如何定义一个事件：使用event关键字



定义了事件成员的类型能提供以下功能：

1.方法能登记事件对它的关注（注册事件）

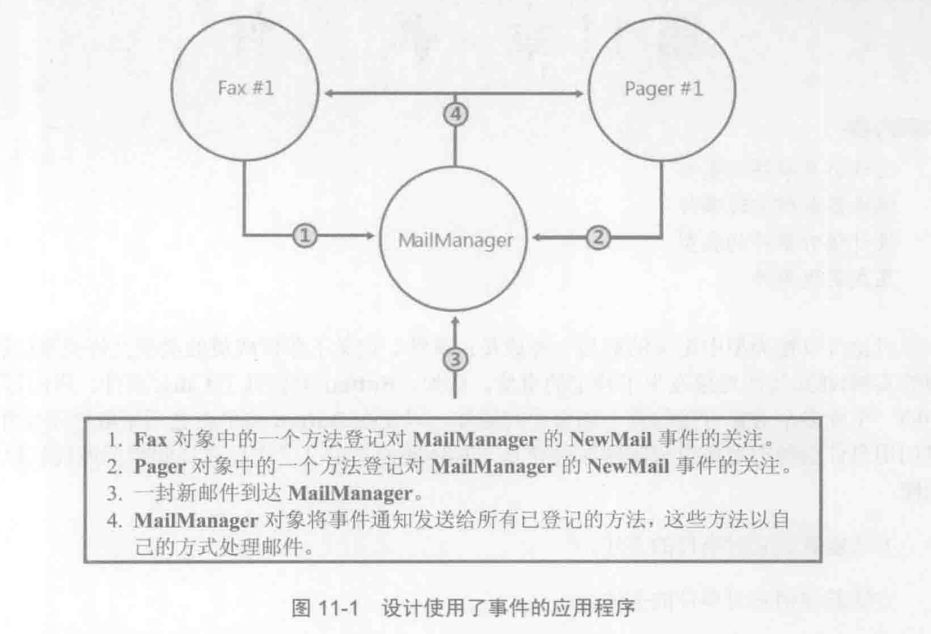
2.方法能注销事情对它的关注（注销事件）

3.事件发生时，登记了的方法将收到通知（回调事件）

Ps：CLR事件模型以委托为基础，因为委托可以以类型安全的方式调用回调方法。

下面是书上给出的例子：

当邮件到达时（到达MailManager），可以选择将它转发到传真机（Fax）或者Pager(寻呼机)



图中的四点完成了一个使用事件的应用程序，下面开始完成这四点

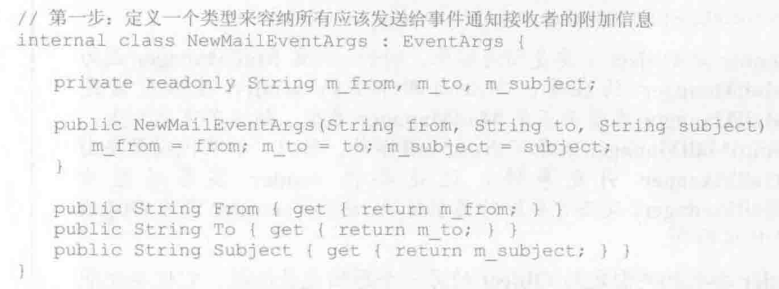
设计要公开事件的类型：

（在本例中即如何设计MailManager类型）

1.定义类型来容纳所有需要发送给事件通知接收者的附加信息

这个类用于装附加信息。附加信息在事件处理中起着非常重要的作用。当一个事件被触发时，通常需要传递一些关于事件的具体信息给事件处理器。这些信息可以帮助事件处理器更好地理解和处理事件（比如本例中的发送人from、邮件的接收者to以及邮件的主题subject）

在这个例子中：



定义一个类继承了EventArgs类，并且类名以EventArgs为结尾，里面定义了一些私有字段和属性以及一些用于公开这些私有字段的只读共有属性。

Ps：关于EventArgs类的定义和实现：

报纸上的文字

描述已自动生成

2.定义事件成员

背景图案

描述已自动生成

本例中这一步定义MailManager对象，然后在这个类里定义一个事件成员NewMail，EventHandler<NewMailEventArgs>是事件成员的类型，因为事件的调用和委托相联系，EventHandler<NewMailEventArgs>是一个委托类型，这个委托类型的定义如下：



EventHandler是微软定义的一个预定义委托，它表示用于处理不具有事件数据的事件的方法1。如果事件确实生成了数据，则必须使用泛型EventHandler<TEventArgs>委托类。

3.定义负责引发事件的方法来通知事件的登记对象

类定义一个受保护的虚方法，引发事件时类以及其派生类中的代码会调用该方法。在事件处理的上下文中，这个负责引发事件的方法的关键字为protect virtual一个主要原因是为了提供更大的灵活性。例如，派生类可能需要在事件被引发之前或之后添加一些额外的逻辑，或者可能需要完全改变事件被引发的方式。通过将引发事件的方法定义为虚方法，我们可以允许派生类根据需要重写这个方法2

文本, 信件

描述已自动生成

Ps：如何以线程安全的方式调用事件（）

报纸上的文字

描述已自动生成

4.定义方法将输入转化为期望事件

本例中在NewMail类中定义了一个方法用于接收输入通知对象事件已经发生

文本

低可信度描述已自动生成

下面是一个完整的例子：

文本

描述已自动生成

定义了一个Car类，它有一个Run事件成员，用于通知所有已登记的方法。还定义了一个CarEventArgs类，它从EventArgs派生，并用于封装附加信息。最后定义了一个RunCar方法，它接受司机和乘客作为参数，并创建一个包含附加信息的CarEventArgs对象。然后调用前面定义的负责引发事件的方法Run来引发事件。

编译器是如何实现事件的：

在上面的例子中，使用了EventHandler<TEventArgs>这个System命名空间下的委托，在编译时，编译器把定义事件成员的那行代码编译成了下面这个样子：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

分别为private可见性EventHandler<EventExample.CarEventArgs>类型的名为Run的字段，一个add\_Run方法和一个remove\_Run方法

Run引用不为null代表此时有1个listener监听该事件，因此引用了EventHandler<class EventExample.CarEventArgs>。在类中Run事件的可见性为public，但在ildasm中即使代码中的可见性为public也会被定义为private，这么做的原因是防止类外部的代码不正确地操作它。

add\_Run和remove\_Run分别调用Combine和Remove函数，代表注册和注销对事件的监听，这两个方法的可见性随事件定义的可见性变化

设计监听事件的类型

在上面Driver和passenger那个例子上加入一个监听名为Run事件的类型

文本

描述已自动生成

Ps：只能使用+=和-=去监听和注销事件，使用add和remove这样显式监听和注销会抛出异常

显式实现事件

暂时没看懂，没理解什么意思