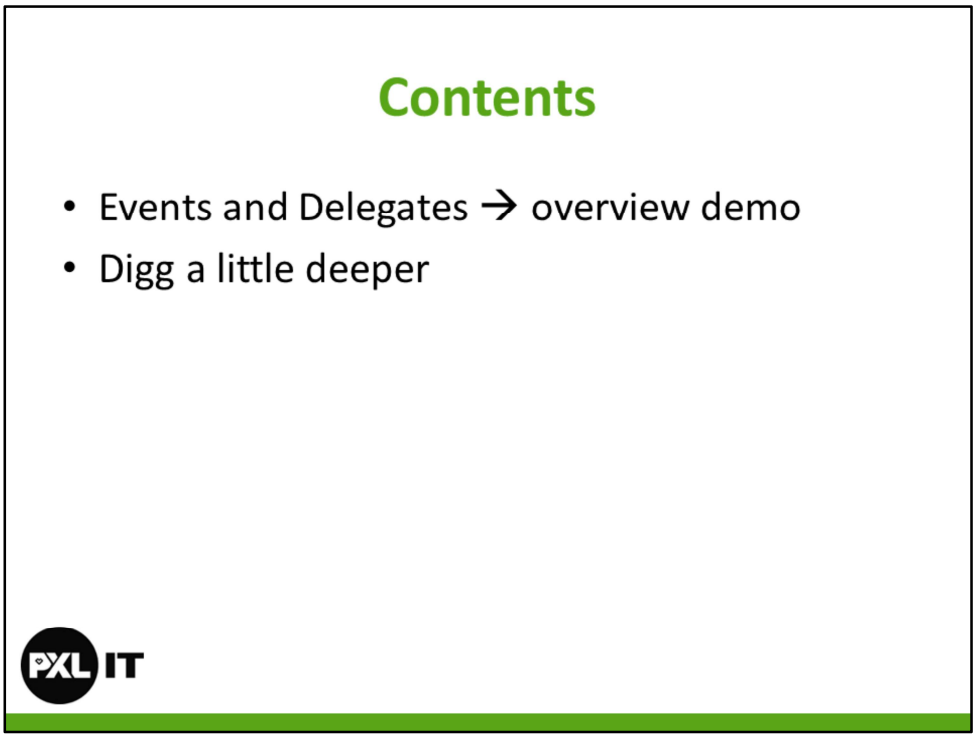
Events and delegates

.NET

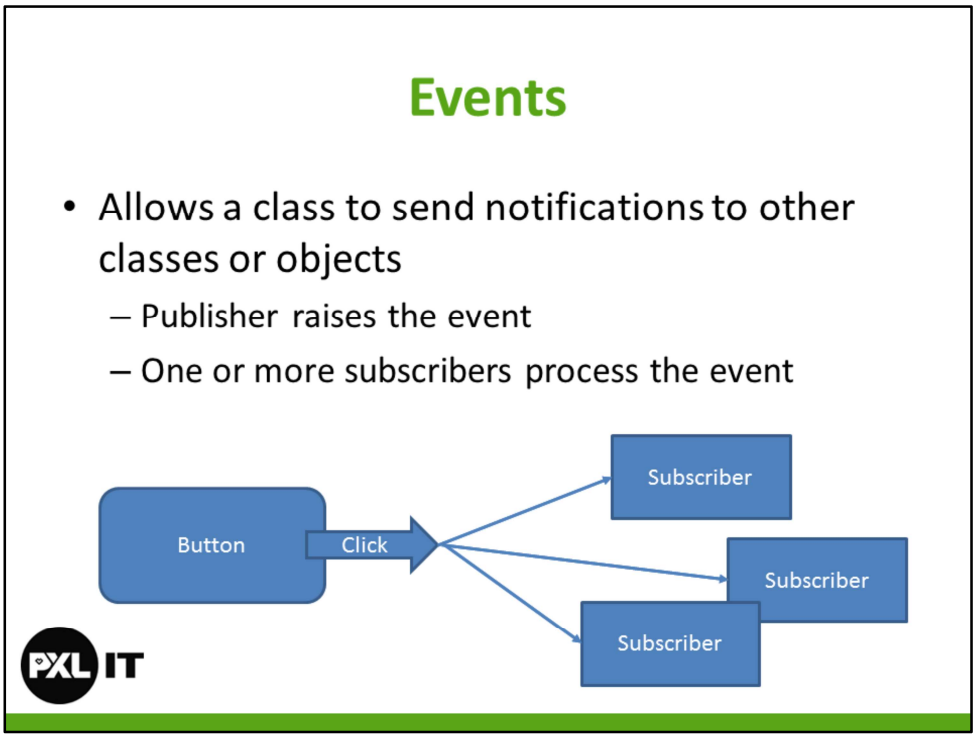
DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



This module focuses on the events and delegates and the mechanism behind it. You are already familiar with events (e.g. the Button click event). Behind the scenes delegates come into play to create what is called a PubSub pattern.

