10.ReactiveCommand

我们先来看下 ReactiveCommand 定义

```
public interface IReactiveCommand<T> : IObservable<T>
{
    IReadOnlyReactiveProperty<bool> CanExecute { get; }

    bool Execute(T parameter);
}
```

它提供了两个 API:

- CanExecte
- Execute

Execute 方法是被外部调用的。也就是这个 Command 的执行。这个很容易理解，只要外部调用的 Execute 就会执行。

而 CanExecute 则是内部使用的，并且对外部提供了只读访问。

当 CanExecute 为 false 时，在外部调用 Execute 则该 Command 不会被执行。

当 CanExecute 为 true 时，在外部调用 Execute 则该 Command 会被执行。

是什么决定 CanExecute 为 false 或 true 呢?

答案是其他的 Observable。

新创建的 ReactiveCommand 默认 CanExecute 为 true。

我们看下代码就好了。

```
/
**********************************************************************
********
```

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

**************************************************************************
********/

using System;
using System.Collections.Generic;
using UniRx;
using UnityEngine;

namespace UniRxLesson
{
    public class ReactiveCommandExample : MonoBehaviour
    {
        void Start()
        {
            ReactiveCommand command = new ReactiveCommand();

            command.Subscribe(_ =>
            {
                Debug.Log("command executed");
            });

            command.Execute();
            command.Execute();
            command.Execute();
        }
    }
}
```

输出结果为:

```
command executed
command executed
command executed
```

非常地简单，只要调用 Execute。command 就会通知 Subscribe 的回调(因为 CanExecute 为 true)。

CanExecute 的开启关闭是由 Observable （事件源）决定的。

示例代码如下:

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

**********************************************************************
********/

using System;
using System.Collections.Generic;
using UniRx;
using UnityEngine;

namespace UniRxLesson
{
    public class MouseUpExample : MonoBehaviour
    {
        void Start()
        {
            var leftMouseClickStream =
Observable.EveryUpdate().Where(_ =>
Input.GetMouseButtonDown(0)).Select(_ => true);
            var rightMouseClickStream =
Observable.EveryUpdate().Where(_ =>
Input.GetMouseButtonUp(0)).Select(_ => false);


            var mouseUp = Observable.Merge(leftMouseClickStream,
rightMouseClickStream);

            var reactiveCommand = new
ReactiveCommand(mouseUp,false);

            reactiveCommand.Subscribe(x =>
                    {
```

```
                        Debug.Log(x);
                    });

            Observable.EveryUpdate()
                    .Subscribe(_ =>
                    {
                        reactiveCommand.Execute();
                    });
        }
    }
}
```

当按下鼠标时持续输出 "()"，当抬起鼠标时，则停止输出。

非常容易理解。

当然 ReactiveCommand 也是可以被订阅(Subscribe) 的，在订阅之前呢，也可以使用 Where 等操作符进行事件操作。

示例代码如下:

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

**************************************************************************
********/

using System;
using System.Collections.Generic;
using UniRx;
using UnityEngine;

namespace UniRxLesson
{
    public class OperatorExample : MonoBehaviour
    {
        void Start()
        {
```

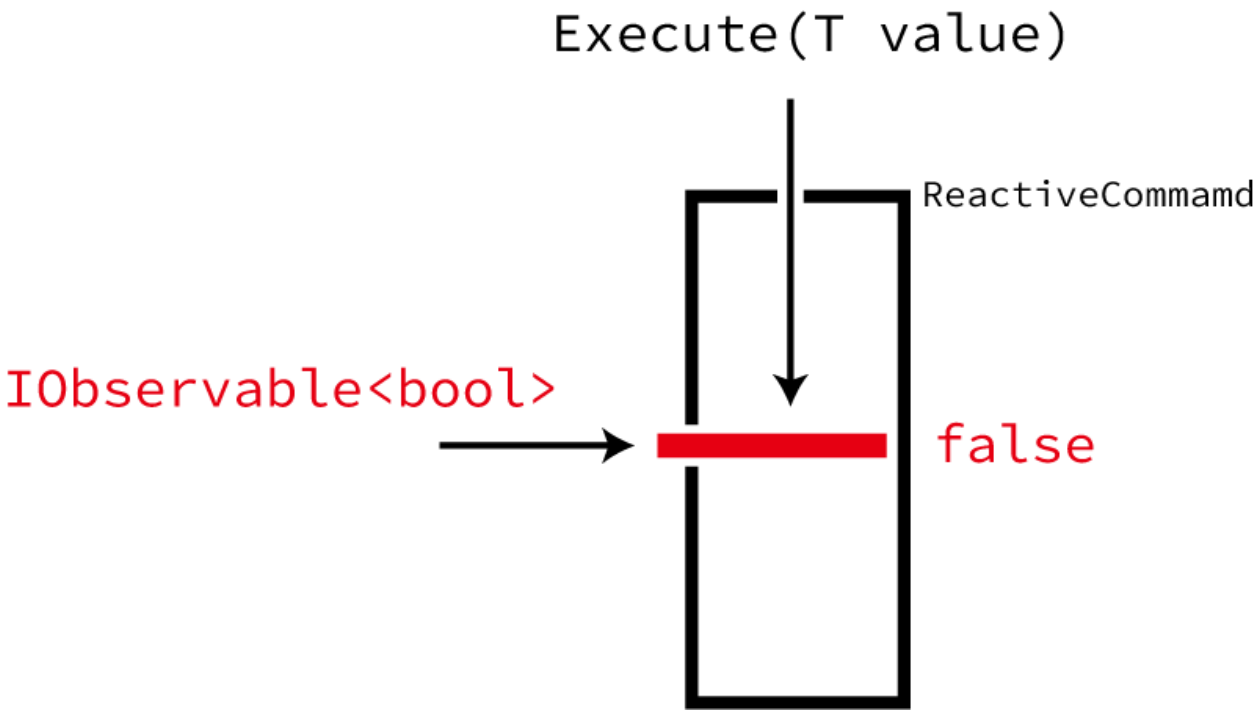
```
            var reactiveCommand = new ReactiveCommand<int>();

