

第16期-深入理解迭代器

2022年5月19日 13:37

迭代器介绍

编写的几乎每个程序都需要循环访问集合。因此需要编写代码来检查集合中的每一项。

```
foreach (var item in collection)
{
    Console.WriteLine(item?.ToString());
}

await foreach (var item in asyncSequence)
{
    Console.WriteLine(item?.ToString());
}
```

为什么能够被迭代?

语法规则:

<https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/statements/iteration-statements?#the-foreach-statement>

foreach: [System.Collections.Generic.IEnumerable<T>](#)

await foreach: [System.Collections.Generic.IAsyncEnumerable<T>](#)

使用迭代器方法的枚举源

同步迭代器

```
public IEnumerable<int> GetSingleDigitNumbers()
{
    yield return 0;
    yield return 1;
    yield return 2;
    yield return 3;
}
```

```
public IEnumerable<int> GetSetsOfNumbers()
```

```

{
    int index = 0;
    while (index < 10)
        yield return index++;

    yield return 50;

    index = 100;
    while (index < 110)
        yield return index++;
}

```

异步迭代器

```

public async IEnumerable<int> GetSetsOfNumbersAsync()
{
    int index = 0;
    while (index < 10)
        yield return index++;

    await Task.Delay(500);

    yield return 50;

    await Task.Delay(500);

    index = 100;
    while (index < 110)
        yield return index++;
}

```

反编译 foreach 原理

```

{
    IL_0019: br.s      IL_002c
    IL_001b: ldloc.s   V_1
    IL_001d: call     instance !0 valuetype [System.Collections]System.Collections.Generic.List`1/Enumerator<string>::get_Current()
    IL_0022: stloc.2
    IL_0023: nop
    IL_0024: ldloc.2
    IL_0025: call     void [System.Console]System.Console::WriteLine(string)
    IL_002a: nop
    IL_002b: nop
    IL_002c: ldloc.s   V_1
    IL_002e: call     instance bool valuetype [System.Collections]System.Collections.Generic.List`1/Enumerator<string>::MoveNext()
    IL_0033: brtrue.s  IL_001b
    IL_0035: leave.s   IL_0046
} // end .try
finally
{
    IL_0037: ldloc.s   V_1
    IL_0039: constrained. valuetype [System.Collections]System.Collections.Generic.List`1/Enumerator<string>
    IL_003f: callvirt instance void [System.Runtime]System.IDisposable::Dispose()
    IL_0044: nop
    IL_0045: endfinally
} // end handler
IL_0046: ret

```

深入理解迭代器

同步迭代器

```
var enumerator = collection.GetEnumerator();

try
{
    while (enumerator.MoveNext())
    {
        var item = enumerator.Current;
        Console.WriteLine(item.ToString());
    }
}
finally
{
    (enumerator as IDisposable)?.Dispose();
}
```

异步迭代器

```
var enumerator = collection.GetAsyncEnumerator();

try
{
    while (await enumerator.MoveNextAsync())
    {
        var item = enumerator.Current;
        Console.WriteLine(item.ToString());
    }
}
finally
{
    if (enumerator is IAsyncDisposable asyncDisposable)
    {
        await asyncDisposable.DisposeAsync();
    }
}
```