Knowing the internals about delegates and events allows you to write more complex software in a clearly manner.



This pattern is known as:

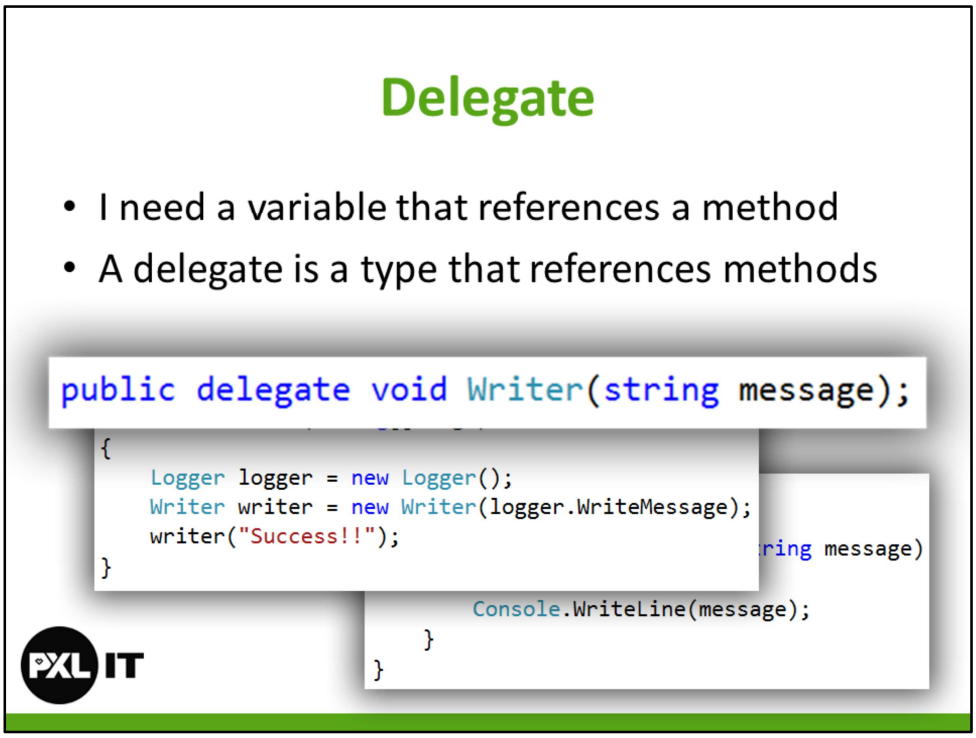
- PubSub
- Observer

This patterns is so important → C# adds specialized language constructs and execution mechanisms.

The pattern promotes separation of concerns and isolation:

- The Button (Publisher) does not need to know who receives its events
- There can me multiple listeners (Subscribers) who don't know each other and can do different things

Events are implemented in C# by a special construct called a **delegate**.



