Tự học
Ngôn ngữ lập trình

Visual Basic 2005

(Dành cho Học sinh - Sinh viên)

KS : NGUYỄN NAM THUẬN

Tự học

Ngôn ngữ lập trình

VISUAL BASIC 2005

(Dành cho học sinh - sinh viên)

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

LỜI NÓI ĐẦU

Cuốn sách "**Tự học ngôn ngữ lập trình Visual Basic 2005**" này được biên soạn nhằm giới thiệu đến các bạn học sinh và sinh viên những tính năng mới trong Visual Basic 2005, .NET Framework 2.0 và công cụ phát triển Visual Studio 2005 thông qua một loạt các bài thực hành với những ví dụ tập trung và súc tích.

Sách được phân thành 6 chương. Mỗi chương trình bày một hạng mục cụ thể trong các phần cải tiến của ngôn ngữ VB 2005 và .NET Framework 2.0:

Chương 1: Visual Studio 2005. Chương này trình bày các điểm nổi bật của Visual Studio 2005, bao gồm IntelliSense và một tính năng code snippet mới với các ví dụ hữu dụng, đặc biệt là sự trở lại của công cụ gỡ rối edit-and-continue (hiệu chỉnh-và-tiếp tục).

Chương 2: Ngôn ngữ Visual Basic 2005. Chương này giới thiệu một số từ khóa mới, trong đó có vài tính năng được tìm thấy trong C# như quá tải toán tử và các tính năng .NET Framework hoàn toàn mới.

Chương 3: Các ứng dụng Windows. Bộ công cụ Windows Forms chưa thay đổi nhiều nhưng các control mới đem đến diện mạo mới cho các thanh công cụ và menu, sự hiệu chỉnh text được tạo mặt nạ và hiển thị trang Web.

Chương 4: Các ứng dụng Web. Trong chương này, bạn sẽ được giới thiệu tổng quan về nhiều tính năng mới, gồm kết buộc dữ liệu không cần viết mã, định hướng site và các giải pháp mới để cá nhân hóa.

Chương 5: Các file, cơ sở dữ liệu và XML. Chương này trình bày các lớp truy cập file, vài điểm mới về ADO.NET và một cách hay hơn để làm việc với XML.

Chương 6: Các dịch vụ nền .NET 2.0. Chương này bao gồm các chủ điểm liên quan đến các tính năng .NET Framework mới, sự hỗ trợ FTP, truy cập hệ thống bảo mật Windows user và triển khai các ứng dụng với công nghệ ClickOnce.

Với cách bố cục như trên, chúng tôi hy vọng cuốn sách này sẽ là một tài liệu tham khảo thật sự hữu ích cho bạn đọc. Chúng tôi rất mong đón nhận sự đóng góp ý kiến quí báu từ bạn đọc. Xin chân thành cảm ơn.

Tác giả.

Chương 1

Visual Studio 2005

Các tính năng mới của Visual Basic 2005 thật ra được cung cấp bởi ba thành phần riêng biệt: Visual Studio 2005 IDE cải tiến, một phiên bản mới của trình biên dịch VB (vbc.exe), và .NET 2.0 Framework đã sửa đổi. Trong chương này, bạn sẽ bắt đầu bằng việc khảo sát Visual Studio 2005.

Ghi chú

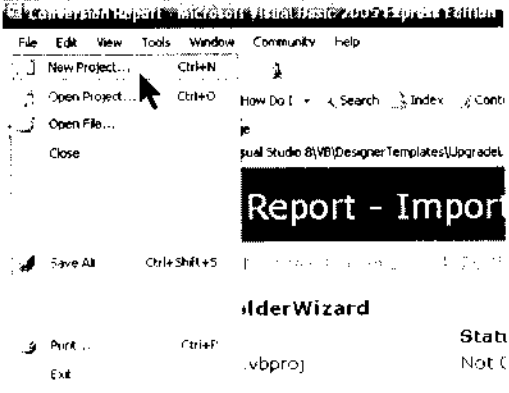
Thoạt đầu, Visual Studio có vẻ không có gì thay đổi nhưng nó muốn bạn đến IDE mới nhất của Microsoft.

