

# Handling Events

Dan Wahlin

Twitter: @DanWahlin


Blog: <http://weblogs.asp.net/dwahlin>



Instantiating  
Delegates and  
Handling Events

Delegate  
Inference

Using  
Anonymous  
Methods



Event Handler


EventArgs

Event Raiser

Delegate

# Instantiating Delegates and Handling Events






Event Handler


Delegate

# Delegate and Handler Method Parameters

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



```
public delegate void WorkPerformedHandler(object sender,  
    WorkPerformedEventArgs e);
```



```
public void Manager_WorkPerformed(object sender,  
    WorkPerformedEventArgs e) {
```



...

}



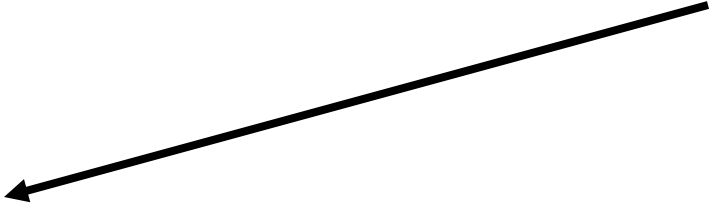
# Defining and Attaching Event Handlers

The += operator is used to attach an event to an event handler:

```
var worker = new Worker();  
worker.WorkPerformed +=  
    new EventHandler<WorkPerformedEventArgs>(worker_WorkPerformed);
```



Attach to Event  
through Delegate



```
void worker_WorkPerformed(object sender, WorkPerformedEventArgs e)  
{  
    Console.WriteLine(e.Hours.ToString());  
}
```

# Delegate Inference






# Delegate Inference

The += operator is used to attach an event to an event handler:

```
var worker = new Worker();  
worker.WorkPerformed += worker_WorkPerformed;
```



Compiler will “infer”  
the delegate

```
void worker_WorkPerformed(object sender, WorkPerformedHandler e)  
{  
    Console.WriteLine(e.);  
}
```


# Using Anonymous Methods



# Attaching Event Handlers Directly to Events

What if you want to attach an event handler directly to an event?

```
var worker = new Worker();  
worker.WorkPerformed += //Attach event handler here
```



```
void worker_WorkPerformed(object sender, WorkPerformedEventArgs e)  
{  
    Console.WriteLine(e.Hours.ToString());  
}
```

# Anonymous Methods

- Anonymous methods allow event handler code to be hooked directly to an event
- Anonymous methods are defined using the delegate keyword:

```
var worker = new Worker();  
worker.WorkPerformed += delegate(object sender,  
                                WorkPerformedEventArgs e)  
{  
    Console.WriteLine(e.Hours.ToString());  
}; //End of anonymous method
```

# Summary

- To listen to an event you must construct a delegate and add it to the invocation list
- The C# compiler provides “delegate inference” functionality
- Event handler methods can be attached directly to events