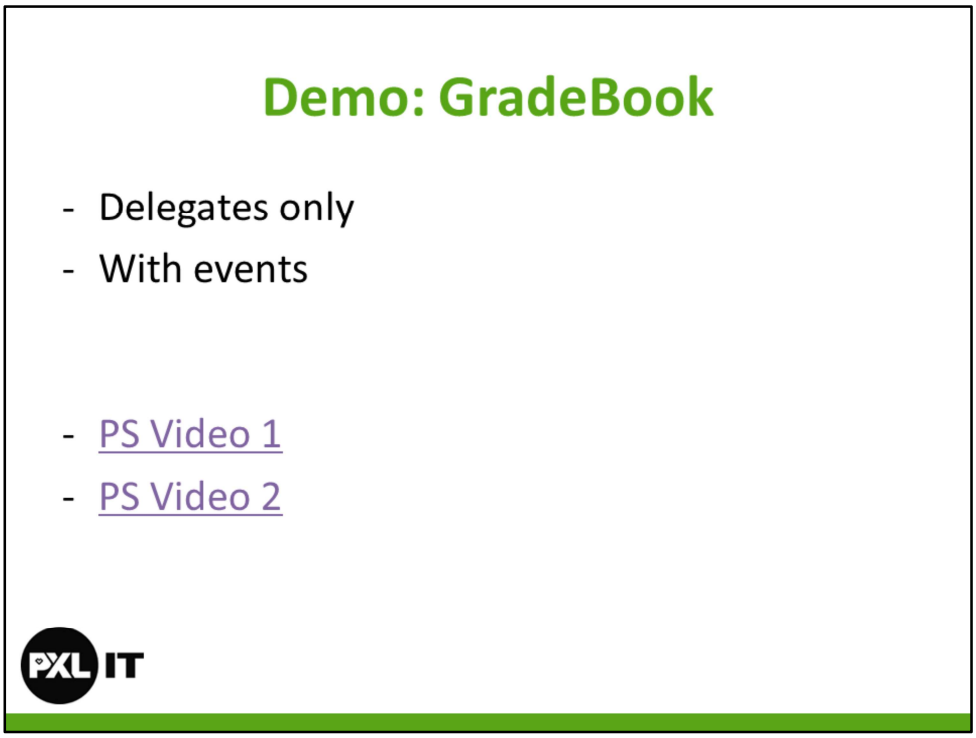
Demo: LoggerWriter

The variable does not hold a value in the traditional way, but holds a reference to a method. To do this in a typesafe way, you declare a delegate, which is a class, to define what kind of methods the variable can hold (parameters, return values).

Then you instantiate the delegate by invoking the constructor (new) and passing along a reference to a method.

Invoking the delegate [returnval] = delegatevar([params])

This executes the method.



Only delegates

You could implement it using only delegates, but the =-operator could overwrite existing methods.

Therefore you need events → only += and -= and certain conventions

Notice the Name property had NO IDEA which method(s) will be executed upon change!

The two video's give a good overview of the demo's.

Other Viewpoint

PXL IT

Event Raiser: girl provides notification, a message that goes out to one or more subscribers

EventArgs: extra info that comes along the event to the subscribers

Delegate: the channel through which the events pass

Event Handler: the subscriber(s)

What is an event?

- Events are notifications
- Play a central role in the .NET framework
- Provide a way to trigger notifications from end users or from objects

PXL IT

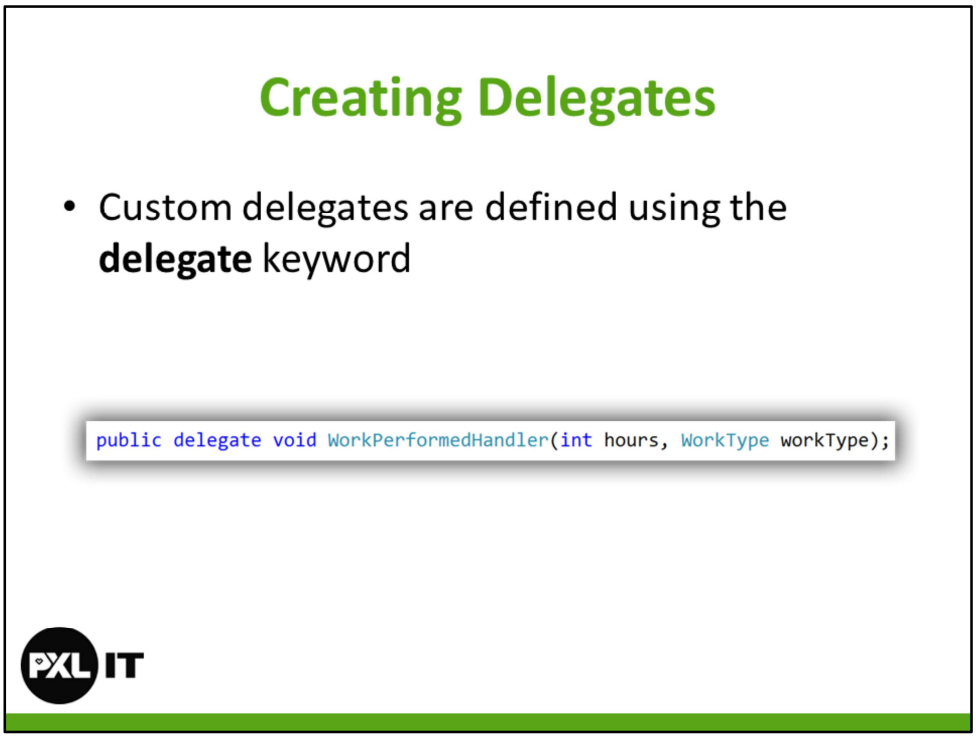
What is a Delegate?