Thủ thuật

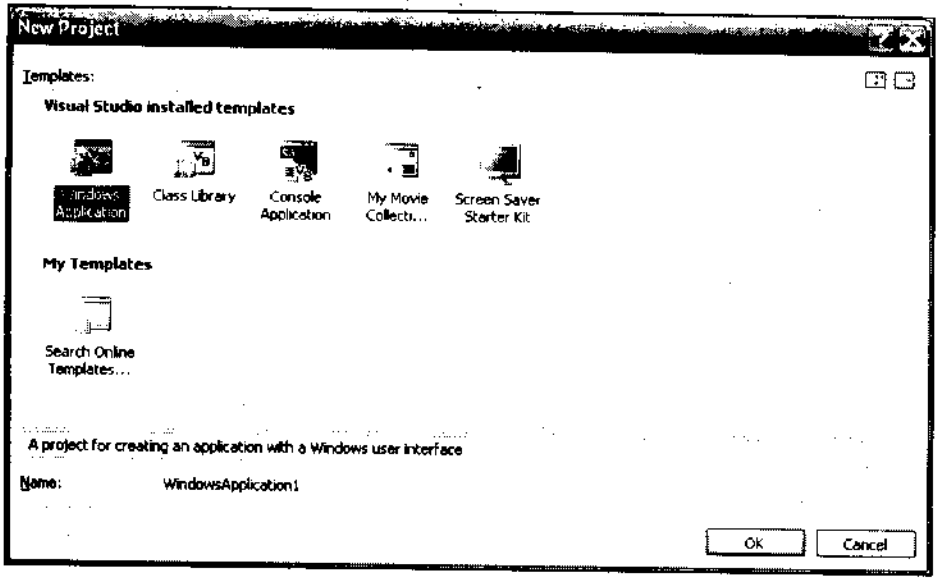
Visual Studio 2005 là hậu duệ trực tiếp của Visual Studio .NET, nó cung cấp bộ công cụ và các tính năng hoàn chỉnh nhất. Visual Basic 2005 Express Edition cho phép bạn tạo các ứng dụng Windows, các ứng dụng bàn điều khiển (console), và các thành phần DLL (nhưng không cho phép bạn tạo các ứng dụng web). Visual Web Developer 2005 Express Edition cho phép bạn tạo chỉ các ứng dụng web. Tuy nhiên, cả ba chương trình này thật sự là những dạng biến đổi của cùng công cụ Visual Studio. Do vậy, các menu, các thanh công cụ, và cách hoạt động của những ứng dụng này về cơ bản là giống nhau.

1.1. Bạn làm điều đó như thế nào

Để bắt đầu tạo một project (dự án) mới, chọn File > New Project từ menu Visual Studio.

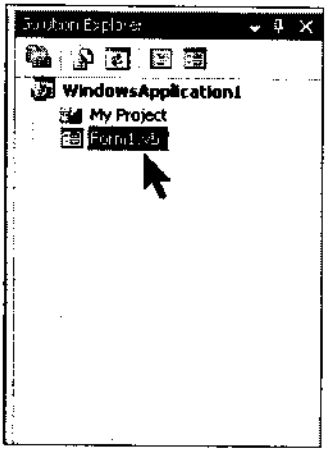


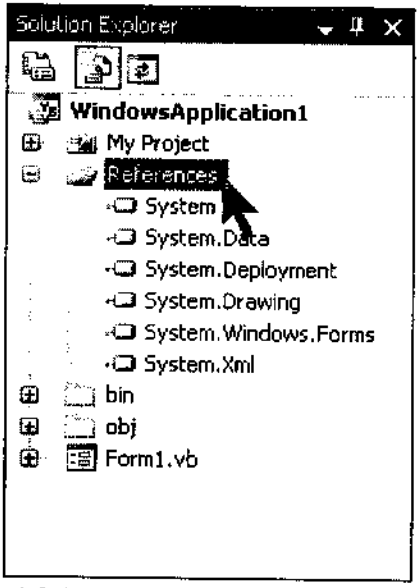
Bạn sẽ thấy một hộp thoại New Project đã được sửa đổi đôi chút, như minh họa ở hình 1.1. Tùy thuộc vào phiên bản nào của Visual Studio mà bạn đang sử dụng, bạn có thể thấy một hệ thống khác của các loại project có sẵn.