            reactiveCommand.Where(x => (x % 2 == 0)).Subscribe(x =>
Debug.LogFormat("{0} is Even numbers .", x));
            reactiveCommand.Where(x => (x % 2 != 0))
                .Timestamp()
                            .Subscribe(x => Debug.LogFormat("{0} is
Odd,{1}", x.Value, x.Timestamp));

            reactiveCommand.Execute(2);//输出“2 is Even numbers.
            reactiveCommand.Execute(3);//输出 3 is Odd,2012/3/4
20:38:51 +08:00
        }
    }
}
```

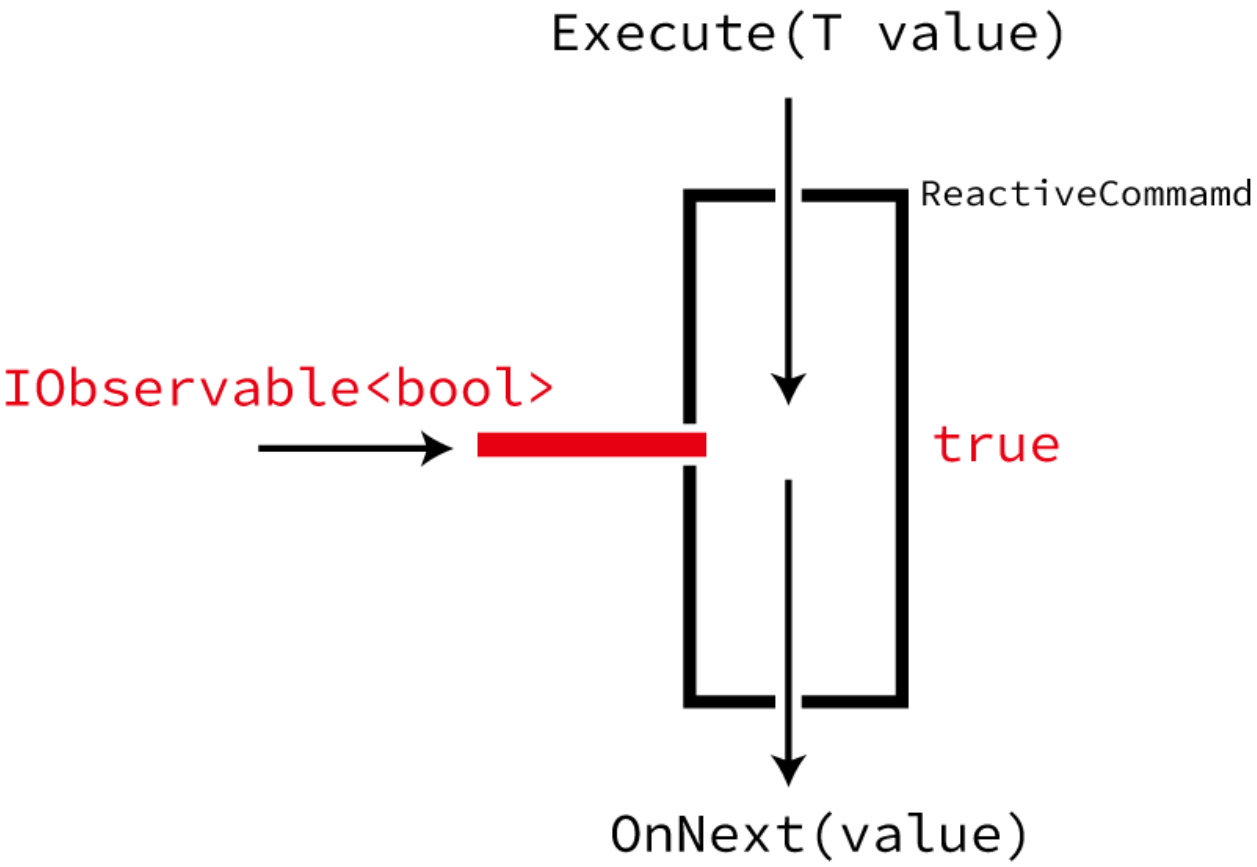
输出结果为

```
2 is Even numbers .
3 is Odd,09/25/2018 05:46:50 +00:00
```

到此，ReactiveCommand 的基本用法，大家应该掌握了。

我们通过以下两图，简单去理解下 ReactiveCommand 执行原理。


Execute(T value)
ReactiveCommamd
IObservable<bool>
false



ReactiveCommand 除了能做以上一些简单的事情外，其实可以做非常多强大的功能。

但是要介绍非常强大的功能之前呢，我们要先学好 UniRx 的入门基础及一点点原理。

所以今天呢，只是对 ReactiveCommand 进行了一个简介，在之后呢会对 ReactiveCommand 进行一个深入地了解。

今天的内容就这些。