- A delegate class is a specialized class often called a “Function Pointer”
- The glue between an event and an event handler
- Based on a MulticastDelegate base class
- A pipeline

PXL IT

What is an Event Handler

- Event handler is responsible for receiving and processing data from a delegate
- Normally receives two parameters
  - Sender
  - EventArgs
- EventArgs responsible for encapsulating event data



Look at the demo: WorkerDemo

This delegate defines the method signature for the handlers.

Delegate and Handler Method Parameters

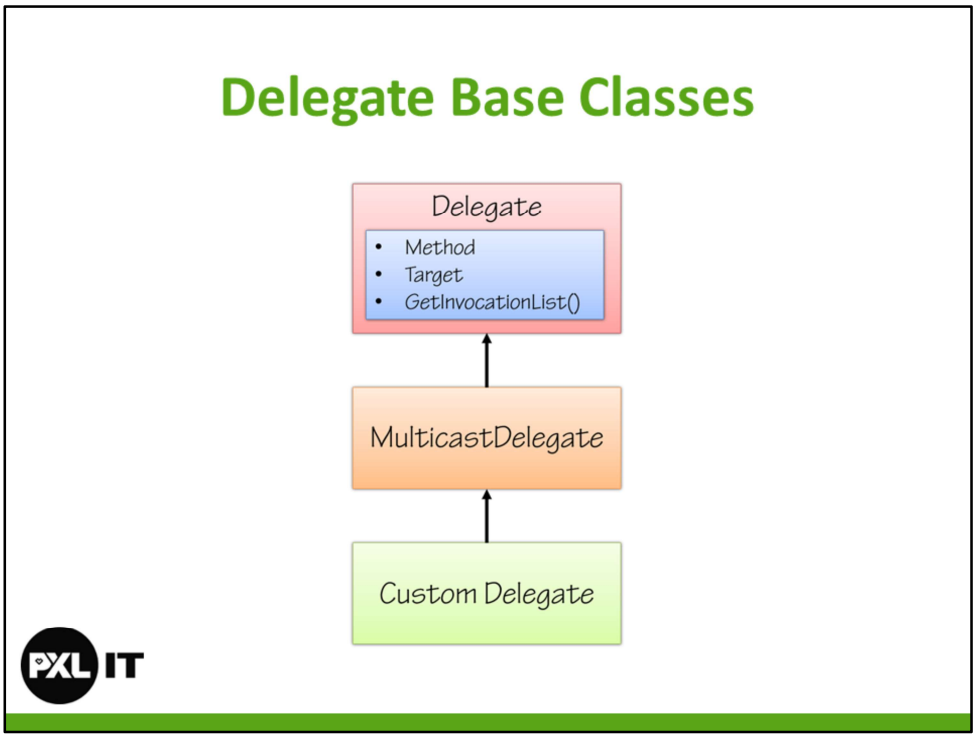
- The delegate signature must be mimicked by a handler method:

```
public delegate void WorkPerformedHandler(int hours, WorkType workType);
```

```
public void Manager_WorkPerformed(int workHours,
                                  WorkType wType)
{
    //...
}
```

Manager_WorkPerformed → parameter int and WorkType, returns void

Adheres to WorkPerformedHandler delegate



Delegate: abstract base class

https://msdn.microsoft.com/en-us/library/system.delegate(v=vs.110).aspx

Method: Gets the method represented by the delegate.

Target: Gets the class instance on which the current delegate invokes the instance method.

GetInvocationList: usefull for multicast

MultiCastDelegate: Represents a multicast delegate; that is, a delegate that can have more than one element in its invocation list.