Hình 1.1. Tạo một project mới

Để tiếp tục, chọn loại project Windows Application và nhấp OK để tạo project mới. Trong Solution Explorer, bạn sẽ thấy project có chứa một form duy nhất, một file cấu hình ứng dụng và một nút My Project (bạn có thể chọn nút này để cấu hình các xác lập tạo project). Tuy nhiên, danh sách các phần tham chiếu hợp ngữ sẽ không xuất hiện trong Solution Explorer, trừ khi bạn trực tiếp chọn Project > Show All Files. Hình 1.2 minh họa cả hai phiên bản của Solution Explorer.





Hình 1.2. Hai khung xem của Solution Explorer

Để lưu project, chọn File > Save [ProjectName] từ menu. Một sự thay đổi mà bạn có thể nhận thấy là Visual Studio không còn yêu cầu bạn chỉ định một đường dẫn thư mục khi bạn tạo project mới nữa. Đó là vì Visual Studio không lưu bất kỳ file nào chừng nào bạn chưa yêu cầu nó.

Thủ thuật

Thật ra hoạt động này phụ thuộc vào các xác lập môi trường Visual Studio. Khi lần đầu tiên bạn cài đặt Visual Studio, bạn có cơ hội để chọn project của nhà phát triển. Nếu bạn chọn Visual Basic Development Settings, bạn sẽ không được yêu cầu lưu project của mình khi lần đầu tiên bạn tạo nó.

Dĩ nhiên, là một nhà lập trình hiểu biết, bạn sẽ biết các file cần được định vị ở một nơi nào đó, và nếu bạn đi sâu hơn, bạn sẽ thấy một thư mục tạm thời như C:\Documents and Settings\[UserName]\Local Settings\Application Data\Temporary Projects\[ProjectName] vốn được sử dụng tự động để chứa các project mới chưa được lưu. Ngay sau khi bạn lưu một project, nó sẽ được chuyển đến nơi bạn chọn.

Ghi chú

Tiến trình tạo các ứng dụng web cũng đã thay đổi khá tinh vi trong Visual Studio 2005, bây giờ bạn không còn cần IIS và một thư mục ảo để kiểm tra web site của bạn nữa. Bạn sẽ tìm hiểu thêm về các web project ở chương 4.

Bạn có thể sử dụng ứng dụng Windows đơn giản mà bạn đã tạo để thực hiện các thử nghiệm khác trong chương này và khảo sát các tính năng mới của Visual Studio.

1.2. Tạo mã, gỡ rối và tiếp tục mà không cần khởi động lại trình ứng dụng

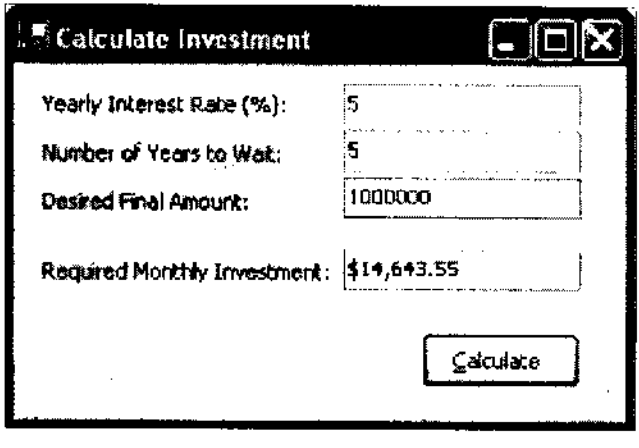
Các nhà phát triển Visual Basic 6 đã quen thực hiện sự thay đổi, điều chỉnh các câu lệnh, tin chỉnh logic, và thậm chí chèn các khối mã hoàn toàn mới trong khi họ làm việc. Nhưng sự giới thiệu kiểu kiến trúc mới lúc biên dịch với .NET 1.0 CLR (Common Language Runtime) đã làm cho tính năng này biến mất khỏi Visual Studio .NET 2002 và 2003. Thật may là nó đã được đưa trở lại vào Visual Basic 2005, với một số phần cải tiến và một điểm lưu ý quan trọng là nó không hoạt động với ASP.NET.

Ghi chú

Tính năng được yêu cầu nhiều nhất từ VB 6 trở lại .NET: một trình gỡ rối cho phép bạn hiệu chỉnh mã mà không cần khởi động lại trình ứng dụng của bạn.

