение - только для методов void. Но я не думаю, что это большое ограничение, потому что если вы используете `#if` метод, который возвращает значение, вы должны скрыть все фрагменты кода, которые его вызывают.

Вот пример:

```
// вызов Class1.ConditionalMethod() будет проигнорирован во время выполнения
// если не определена константа ОТЛАДКИ


использование системы.Диагностика;
класс Class1
{
[Условный("ОТЛАДКА")]
public static void ConditionalMethod() {
Консоль.WriteLine("Executed Class1.ConditionalMethod");
}
}
```

Краткие сведения:

Я бы использовал `#ifdef` в C++, но с C#/VB я бы использовал условный атрибут. Таким образом, вы скрываете определение метода без необходимости скрывать фрагменты кода, которые его вызывают. Вызывающий код по-прежнему компилируется и проверяется компилятором, однако метод не вызывается во время выполнения. Возможно, вы захотите использовать `#if` , чтобы избежать зависимостей, потому что с условным атрибутом ваш код все еще компилируется.

Поделиться Улучшите этот ответ Подписаться

отредактировано 30 декабря 2014 в 0:52 ответил 10 июня 2009 года в 12:52

Павел Николов
8 943 5 42 54

1 +1 Это действительно хорошо, но имеет ограничения, например, когда вы пытаетесь вернуть значение из условного метода (как я понимаю). Я думаю, что встроенный пример поможет. –Хэмиш Грубиджан 18 марта 2011 года в 23:05

1 Он также не препятствует компиляции кода, он просто не позволяет этому коду. Это различие важно, когда вы хотите удалить зависимости и тому подобное. –Ли Лувьер 7 декабря 2011 года в 22:16

В C# есть препроцессор. Он работает немного иначе, чем в C++ и C.

Вот ссылки MSDN - раздел, посвященный всем директивам препроцессора.

Поделиться Улучшите этот ответ Подписаться

отредактировано 13 июля 2012 в 11:35 ответил 10 июня 2009 года в 12:53

Сума
31.5 k 15 119 182

Карл Т.
492 3 3

11 It's a minor point, but C# does NOT have a preprocessor. # directives are processed by the main compiler as if there was a preprocessor. See here: msdn.microsoft.com/en-us/library/ed8yd1ha.aspx The main outcome of this distinction is that c/c++ style macros don't work. – Simon P Stevens Jun 10, 2009 at 13:19

I was able to achieve the behavior by adding the `<DefineConstants>` tag into the csproj's xml.

1. Open the project file with a text editor.
2. Find the first `<PropertyGroup>` with all of your project property definitions.
3. Add `<DefineConstants>SOME_VARIABLE_NAME</DefineConstants>` inside this group.
4. Add `#if SOME_VARIABLE_NAME` to your code to conditionally enable the code.

Someone has written up a much better explanation here : stackoverflow.com/a/43442076/2860267 – Kylaaa Nov 2, 2021 at 17:15