Events Overview (Windows Forms)

.NET Framework 4.5 2 out of 4 rated this helpful

An event is an action which you can respond to, or "handle," in code. Events can be generated by a user action, such as clicking the mouse or pressing a key; by program code; or by the system.

Event-driven applications execute code in response to an event. Each form and control exposes a predefined set of events that you can program against. If one of these events occurs and there is code in the associated event handler, that code is invoked.

The types of events raised by an object vary, but many types are common to most controls. For example, most objects will handle a Click event. If a user clicks a form, code in the form's Click event handler is executed.

Note

Many events occur in conjunction with other events. For example, in the course of the DoubleClick event occurring, the MouseDown, MouseUp, and Click events occur.

For information about how to raise and consume an event, see Handling and Raising Events. For an example of how to connect event handlers and methods, see How to: Connect Event Handler Methods to Events.

Delegates and Their Role

Delegates are classes commonly used within the .NET Framework to build event-handling mechanisms. Delegates roughly equate to function pointers, commonly used in Visual C++ and other object-oriented languages. Unlike function pointers however, delegates are object-oriented, type-safe, and secure. In addition, where a function pointer contains only a reference to a particular function, a delegate consists of a reference to an object, and references to one or more methods within the object.

This event model uses *delegates* to bind events to the methods that are used to handle them. The delegate enables other classes to register for event notification by specifying a handler method. When the event occurs, the delegate calls the bound method. For more information about how to define delegates, see Handling and Raising Events.

Delegates can be bound to a single method or to multiple methods, referred to as multicasting. When creating a delegate for an event, you (or the Windows Forms Designer) typically create a multicast event. A rare exception might be an event that results in a specific procedure (such as displaying a dialog box) that would not logically repeat multiple times per event. For information about how to create a multicast delegate, see How to: Combine Delegates (Multicast Delegates)(C# Programming Guide).

A multicast delegate maintains an invocation list of the methods it is bound to. The multicast delegate supports a Combine method to add a method to the invocation list and a Remove method to remove it.

When an event is recorded by the application, the control raises the event by invoking the delegate for that event. The delegate in turn calls the bound method. In the most common case (a multicast delegate) the delegate calls each bound method in the invocation list in turn, which provides a one-to-many notification. This strategy means that the control does not need to maintain a list of target objects for event notification—the delegate handles all registration and notification.

Delegates also enable multiple events to be bound to the same method, allowing a many-to-one notification. For example, a button-click event and a menu-command–click event can both invoke the same delegate, which then calls a single method to handle these separate events the same way.

The binding mechanism used with delegates is dynamic: a delegate can be bound at run time to any method whose signature matches that of the event handler. With this feature, you can set up or change the bound method depending on a condition and to dynamically attach an event handler to a control.

See Also

Concepts

Event Handlers Overview (Windows Forms)

Other Resources

Creating Event Handlers in Windows Forms

Was this page helpful? ○ Yes ○ No