Creating Delegates, Events and EventArgs

Dan Wahlin

Twitter: @DanWahlin

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

Creating a Delegate

Defining an Event

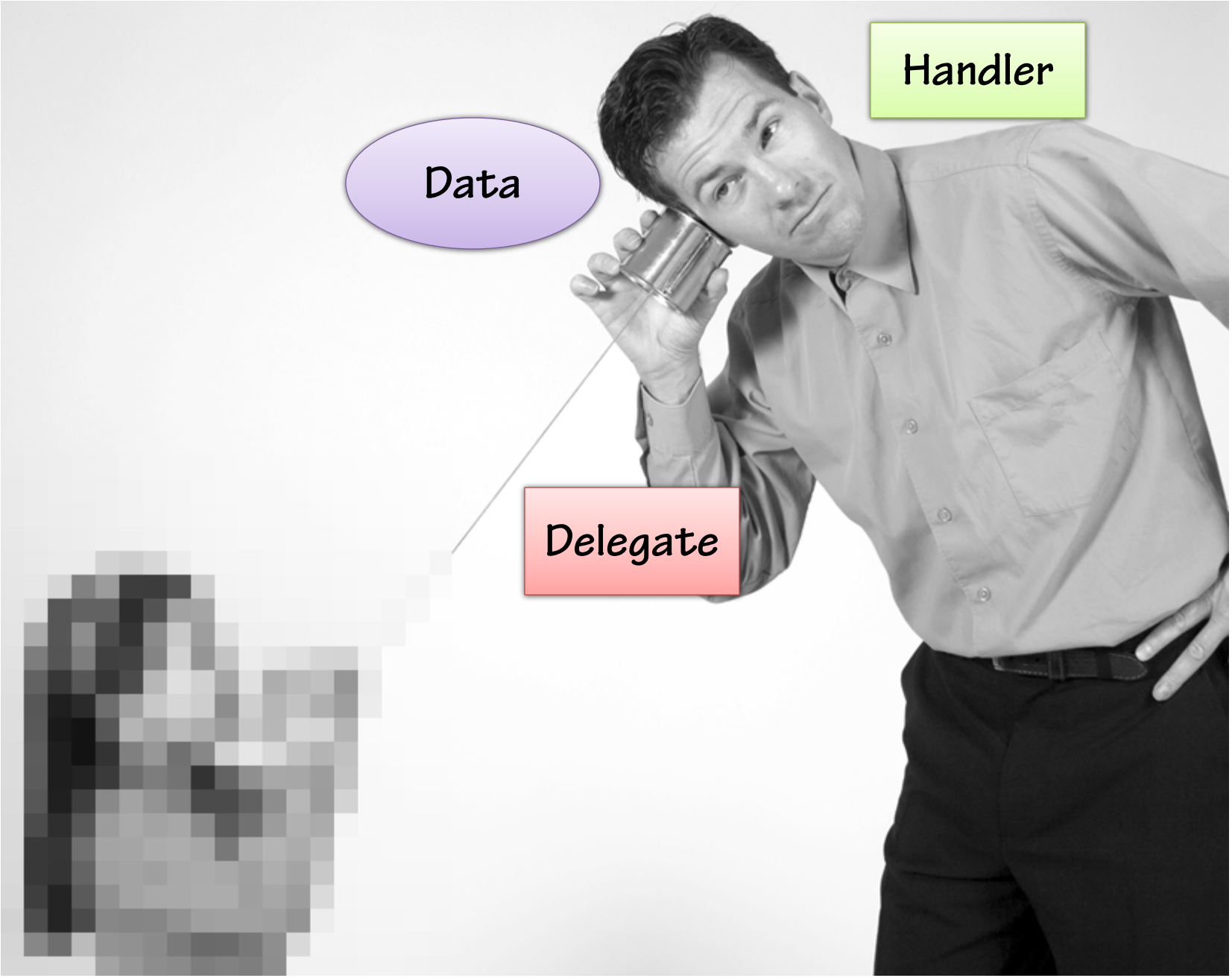
Raising Events

Creating an EventArgs Class


Event Handler
EventArgs
Event Raiser
Delegate

Creating a Delegate

pluralsight
hardcore developer training


Handler
Data
Delegate

Creating Delegates

- Custom delegates are defined using the **delegate** keyword:

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

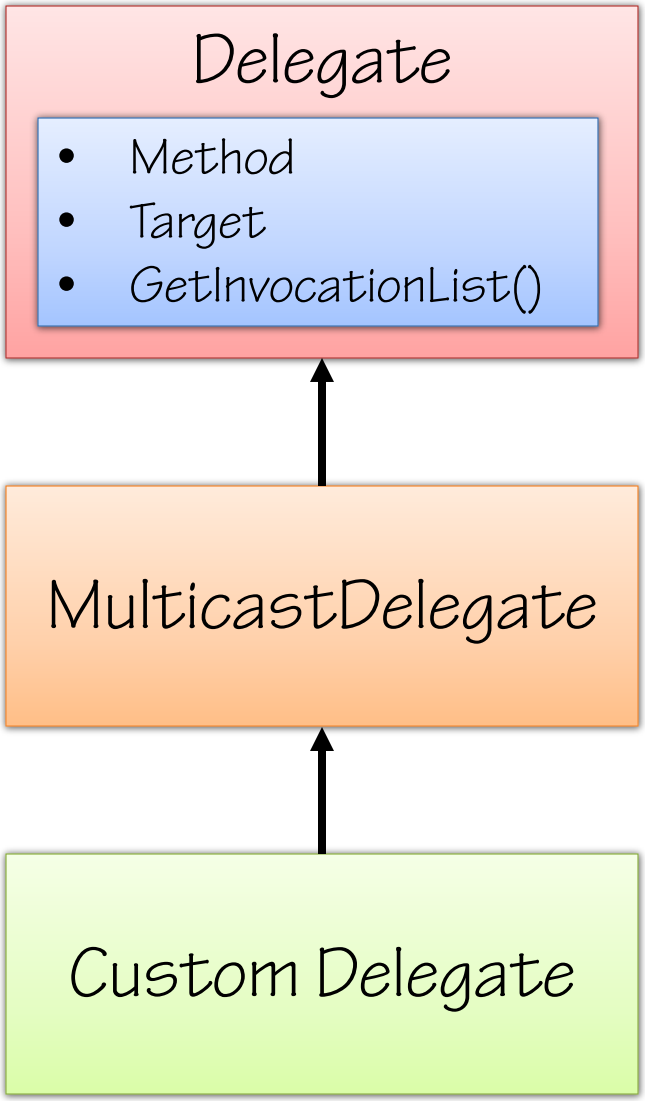
Delegate and Handler Method Parameters

- 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) {

    ...

}
```

Delegate Base Classes



What is a Multicast Delegate?

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

Creating a Delegate Instance

Delegate

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

Delegate Instance

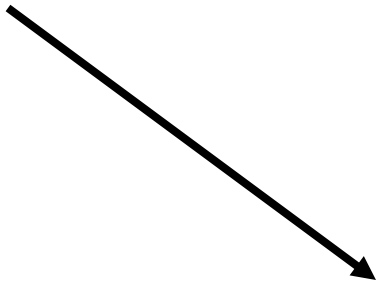
```
WorkPerformedHandler del1 =
  new WorkPerformedHandler(WorkPerformed1);
```

Handler

```
static void WorkPerformed1(int hours, WorkType workType)
{
    Console.WriteLine("WorkPerformed1 called");
}
```

Invoking a Delegate Instance

```
WorkPerformedHandler del1 =
  new WorkPerformedHandler(WorkPerformed1);

