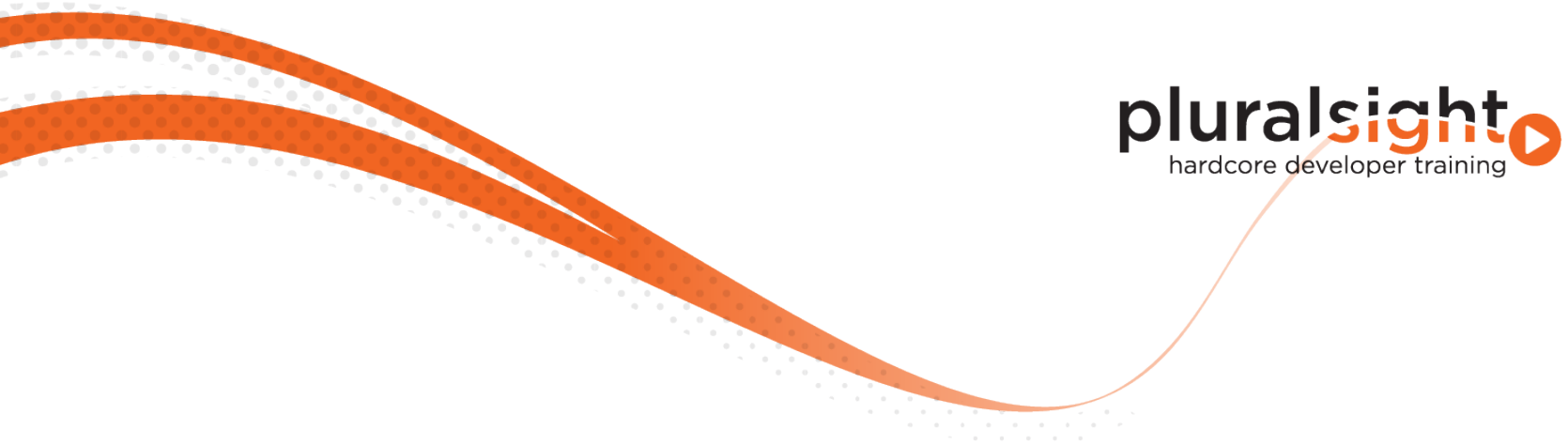
Lambdas, Action<T> and Func<T,TResult>

Dan Wahlin

Twitter: @DanWahlin

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



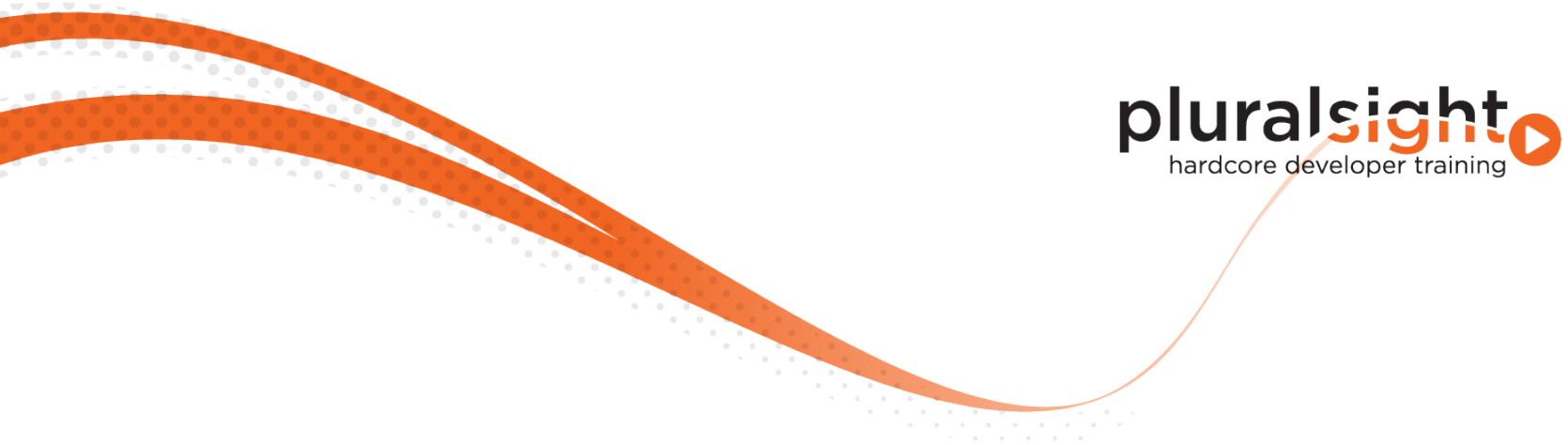
pluralsight
hardcore developer training

Lambdas and Delegates

Using Action<T>

Using Func<T,TResult>

Lambdas and Delegates

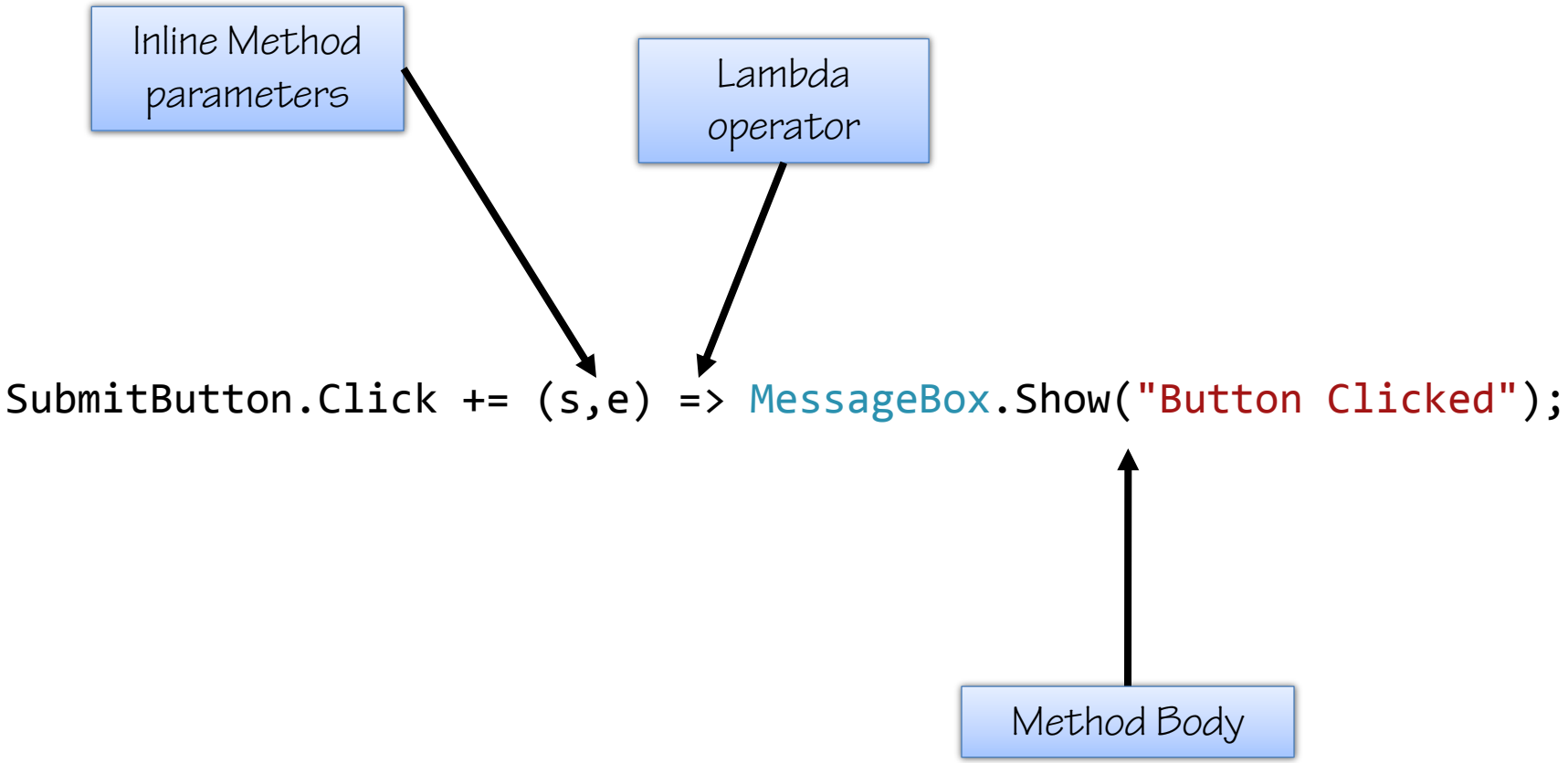


pluralsight
