**如何：使用 foreach 访问集合类（C# 编程指南）**

下面的代码示例演示如何编写可与 [foreach](https://msdn.microsoft.com/zh-cn/library/ttw7t8t6.aspx) 结合使用的非泛型集合类。该示例定义了字符串 tokenizer 类。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 此示例描述的是仅当您无法使用泛型集合类时才采用的推荐做法。有关如何实现支持 [IEnumerable<T>](https://msdn.microsoft.com/zh-cn/library/9eekhta0.aspx) 的类型安全的泛型集合类，请参见[迭代器（C# 和 Visual Basic）](https://msdn.microsoft.com/zh-cn/library/dscyy5s0.aspx)。 |

在该示例中，以下代码段使用 Tokens 类通过“ ”和“-”分隔符将句子“This is a sample sentence.”分成若干标记。该代码然后使用 **foreach** 语句显示这些标记。

Tokens f = new Tokens("This is a sample sentence.", new char[] {' ','-'});

// Display the tokens.

foreach (string item in f)

{

System.Console.WriteLine(item);

}

在内部，Tokens 类使用数组存储这些标记。因为数组可实现 [IEnumerator](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.aspx) 和 [IEnumerable](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.aspx)，所以代码示例使用了数组的枚举方法（[GetEnumerator](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.getenumerator.aspx)、[MoveNext](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.movenext.aspx)、[Reset](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.reset.aspx) 和 [Current](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.current.aspx)），而不是在 Tokens 类中定义这些方法。方法定义包括在该示例中，以明确如何定义它们以及每个定义的内容。

using System.Collections;

// Declare the Tokens class. The class implements the IEnumerable interface.

public class Tokens : IEnumerable

{

private string[] elements;

Tokens(string source, char[] delimiters)

{

// The constructor parses the string argument into tokens.

elements = source.Split(delimiters);

}

// The IEnumerable interface requires implementation of method GetEnumerator.

public IEnumerator GetEnumerator()

{

return new TokenEnumerator(this);

}

// Declare an inner class that implements the IEnumerator interface.

private class TokenEnumerator : IEnumerator

{

private int position = -1;

private Tokens t;

public TokenEnumerator(Tokens t)

{

this.t = t;

}

// The IEnumerator interface requires a MoveNext method.

public bool MoveNext()

{

if (position < t.elements.Length - 1)

{

position++;

return true;

}

else

{

return false;

}

}

// The IEnumerator interface requires a Reset method.

public void Reset()

{

position = -1;

}

// The IEnumerator interface requires a Current method.

public object Current

{

get

{

return t.elements[position];

}

}

}

// Test the Tokens class.

static void Main()

{

// Create a Tokens instance.

Tokens f = new Tokens("This is a sample sentence.", new char[] {' ','-'});

// Display the tokens.

foreach (string item in f)

{

System.Console.WriteLine(item);

}

}

}

/\* Output:

This

is

a

sample

sentence.

\*/

在 C# 中，集合类不必通过实现 [IEnumerable](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.aspx) 和 [IEnumerator](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.aspx) 来与 **foreach** 兼容。如果此类具有所需的 [GetEnumerator](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.getenumerator.aspx)、[MoveNext](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.movenext.aspx)、[Reset](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.reset.aspx) 和 [Current](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.current.aspx) 成员，则可与 **foreach** 结合使用。省略接口有一个好处：即，您可以比 [Object](https://msdn.microsoft.com/zh-cn/library/system.object.aspx) 更为具体地定义 Current 的返回类型。这会提供类型安全。

例如，可更改上述示例中的以下行。

// Change the Tokens class so that it no longer implements IEnumerable.

public class Tokens

{

// . . .

// Change the return type for the GetEnumerator method.

public TokenEnumerator GetEnumerator()

{ }

// Change TokenEnumerator so that it no longer implements IEnumerator.

public class TokenEnumerator

{

// . . .

// Change the return type of method Current to string.

public string Current

{ }

}

}

由于 Current 返回字符串，因此编译器能够检测何时在 **foreach** 语句中使用了不兼容的类型，如以下代码所示。

// Error: Cannot convert type string to int.

foreach (int item in f)

省略 [IEnumerable](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.aspx) 和 [IEnumerator](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerator.aspx) 的缺点是：集合类不再与其他公共语言运行时语言的 **foreach** 语句或等效语句交互。

**请参阅**

[System.Collections.Generic](https://msdn.microsoft.com/zh-cn/library/system.collections.generic.aspx)

[C# 参考](https://msdn.microsoft.com/zh-cn/library/618ayhy6.aspx)

[C# 编程指南](https://msdn.microsoft.com/zh-cn/library/67ef8sbd.aspx)

[数组（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/9b9dty7d.aspx)

[集合（C# 和 Visual Basic）](https://msdn.microsoft.com/zh-cn/library/ybcx56wz.aspx)