2.迭代器模式 与 IEnumerable

什么是迭代器模式

在不知道集合内部细节的情况下，提供一个按序方法存取的一个对象集合体的每一个单元。—GoF

提供一种方法顺序访问一个集合对象中的各个元素，又不暴露该对象的内部表示

迭代器模式由于经常使用到，已经被 .Net 收录到 API 中。

在 C# 中，经常使用泛型存储对象，当想按序存取这些泛型容器时，都会使用 C# 的 foreach 语句。

foreach 语句就是一个能顺序访问一个集合的方法。

它就是 C# 语言内置的迭代器模式。

基本上，以上两个定义，足够我们理解了。具体的经典的 迭代器模式的 UML 图 实现太过抽象。而且本教程也不是专门讲解迭代器模式的教程，所以我们先定一个容易达到的目标:掌握 IEnumerable 和 IEnumerator 两个接口。

如何使用 IEnumerable

在使用之前，我们先理解一下 IEnumerable。这里老师直接给一个比较容易记住的定义，IEnumerable 中文意思是 可枚举的，教练直接把他理解为 IForeachable（可遍历的），虽然定义不是很准确，但是在区分 IEnumerable 和 IEnumerator 时候很容易搞清楚。

我们直接看下 IEnumerable 的定义:

```
//这个接口告知调方对象的子项可以枚举
```

```
public interface IEnumerable
{
    IEnumerator GetEnumerator();
}
```

示例代码如下:

```
/*************************************************************************
 * http://sikiedu.com liangxie
 *************************************************************************/

using System.Collections;
using UnityEngine;

namespace UniRxLesson
{
    public class IEnumerableExample : MonoBehaviour
    {
        class ForEachable : IEnumerable
        {
            object[] mObjArray = new object[4]
            {
                "1","2","3","4",
            };


            public IEnumerator GetEnumerator()
            {
                return mObjArray.GetEnumerator();
            }
        }

        private void Start()
        {
            var foreachAble = new ForEachable();

            foreach (var number in foreachAble)
            {
                Debug.Log(number);
```

```
                }
            }

        }
}
```

输出结果为:

```
1
2
3
4
```

非常容易理解。

只要实现一个 GetEnumerator 就可以使用 foreach 了。

今天的内容就这些。