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

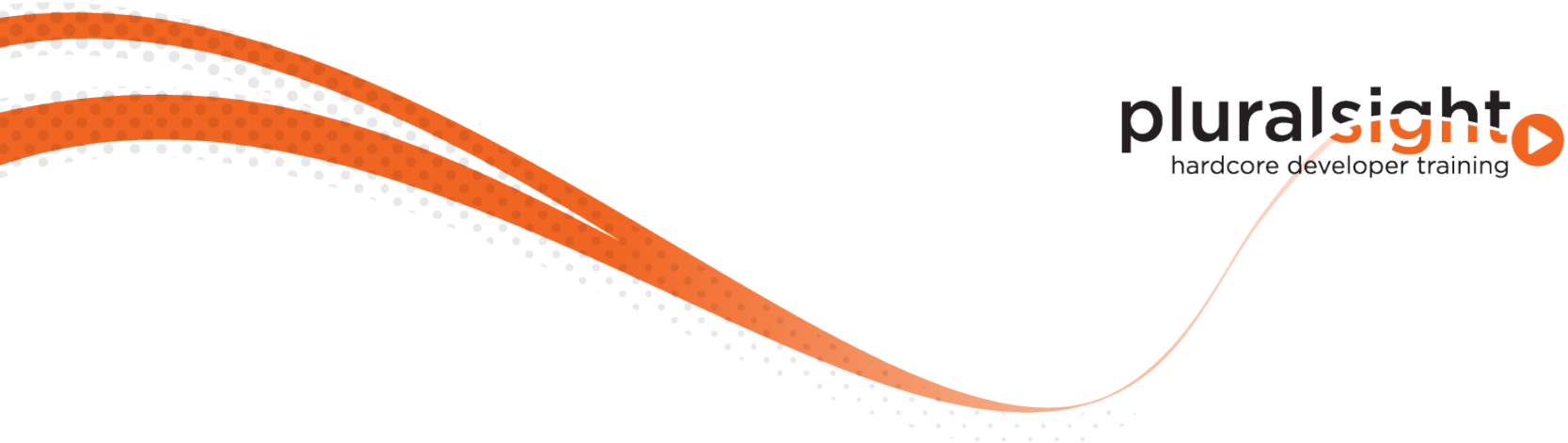


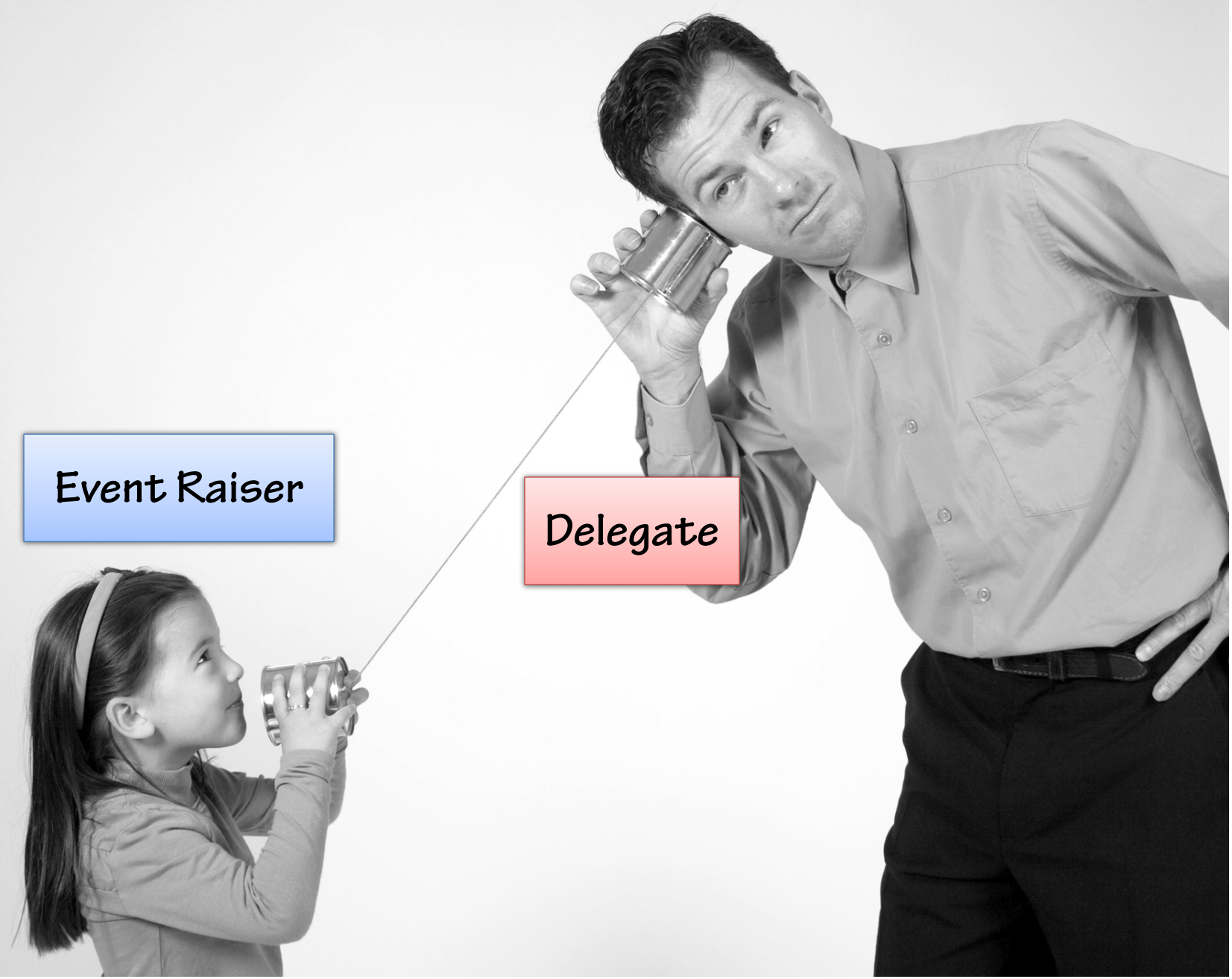
```
static void WorkPerformed1(int hours, WorkType workType)
{
    Console.WriteLine("WorkPerformed1 called");
}
```