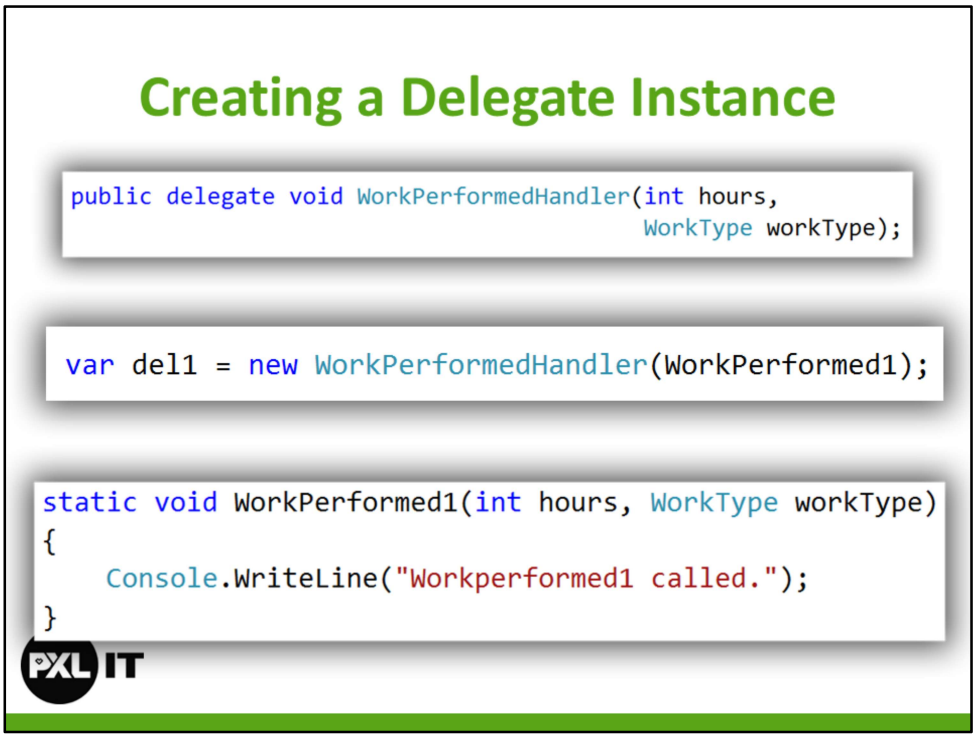
https://msdn.microsoft.com/en-us/library/system.multicastdelegate(v=vs.110).aspx

Custom Delegate: user defined

NOTE: you don't explicitly use the inheritance mechanism, but you must use de delegate keyword.

Multicast Delegate

- Can reference one or more delegate functions
- Tracks delegate references using an invocation list
- Delegates in the list are invoked sequentially

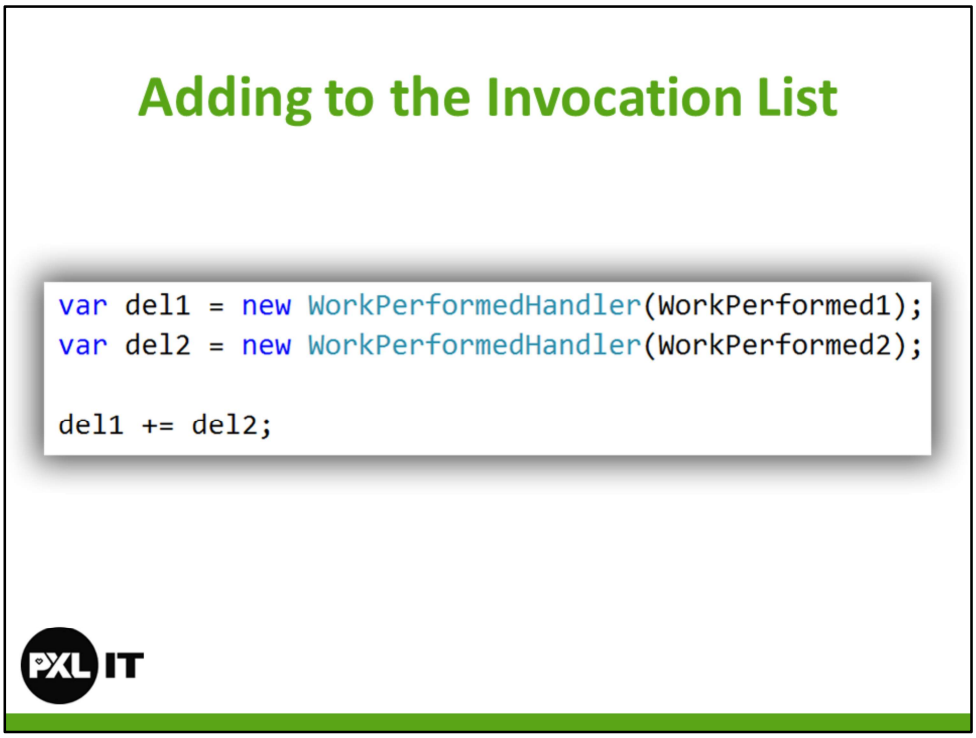


del1 is the “delegate instance”

