



Сборки, атрибуты, директивы.

Сборка

- **Сборка** является базовой структурной единицей в .NET, на уровне которой проходит контроль версий, развертывание и конфигурация приложения.

Состав сборки

- Манифест, который содержит метаданные сборки
- Метаданные типов. Используя эти метаданные, сборка определяет местоположение типов в файле приложения, а также места размещения их в памяти
- IL-код
- Ресурсы

MyApp.dll

Манифест сборки

Метаданные типов

MSIL-код

Ресурсы

Дополнительные сведения о сборке

- Сборки реализованы как EXE- или DLL-файлы.
- Сборки можно поместить в глобальный кэш сборок, чтобы обеспечить их использование несколькими приложениями. В глобальный кэш сборок могут быть включены только сборки со строгими именами.
- Сборки загружаются в память только по мере необходимости. Если сборки не используются, они не загружаются.
- Для программного получения сведений о сборках используется класс reflection.

Манифест сборки

- Ключевым компонентом сборки является ее **манифест**. Если у сборки отсутствует манифест, то заключенный в ней код MSIL выполняться не будет. Манифест может находиться в одном файле с исполняемым кодом сборки, а может размещаться и в отдельном файле.

Состав манифеста

- **Имя сборки**
- **Номер версии:** основной и дополнительный номера. Используется для управления версиями
- **Язык и региональные параметры:** информация о языке и региональных параметрах, которые поддерживает сборка
- **Информация о строгом имени:** открытый ключ издателя
- **Список всех файлов сборки:** хэш и имя каждого из входящих в сборку файлов
- **Список ссылок на другие сборки,** которые использует текущая сборка
- **Список ссылок на типы,** используемые сборкой

Директивы

- Директива – это указание компилятору языка C# выполнить то или иное действие в момент компиляции программы.

Список директив

- #define #include
- #error #using
- #import #else
- #undef #ifdef
- #elif #line
- #if #endif
- #ifndef
- #pragma

Размещение директивы

- Знак решетки (#) должен быть первым непробельным символом в строке, содержащей директиву; между знаком решетки и первой буквой директивы пробельные символы допускаются.
- Любой текст, следующий после директивы (за исключением аргумента или значения, представляющих собой часть директивы), должен предваряться разделителем однострочного комментария (//) или заключаться в разделители комментариев (/ * */).

Директива #region #endregion

- Директива **#region** позволяет указать блок кода, который можно разворачивать и сворачивать с помощью функции структурирования в редакторе кода Visual Studio.

```
#region MyClass definition
public class MyClass
{
    static void Main()
    {
    }
}
#endregion
```

Директива `#define`

- Директива **`#define`** позволяет определить символ, который при передаче его директиве **`#if`** как выражения приведет к получению значения **`true`**.
- Также можно определить символ с помощью параметра компилятора [`/define`](#). Для отмены определения символа служит

Директива `#if`

- При обнаружении компилятором C# директивы `#if`, за которой далее следует директива `#endif`, компиляция кода между двумя директивами выполняется только в том случае, если определен указанный символ.

Примеры использования

- `#define DEBUG`
- `// ...`
- `#if DEBUG`
- `Console.WriteLine("Debug
version");`
- `#endif`

Примеры использования

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

Атрибуты

- Атрибуты обеспечивают эффективный способ связывания метаданных или декларативной информации с кодом (сборками, типами, методами, свойствами и т. д.).

Сведения о атрибутах

- Атрибуты расширяют в программе сведения в метаданных.
- Один или несколько атрибутов могут применяться к сборкам, модулям или более мелким программным элементам, таким как классы и свойства.
- Атрибуты могут принимать аргументы.
- Программа может проверить собственные метаданные или метаданные в других программах с помощью *рефлексии*.

Использование атрибутов

- Атрибуты могут быть размещены в большинстве объявлений, хотя определенный атрибут может ограничить типы объявлений, в которых он допустим. В C# атрибут задается путем размещения его имени, заключенного в квадратные скобки ([]), перед объявлением сущности

Целевой объект атрибута

- *Целевым объектом атрибута является сущность, к которой относится атрибут.*
- *Например, атрибут может относиться к классу, отдельному методу или целой сборке. По умолчанию атрибут относится к тому элементу, перед которым он указан.*

Параметры атрибутов

- Многие атрибуты имеют параметры, которые могут быть **позиционными**, **неименованными** или **именованными**. Любые позиционные параметры следует указывать в определенном порядке, их нельзя опустить; именованные параметры являются необязательными и могут быть указаны в любой последовательности.

Пример кода

- `// default: applies to method`
- `[SomeAttr]`
- `int Method1() { return 0; }`
-
- `// applies to method`
- `[method: SomeAttr]`
- `int Method2() { return 0; }`
-
- `// applies to return value`
- `[return: SomeAttr]`
- `int Method3() { return 0; }`