1.2.1. Bạn làm điều đó như thế nào?

Để xem phần hiệu chỉnh và gỡ rối ở khía cạnh đơn giản nhất của nó, chúng ta sẽ xem xét một ví dụ mà trong đó một sự cố đã làm sai lệch mã của bạn và cách bạn có thể phục hồi nó nhanh chóng. Hình 1.3 minh họa một ứng dụng máy tính tài chính (financial calculator) có khả năng tính khoảng thời gian cần thiết để bạn trở thành một triệu phú, sử dụng hàm Pmt ( ) tiện lợi của Visual Studio.



Hình 1.3. Một form đơn giản cho một phép tính tài chính

Để tạo chương trình này, trước tiên bạn thêm bốn hộp text (nhãn là tùy ý), rồi đặt tên cho chúng là txtInterestRate, txtYears, txtFutureValue, và txtMonthlyPayment (từ trên xuống). Sau đó, thêm một nút với event handler (trình điều hoãn sự kiện) sau đây:

```
Private Sub btnCalculate_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnCalculate.Click

  Dim InterestRate, Years, FinalValue, MonthlyPayment As Double
  InterestRate = Val(txtInterestRate.Text)
  FinalValue = Val(txtFutureValue.Text)
  MonthlyPayment = Pmt(InterestRate / 12 / 100, _
     Years * 12, 0, -FinalValue, DueDate.BegOfPeriod)
  txtMonthlyInvestment.Text = MonthlyPayment.ToString("C")

End Sub
```

Bây giờ hãy chạy ứng dụng, nhập một vài giá trị mẫu và nhấp nút. Bạn sẽ nhận được một ngoại lệ runtime (với phần giải thích "Argument NPer is not a valid value", nghĩa là đối số NPer không phải là một giá trị hợp lệ) khi mã của bạn cố tính trị MonthlyPayment. Một cách để phát hiện nguyên nhân gây ra sự cố là di chuyển chuột qua tất cả các thông số trong câu lệnh và xác nhận rằng chúng phản ánh đúng những gì bạn mong muốn. Trong trường hợp này, sự cố là biến Years chưa bao giờ được xác lập và do đó chứa trị 0.

Nhờ hiệu chỉnh và gỡ rối liên tục, bạn có thể xử lý sự cố này mà không cần khởi động lại trình ứng dụng của bạn. Khi lỗi xảy ra, nhấp liên kết "Enable editing" trong cửa sổ lỗi. Sau đó thêm dòng còn sót sau đây:

```
Private Sub btnCalculate_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnCalculate.Click

  Years = Val(txtYears.Text)
  ...

End Sub
```

Bây giờ, hãy tìm mũi tên màu vàng trong lề vốn cho biết trình gỡ rối (debugger) đang ở nơi nào trong mã. Nhấp và rê mũi tên màu vàng này lên đến dòng mới được bổ sung vào để nó sẽ được thực thi vào lần kế

tiếp. Khi bạn nhấn F5 hoặc nhấp nút Start, mã tiếp tục từ điểm này và phép tính hoàn tất.