It is really an object instantiation of the class WorkPerformedHandler

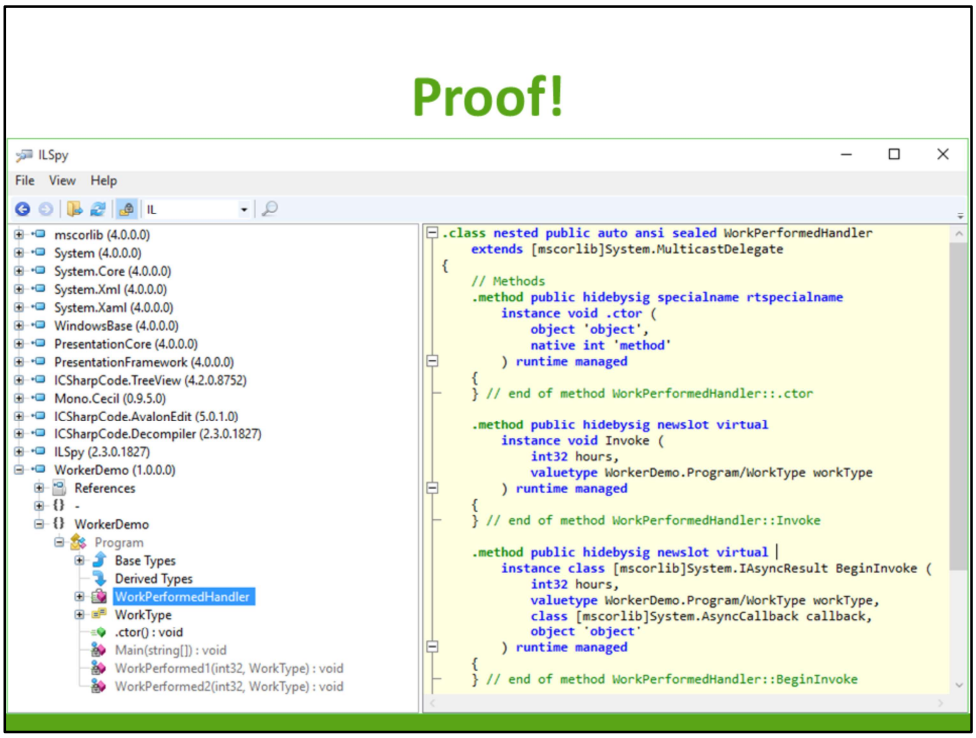
Invoking a Delegate instance

```
del1(5, WorkType.Golf);
```



What is the output of the call del1(5, WorkType.GoToMeetings) ?

Proof!



By opening up ILSpy, you can “proof” that a delegate is a class that inherits from MulticastDelegate

Return a value from a delegate

- a delegate returns a type
- Which value is returned from the Invocation List?
- → the last one

Proof this in your labs

Define an event

- Events can be defined by the **event** keyword

```
public delegate void WorkPerformedHandler(int hours,
                                           WorkType workType);

public event WorkPerformedHandler WorkPerformed;
```

