IObservable<T>和IObserver<T>支持观察者模式的接口。可以通过实际的例子来学习Observer模式。

Observer模式：重用，解耦合。又称发布-订阅模式。订阅的网络杂志、话费清单的实现类似于Observer模式。

简单的Observer模式的例子：

参考：<http://blog.chinaunix.net/uid-576762-id-2733752.html>

例：程序observer\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// Observer模式

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace observer\_test1

{

// 定义事件参数

internal class CheckListEventArgs:EventArgs

{

private readonly string check\_list;

public CheckListEventArgs(string check\_list)

{

this.check\_list = check\_list;

}

public string CheckList

{

get { return check\_list; }

}

}

internal delegate void PublishEventHandler(object sender, CheckListEventArgs args);

// 发布者基类

internal class Subject

{

public event PublishEventHandler PublishCheckListEvent;

// 通知所有订阅者

public void Notify(CheckListEventArgs args)

{

if (PublishCheckListEvent != null)

PublishCheckListEvent(this, args);

}

}

// 订阅者基类

internal abstract class Observer

{

public abstract void Received(object sender, CheckListEventArgs args);

}

// 发布者具体类

class Moblie : Subject

{

public void SimulateCheckList()

{

Console.WriteLine("--------话费清单-------");

CheckListEventArgs args = new CheckListEventArgs("市话话费12\t\t长途话费20\n");

base.Notify(args);

}

}

// 订阅者具体类

class Jerry:Observer

{

public override void Received(object sender, CheckListEventArgs args)

{

Console.WriteLine("Jerry收到话费通知:\n{0}", args.CheckList);

}

}

// 另一个订阅者

class Anco:Observer

{

public void Subscribe(Subject obj)

{

obj.PublishCheckListEvent += Received;

}

public void Unsubscribe(Subject obj)

{

obj.PublishCheckListEvent -= Received;

}

public override void Received(object sender, CheckListEventArgs args)

{

Console.WriteLine("Anco收到话费通知:\n{0}", args.CheckList);

}

}

class ObserverTest

{

static void Main(string[] args)

{

Moblie mobile = new Moblie();

Jerry jerry = new Jerry();

Anco anco = new Anco();

// 订阅

mobile.PublishCheckListEvent += jerry.Received;

// 订阅

anco.Subscribe(mobile);

// 从结果可以看出，同一个对象

// 同一个事件多次注册了同一个处理函数

// 当事件触发时，也会执行多次

anco.Subscribe(mobile);

mobile.SimulateCheckList();

}

}

}

输出结果：

--------话费清单-------

Jerry收到话费通知:

市话话费12 长途话费20

Anco收到话费通知:

市话话费12 长途话费20

Anco收到话费通知:

市话话费12 长途话费20

定义对象间的一种一对多的依赖关系,当一个对象的状态发生改变时, 所有依赖于它的对象都得到通知并被自动更新。

Observer模式描述了如何建立这种关系。这一模式中的关键对象是目标(Subject)和观察者(Observer)。一个目标可以有任意数目的依赖它的观察者。一旦目标的状态发生改变, 所有的观察者都得到通知。作为对这个通知的响应，每个观察者都将查询目标以使其状态与目标的状态同步。