hardcore developer training

Anonymous Methods in Action

```
SubmitButton.Click += delegate(object sender, EventArgs e)
 {
     MessageBox.Show("Button Clicked");
 };
```

Understanding Lambda Expressions



Assigning a Lambda to a Delegate

- Lambda expressions can be assigned to any delegate:

```
delegate int AddDelegate(int a, int b);

static void Main(string[] args){
    AddDelegate ad = (a,b) => a + b;
    int result = ad(1,1); //result = 2
}
```

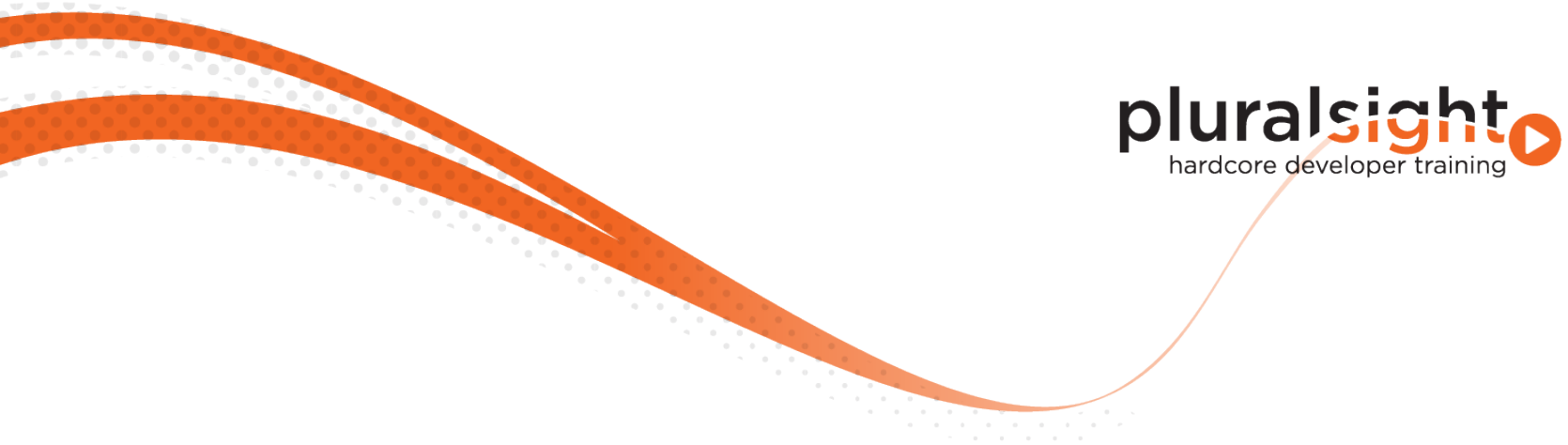
Handling Empty Parameters

- Delegates that don't accept any parameters can be handled using lambdas:

```
delegate bool LogDelegate();

static void Main(string[] args)
{
    LogDelegate ld = () =>
    {
        UpdateDatabase();
        WriteToEventLog();
        return true;
    };
    bool status = ld();
}
```