Raising Events

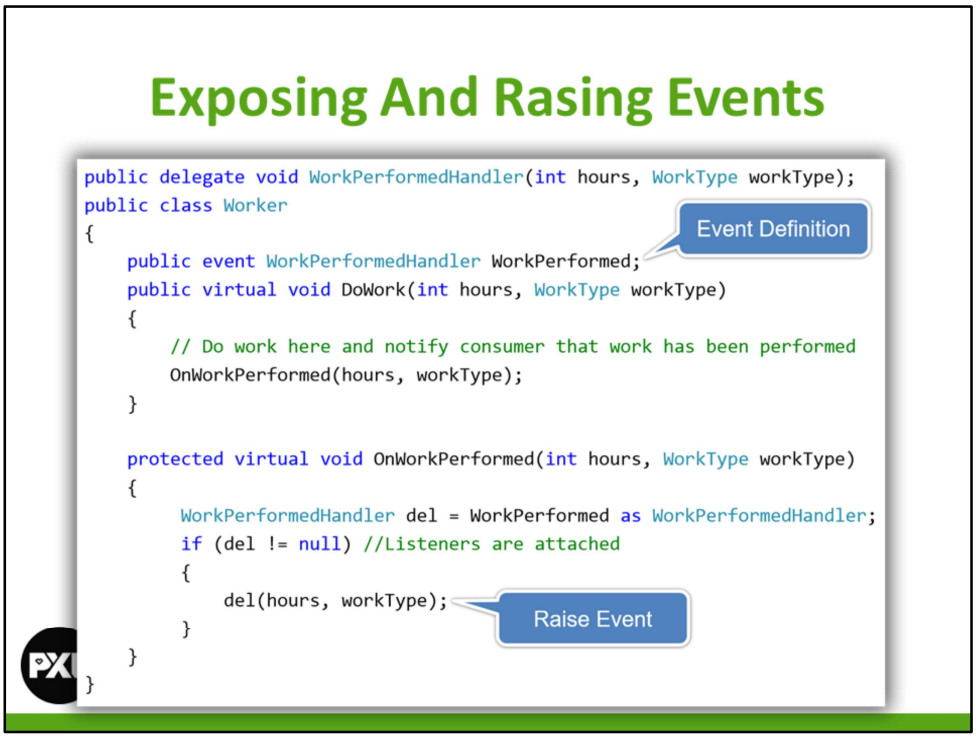
- Events are raised by calling the event like a method:

```
if (WorkPerformed != null)
{
    WorkPerformed(8, WorkType.GenerateReports);
}
```

Raising Events

- Or by accessing the event's delegate and invoking it directly:

```
WorkPerformedHandler del = WorkPerformed as WorkPerformedHandler;
if (del != null)
{
    del(8, WorkType.GenerateReports);
}
```



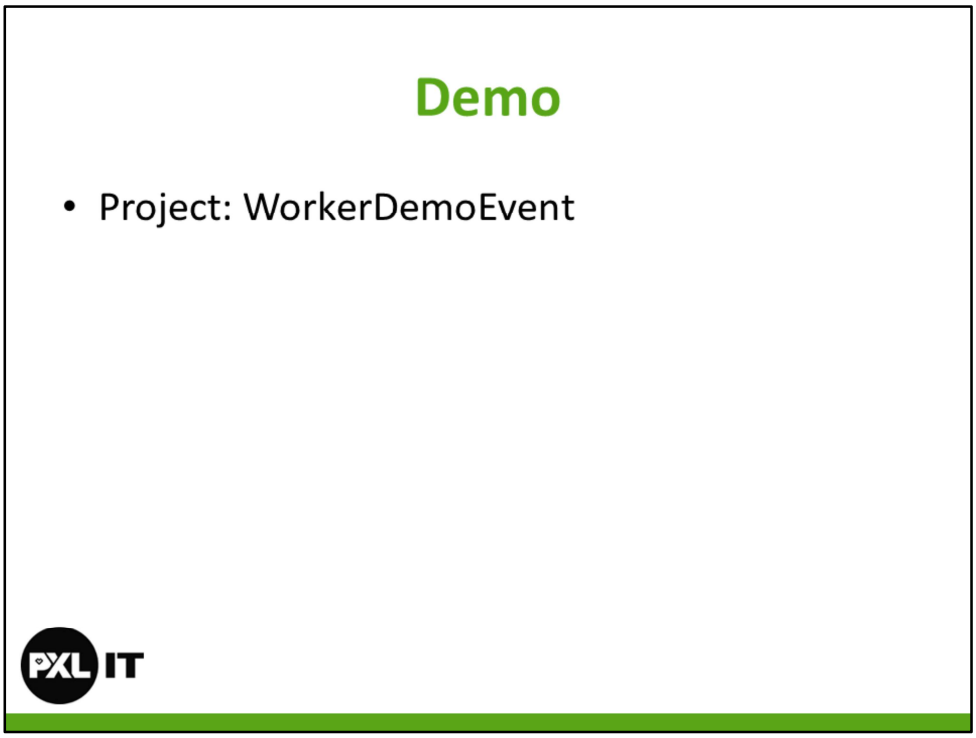
Best Practices / Conventions

You could raise the event inside DoWork directly: WorkPerformed(hours, workType)

Instead:

OnEventName → **On**WorkPerformed

Define this as a protected method, so you could override it.



Two event types:

- WorkPerformedHandler → based on custom delegate
- EventHandler → built in .NET delegate, the same you use with e.g. a button click.