Delegates and Events

Authored by : Sushant Banerjee
Email : sushantba@cybage.com

Presented by : Sushant Banerjee
Extn. 7210

Agenda

- Introduction to Delegates
- Single Cast Delegates
- Multi-Cast Delegates
- Introduction to Events
- Events with Delegates

What is a Delegate

- A delegate is a reference type that allows you to reference a method.
- A delegate is similar to Function Pointers in C and C++.
- Unlike function pointers in C or C++, delegates are object-oriented, type-safe, and secure.
- Why do you need a reference to a method?".
- Because you get maximum flexibility to implement any functionality you want at runtime.

How a Delegate works

- Using a delegate allows the programmer to encapsulate a reference to a method inside a delegate object.
- The delegate object can then be passed to code which can call the referenced method, without having to know at compile time which method will be invoked.
- Delegates are used to pass methods as parameter to other methods such as event handlers.

Declaring a Delegate

- Delegate declarations look somewhat like methods, except they have the delegate modifier, are terminated with a semi-colon (;), and have no implementation.
- public delegate int Comparer(object obj1, object obj2);
- This delegate declaration defines the signature of a delegate handler method that this delegate can refer to.

Using Delegate

- To use a delegate, you must create an instance of it.
- The instance is created, similar to a class instance, with a single parameter identifying the appropriate delegate handler method, as shown below.

```
Comparer cmp = new Comparer(Name.CompareFirstNames);
Or
Comparer cmp = Name.CompareFirstNames;
```

Types of Delegates

- There Are two types of Delegates

- Single-cast Delegate
- Multi-cast delegate

Multi cast Delegate

- Multi-cast delegates can hold reference to more than one methods.
- A delegate object can maintain a list of methods to call.
- To add a method to the invocation list of a delegate object, use the overloaded += operator.
- To remove a method from the invocation list, use the overloaded operator -= .
- The Multicast delegate here contain methods that return void, if you want to create a multicast delegate with return type you will get the return type of the last method in the invocation list.

What is an Event

- A C# event is a class member that is activated whenever the event it was designed for occurs.
- I like to use the term "fires" when the event is activated.
- Anyone interested in the event can register and be notified as soon as the event fires.
- At the time an event fires, registered methods will be invoked.

How it Works

- The publisher determines when an event is raised; the subscribers determine what action is taken in response to the event.
- An event can have multiple subscribers. A subscriber can handle multiple events from multiple publishers.
- Events are typically used to signal user actions such as button clicks or menu selections in graphical user interfaces.
- In the .NET Framework class library, events are based on the EventHandler delegate and the EventArgs base class.

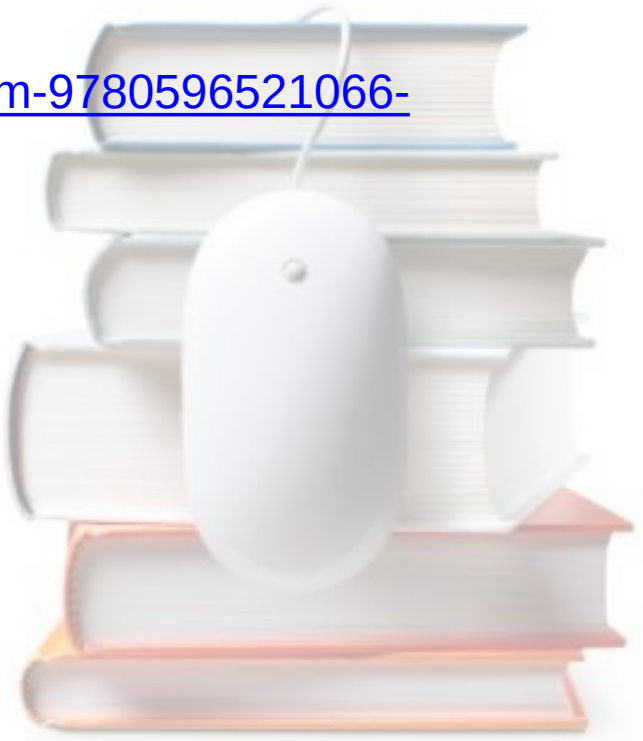
Delegate and Event

- Events and delegates work hand-in-hand to provide a program's functionality.
- It starts with a class that declares an event.
- Any class, including the same class that the event is declared in, may register one of its methods for the event.

Demo

Bibliography, Important Links

- http://www.codeproject.com/Articles/4773/Events-and-Delegates-Simplified
- http://msdn.microsoft.com/en-in/library/orm-9780596521066-01-17.aspx



Any Questions?

Thank you!