**GIỚI THIỆU**

**User control:** là loại control đơn giản nhất. Chúng kế thừa từ lớp System.Windows.Forms.UserControl và theo mô hình thành phần. Là một container chứa tất các cả các control khác, thông thường người dùng sẽ kết hợp nhiều hơn một control có sẵn. Thông thường, các user control kết hợp nhiều hơn một control trong một đơn vị logic (như một nhóm text box để nhập thông tin địa chỉ, một nhóm button,…).

Nhúng user control trong window form cũng giống như thêm button đơn giản, các text box đã được cung cấp sẵn bởi .NET.

Trong .NET, việc tạo một user control tùy chỉnh cũng như tạo một lớp thông thường. Mình có thể tạo những lớp control.

**GIAO TIẾP**

Vì trong của basic .NET form, controls chứa những user control.

Những events của user control sẽ không được kích hoạt cho các ứng dụng, form chứa nó.

Vì vậy các user control sẽ được xử lý như mọi đối tượng khác và phải triển khai các thuộc tính riêng của nó (bên cạnh các thuộc tính được kế thừa từ System.Windows.Forms.Control) và các sự kiện.

\*NOTE

**Types of Custom Controls**

Developers often make a distinction between three or four types of controls:

|  |
| --- |
| * **User controls** are the simplest type of control. They inherit from the System.Windows.Forms.UserControl class, and follow a model of composition. Usually, user controls combine more than one control in a logical unit (like a group of text boxes for entering address information). |
| * **Inherited controls** are generally more powerful and flexible. With an inherited control, you choose the existing .NET control that is closest to what you want to provide. Then, you derive a custom class that overrides or adds properties and methods. The examples you've looked at so far in this book, including the custom TreeViews and ListViews, have all been inherited controls. |
| * **Owner-drawn controls** generally use GDI+ drawing routines to generate their interfaces from scratch. Because of this, they tend to inherit from a base class like System.Windows.Forms.Control. Owner-drawn controls require the most work and provide the most customizable user interface. |
| * **Extender providers**, which aren't necessarily controls at all. These components add features to other controls on a form, and provide a remarkable way to implement extensible user interfac |