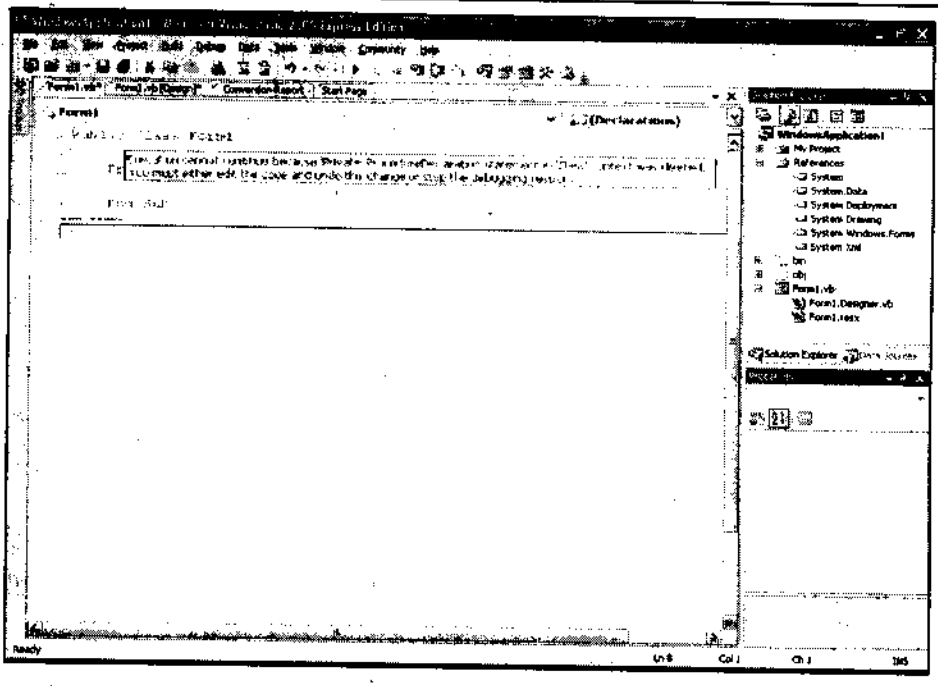
Ghi chú

Bạn không cần chờ đợi một lỗi xảy ra để sử dụng kỹ thuật hiệu chỉnh và gỡ rối liên tục. Bạn cũng có thể xác lập một điểm ngắt trong mã và chọn Debug > Break từ menu vào bất kỳ lúc nào.

1.2.2. Thế còn...

... các thay đổi mà trình gỡ rối edit-and-continue (hiệu chỉnh và tiếp tục) không hỗ trợ thì sao? Phần lớn kỹ thuật hiệu chỉnh và gỡ rối liên tục trong Visual Basic 2005 hỗ trợ nhiều thao tác hiệu chỉnh hơn so với trong Visual Basic 6. Tuy nhiên, vẫn có một số loại hiệu chỉnh đòi hỏi bạn phải khởi động lại trình ứng dụng. Một ví dụ là nếu bạn xóa phương thức mà mã của bạn đang thực thi bằng phương thức đó. Để xem danh sách đầy đủ, bạn hãy vào phần trợ giúp MSDN, bên dưới mục index "Edit and Continue > unsupported declaration edits" (mô tả các thay đổi không được phép cho các phần khai báo, như các thuộc tính, phương thức, và các lớp) và "Edit and Continue > unsupported property and method body edits" (mô tả các phần thay đổi không được phép bên trong các thường trình mã thật sự của bạn).

Để cảnh báo bạn khi nào bạn thực hiện một phần hiệu chỉnh không được hỗ trợ, Visual Studio gạch dưới phần khai báo của lớp hiện hành bằng một đường nguệch ngoạc màu xanh lục. Nếu bạn tựa con trỏ trên đường đó, một ToolTip xuất hiện giải thích sự thay đổi không được phép. Hình 1.4 minh họa một ví dụ.



Hình 1.4. Một ToolTip giải thích phần hiệu chỉnh không được hỗ trợ

Vào lúc này, bạn có thể hoặc undo sự thay đổi này hoặc tiếp tục (biết rằng bạn sẽ cần khởi động lại chương trình). Nếu bạn cố tiếp tục thực thi (bằng cách nhấn F5 hay F8), Visual Studio hỏi bạn muốn ngưng gỡ rối hay muốn hủy yêu cầu và tiếp tục hiệu chỉnh mã câu hỏi.

Một hạn chế quan trọng hơn của tính năng edit-and-continue mới là nó không hỗ trợ các ứng dụng web ASP.NET. Tuy nhiên, các nhà phát triển Visual Studio (và C#) vẫn nhận được một vài cải tiến trong kinh nghiệm gỡ rối ứng dụng web của mình. Visual Studio 2005 biên dịch mỗi trang web một cách riêng biệt, thay vì vào một hợp ngữ duy nhất (như mô hình trong các phiên bản trước đây). Do vậy, khi bạn phát hiện một số mã bị sai trong trang web, bạn có thể tạm ngưng trình gỡ rối, hiệu chỉnh mã, và làm mới trang web bằng cách nhấp Refresh trong trình duyệt của bạn. Cách xử lý này giúp bạn có thêm một kinh nghiệm tương tự như edit-and-continue, nhưng nó chỉ có tác dụng trên cơ sở từng trang. Thật không may, tính năng này sẽ không giúp ích cho bạn nếu bạn đang ở giữa tiến trình gỡ rối một thường trình phức tạp bên trong một trang web. Trong trường hợp đó, bạn vẫn cần yêu cầu lại trang web sau khi bạn thực hiện sự thay đổi và bắt đầu lại.