Adding to the Invocation List

```
WorkPerformedHandler del1 =
  new WorkPerformedHandler(WorkPerformed1);

WorkPerformedHandler del2 =
  new WorkPerformedHandler(WorkPerformed2);

del1 += del2;

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

Defining an Event




Event Raiser
Delegate

Defining an Event

- Events can be defined in a class using the **event** keyword

```
public event WorkPerformedHandler WorkPerformed;
```

Delegate

Event name

Defining an Event with add/remove

- Events can be defined using add/remove accessors:

```
private WorkPerformedHandler _WorkPerformedHandler;
public event WorkPerformedHandler WorkPerformed
{
    [MethodImpl(MethodImplOptions.Synchronized)]
    add
    {
        _WorkPerformedHandler = (WorkPerformedHandler)Delegate.Combine(
          _WorkPerformedHandler, value);
    }
    [MethodImpl(MethodImplOptions.Synchronized)]
    remove
    {
        _WorkPerformedHandler = (WorkPerformedHandler)Delegate.Remove(
           _WorkPerformedHandler, value);
    }
}
```

Raising Events

pluralsight
hardcore developer training

Raising Events

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

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

- Another option is to access the event's delegate and invoke it directly:

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

Exposing and Raising Events

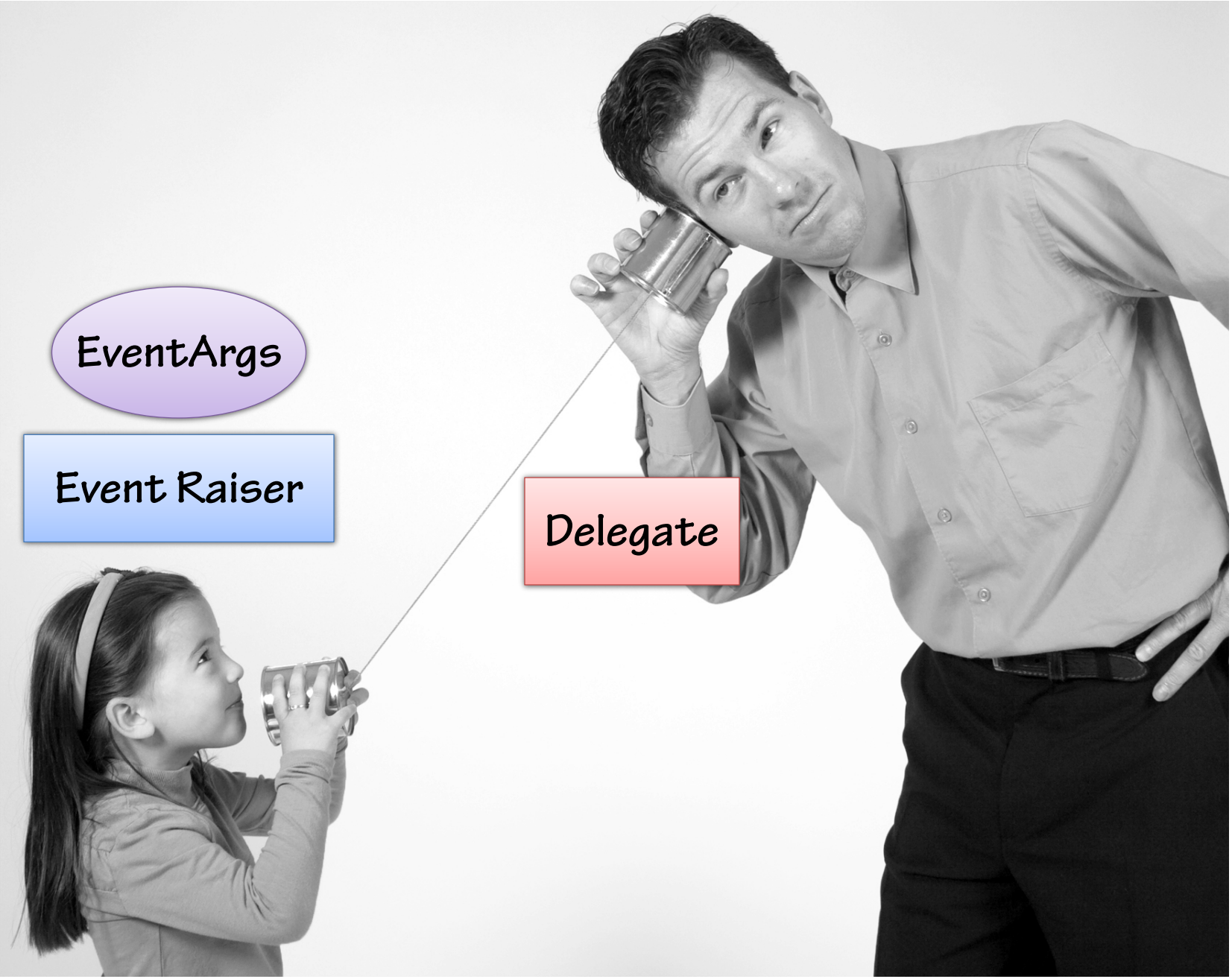
```
public delegate void WorkPerformedHandler(int hours, WorkType workType);
public class Worker
{
    public event WorkPerformedHandler WorkPerformed;
    public virtual void DoWork(int hours, WorkType workType)
    {
        // Do work here and notify consumer that work has been performed