Using Action<T>



pluralsight
hardcore developer training

Delegates in .NET

- The .NET framework provides several different delegates that provide flexible options:
  - **Action<T>** - Accepts a single parameter and returns no value
  - **Func<T,TResult>** - Accepts a single parameter and return a value of type TResult

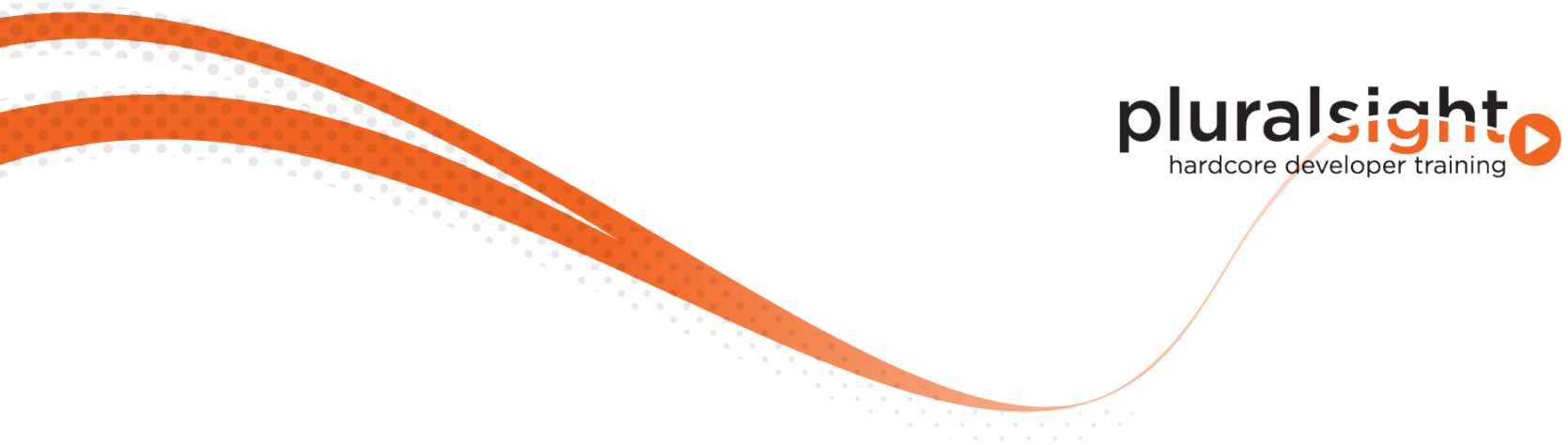
Using Action<T>

- Action<T> can be used to call a method that accepts a single parameter of type T:

```
public static void Main(string[] args)
{
    Action<string> messageTarget;
    if (args.Length > 1) messageTarget = ShowWindowsMessage;
    else messageTarget = Console.WriteLine;
    messageTarget("Invoking Action!");
}

private static void ShowWindowsMessage(string message)
{
    MessageBox.Show(message);
}
```

Using Func<T,TResult>



pluralsight
hardcore developer training

Using Func<T,TResult>

- **Func<T,TResult>** supports a single parameter (T) and returns a value (TResult):

```
public static void Main(string[] args)
{
    Func<string, bool> logFunc;
    if (args[0] == "EventLog") logFunc = LogToEventLog;
    else logFunc = LogToFile;
    bool status = logFunc("Log Message");
}

private static bool LogToEventLog(string message) { /* log */ }
private static bool LogToFile(string message) { /*log */ }
```

Summary

- Lambdas provide a way to define inline methods using a concise syntax

- The .NET Framework provides several built-in delegate types such as:
  - Action<T>
  - Func<T, TResult>