C#中的事件和委托起源于C++中的函数指针。

参考：<http://blog.chinaunix.net/uid-576762-id-2733751.html>

委托与C++函数指针一个比较重要的区别是委托支持多播委托，通过声明一个委托，来调用多个函数。

委托是一个特殊的类型，有几个自身的成员方法：BeginInvoke，EndInvoke，Invoke，是定义委托时，编译器自动生成的，而且都是virtual函数，在运行时实现。

C#中的事件是特殊的委托。

定义一个事件时，编译器会自动加上“订阅”和“取消订阅”的两个方法。

例：程序event\_test6

// Copyright 2016.刘珅珅

// author：刘珅珅

// 事件与委托

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace event\_test6

{

// 定义事件引发时，传递的参数

class NewMailEventArgs : *EventArgs*

{

private readonly string from;

private readonly string to;

private readonly string subject;

public NewMailEventArgs(string from, string to, string subject)

{

this.from = from;

this.to = to;

this.subject = subject;

}

public string From

{

get { return from; }

}

public string To

{

get { return to; }

}

public string Subject

{

get { return subject; }

}

}

// 事件所用的委托

delegate void NewMailEventHandler(object sender, NewMailEventArgs e);

// 提供事件的类

// 在Observer模式中，相当于主题类

class MailManager

{

public event NewMailEventHandler NewMail;

// 通知已订阅事件的对象

protected virtual void OnNewMail(NewMailEventArgs e)

{

if (NewMail != null)

NewMail(this, e);

}

// 引发事件

public void SimulateNewMail(string from, string to, string subject)

{

NewMailEventArgs e = new NewMailEventArgs(from, to, subject);

OnNewMail(e);

}

}

// 使用事件的类

// 在Observer模式中相当于观察者类

class Fax

{

public Fax(MailManager m)

{

m.NewMail += FaxNewMail;

}

private void FaxNewMail(object sender, NewMailEventArgs e)

{

*Console*.*WriteLine*("Message arrived at Fax...");

*Console*.*WriteLine*("From = {0}, To = {1}, Subject = '{2}' ", e.From, e.To, e.Subject);

}

}

// 另一个使用事件的类

class Print

{

public Print(MailManager m)

{

m.NewMail += PrintNewMail;

}

private void PrintNewMail(object sender, NewMailEventArgs e)

{

*Console*.*WriteLine*("Message arrived at Print...");

*Console*.*WriteLine*("From = {0}, To = {1}, Subject = {2}", e.From, e.To, e.Subject);

}

}

class EventTest

{

static void Main(string[] args)

{

MailManager m = new MailManager();

Fax fax = new Fax(m);

Print prt = new Print(m);

// 触发事件

m.SimulateNewMail("Anco", "Jerry", "Event Test");

}

}

}

输出结果：

Message arrived at Fax...

From = Anco, To = Jerry, Subject = 'Event Test'

Message arrived at Print...

From = Anco, To = Jerry, Subject = Event Test