        OnWorkPerformed(hours, workType);
    }

    protected virtual void OnWorkPerformed(int hours, WorkType workType)
    {
         WorkPerformedHandler del = WorkPerformed as WorkPerformedHandler;
         if (del != null) //Listeners are attached
         {
             del(hours, workType);
         }
    }
}
```

Event Definition

Raise Event

Creating an EventArgs Class

pluralsight
hardcore developer training


EventArgs
Event Raiser
Delegate

Creating a Custom EventArgs Class

- The **EventArgs** class is used in the signature of many delegates and event handlers:

```
public void button_Click(object sender, EventArgs e)
{
    // Handle button click
}
```

- When custom data needs to be passed the **EventArgs** class can be extended.

Deriving from the System.EventArgs Class

```
public class WorkPerformedEventArgs : System.EventArgs
{
    public int Hours { get; set; }
    public WorkType WorkType { get; set; }
}
```

Using a Derived System.EventArgs Class

- To use a custom EventArgs class, the delegate must reference the class in its signature:

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

Holds event data

Sender of event

Using EventHandler<T>

.NET includes a generic **EventHandler<T>** class that can be used instead of a custom delegate:

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

```
public event EventHandler<WorkPerformedEventArgs> WorkPerformed;
```

Built-in Delegate

Summary

- Events are associated with Delegates
- A standard .NET event signature accepts:
  - Sender
  - Custom EventArgs
- The parameter signature for an event handler matches the delegate signature
- EventHandler<T> provides a simple way to create a custom delegate for an event