La gestione degli eventi

La **gestione degli eventi** nel Framework .NET 2.0 e in Visual Studio 2005 è rimasta pressoché immutata rispetto alle versioni precedenti. Continuiamo a lavorare con la Windows Form realizzata nella scorsa lezione, precisando che quanto diremo in seguito a proposito degli eventi vale **per ogni tipo di controllo**, sia esso una form, un pulsante, una casella di testo, un comando di menu, etc.
Possiamo definire un evento come il **verificarsi di una condizione**: dalla pressione di un pulsante, alla digitazione in una casella di testo. Quando si verifica una di queste condizioni, diciamo che il controllo genera (oppure lancia) un evento. Ad esempio, quando si preme il pulsante sinistro del mouse su di un bottone, esso genera l'evento «Click».
Ad ogni evento può essere associata una azione descritta da una particolare routine che viene eseguita ogni volta che l'evento si verifica.

Supponiamo di voler eseguire una certa azione in fase di caricamento del form. Occorre anzitutto creare un **gestore di eventi** (detto Event Handler) per l'evento «Load» della finestra Form1. Aggiungiamo il seguente codice nel costruttore della classe:

```
this.Load += new EventHandler(Form1_Load);
```

Per installare un event handler, occorre specificare il nome dell'oggetto che espone l'evento: in questo caso abbiamo usato this, che, come abbiamo già accennato, rappresenta la classe corrente (in questo caso il form).
Di seguito, dopo il punto, si indica il nome dell'**evento da gestire**: gli eventi disponibili sono riconoscibili, nell'elenco visualizzato tramite l'IntelliSense, dal simbolo di un fulmine.

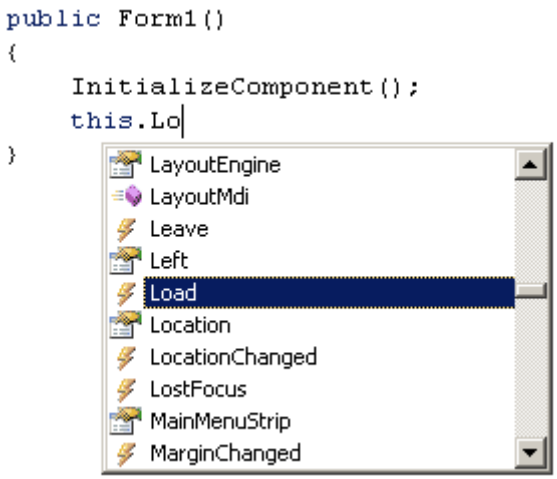

Figura 1: Gli eventi del form

L' **operatore '+='** indica che si sta aggiungendo un gestore per l'evento. Si usa '+=', e non solo '=', perché è possibile specificare **più handler per lo stesso evento.**
L'IntelliSense ci viene ancora una volta in aiuto, proponendoci un nome per il gestore degli eventi: è sufficiente premere il tasto TAB per confermare il suggerimento.

Il metodo che rappresenta l'azione collegata all'evento viene detto "**delegato**" (delegate). Infatti viene delegata a quel metodo la gestione dell'evento. Form1_Load() in questo caso è un delegato. Questo metodo di creazione degli event handler corrisponde ad «AddHandler» in VB.NET

EventHandler e tutte le classi derivate sono gestori di eventi. Ogni oggetto istanza di EventHanlder richiede il **riferimento a un delegato** per la gestione di un evento, la stringa "Form1_Load" senza parentesi altro non è che un riferimento (puntatore) al metodo Form1_Load().
Un delegato in astratto ha però una certa **firma** e per rimpiazzarlo con un altro metodo come Form_Load() occorre che questo abbia i medesimi argomenti in tipo e numero e lo stesso tipo di ritorno.
Con EventHandler(Form1_Load) si esprime il fatto che il metodo Form1_Load() avrà argomenti uguali in numero e tipo a quelli indicati nella dichiarazione del delegate EventHandler e avrà come tipo di ritorno il medesimo tipo del delegate.

Un'altra pressione del tasto TAB aggiunge automaticamente, all'interno della classe, il metodo che sarà richiamato quando si genera l'evento in questione:

```
void Form1_Load(object sender, EventArgs e)
{
    throw new Exception("Metodo non implementato.");
}
```

L'istruzione «throw new Exception()» solleva una eccezione. Ci occuperemo delle eccezioni in seguito.

Gli argomenti di questo metodo sono due:

- **Object sender** rappresenta l'oggetto (il controllo, la classe) che ha generato l'evento, ed è utile nel caso in cui la stessa routine venga eseguita in risposta ad eventi lanciati da oggetti diversi, per sapere chi effettivamente lo ha generato;
- **EventAgrs e**. EventArgs, come pure tutte le classi da essa derivate, contiene gli argomenti associati all'evento. Nel nostro caso «e» è di tipo EventArgs, ma, può essere specializzata a seconda dell'evento.
  Ad esempio, nel caso di eventi di gestione del mouse (MouseClick,MouseMove, etc.), è di tipo «MouseEventArgs» e permette di conoscere la posizione del cursore o quale tasto del mouse è stato premuto.

Proviamo ad utilizzare la routine Form1_Load. Vogliamo fare in modo che, all'avvio del programma, venga visualizzata una MessageBox con la scritta "Ciao Mondo!". A tal scopo, è sufficiente definire il metodo Form1_Load nel modo seguente:

```
void Form1_Load(object sender, EventArgs e)
{
  MessageBox.Show(
    "Ciao Mondo!", "Evento",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

MessageBox è una classe che contiene un metodo statico, **Show()** che visualizza una finestra di messaggio. Esso dispone di parecchie

definizioni tramite overloading (21 per la precisione); la versione più utilizzata, anche da noi, prende come argomenti il messaggio da visualizzare, il titolo della MessageBox, i pulsanti e l'icona della finestra.
Eseguiamo ora l'applicazione facendo clic sul pulsante della barra degli strumenti (oppure premendo il tasto F5). Se non sono stati commessi errori, dopo la compilazione dovrebbe comparire la finestra riportata in figura.

Figura 2: L'output del programma

Gli event handler possono essere creati anche in **un altro modo.**
All'interno della finestra di progettazione, selezionare il controllo per cui si vuole definire il gestore, quindi spostarsi nella finestra **Properties** (Proprietà) e fare clic sull'icona raffigurante un fulmine.
Comparirà un elenco contenente tutti gli eventi generati dal controllo in questione. Selezionandone uno, nella parte bassa della finestra sarà visualizzato un breve messaggio che indica quando tale evento viene lanciato.

Clicchiamo su un evento, digitiamo il nome di una routine nella casella di testo corrispondente e premiamo il tasto INVIO, Visual Studio aggiungerà automaticamente il codice per installare l'event handler (all'interno del file del Designer) ed inserirà nel file del form una routine con il nome indicato.

In alternativa, facendo doppio clic sul nome dell'evento, sarà creato un event handler e una routine di gestione con il nome predefinito (formato da nome del controllo, trattino basso e nome dell'evento).

Per eliminare un gestore degli eventi creato in questo modo, fare clic con il tasto destro del mouse sull'evento e selezionare il comando **Reset** (Annulla). Infine, per tornare alla visualizzazione delle proprietà del controllo, premere il

pulsante **Properties** (Proprietà) che si trova a sinistra dell'icona con il fulmine.

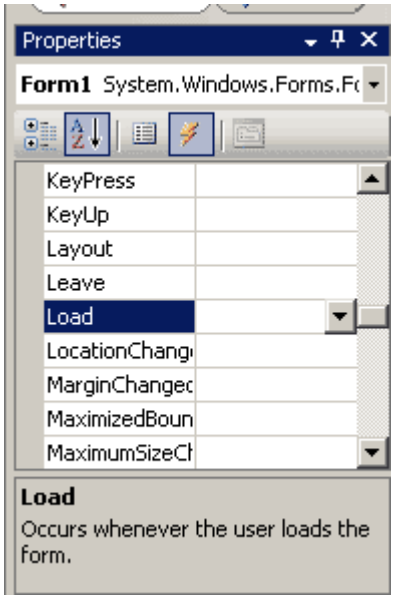

Figura 3: La finestra per la definizione degli event handler

Ancora, facendo doppio clic su di un oggetto, Visual Studio definirà automaticamente un event handler per l'evento predefinito dell'oggetto in questione (Load per i form, Click per i pulsanti, **TextChanged** per le caselle di testo, ecc.).

Questa rapida panoramica sulla Windows Form ha lo scopo di fornire le conoscenze di base che permettano di approfondire l'argomento senza troppa fatica.