1.3. Tìm bên trong một đối tượng khi đang gỡ rối

Visual Studio luôn luôn tạo cơ hội cho bạn nhìn vào các biến (variables) trong khi đang gỡ rối mã, chỉ bằng việc tựa con trỏ chuột trên

chúng. Nhưng luôn có hạn chế. Nếu biến là một thể hiện của một đối tượng, tất cả những gì bạn thấy là trị được trả về bởi phương thức ToString ( ), không chỉ đơn giản là tên được định tính đầy đủ của chính lớp. Hơn nữa, bạn không thể xem nội dung của các bộ tạo chỉ mục và các thuộc tính chung. Các cửa sổ Watch và Locals đã có thêm một vài cải tiến, nhưng chúng không hoàn toàn tiện lợi. Visual Studio 2005 đã thay đổi hình ảnh với một tính năng mới được gọi là trình gỡ rối DataTips.

Ghi chú

Trong Visual Studio 2005, việc xem nội dung của các đối tượng phức tạp trong khi gỡ rối còn dễ dàng hơn.

1.3.1. Bạn làm điều đó như thế nào?

Để sử dụng trình gỡ rối DataTips, bạn cần có một lớp tùy ý để làm việc. Mã trong ví dụ 1.1 minh họa phần khai báo cho hai lớp rất đơn giản biểu thị các nhân viên và phòng ban.

Ví dụ 1.1. Hai lớp đơn giản

```
Public Class Employee
    Private _ID As String
    Public ReadOnly Property ID( ) As String
        Get
            Return _ID
        End Get
    End Property

    Private _Name As String
    Public ReadOnly Property Name( ) As String
        Get
            Return _Name
        End Get
    End Property

    Public Sub New(ByVal id As String, ByVal name As String)
        _ID = id
        _Name = name
    End Sub
End Class
```

```
Public Class Department
    Private _Manager As Employee
    Public ReadOnly Property Manager( ) As Employee
        Get
            Return _Manager
        End Get
    End Property

    Private _DepartmentName As String
    Public ReadOnly Property Name( ) As String
        Get
            Return _DepartmentName
        End Get
    End Property

    Public Sub New(ByVal departmentName As String, ByVal manager As Employee)
        _DepartmentName = departmentName
        _Manager = manager
    End Sub
End Class
```

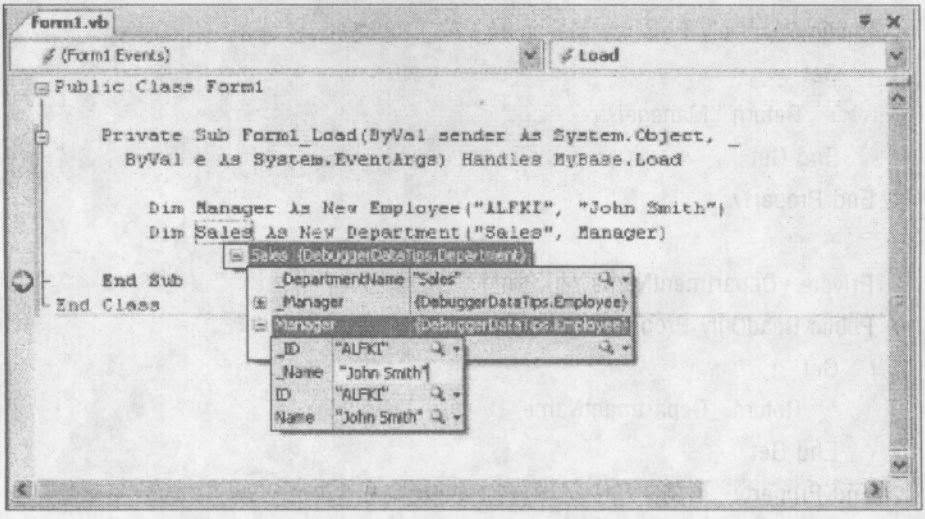
Bây giờ bạn có thể bổ sung thêm mã vốn sử dụng các đối tượng này. Thêm event handler sau đây vào bất kỳ form để tạo một đối tượng Employee và Department mới khi form tải lần đầu tiên.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles MyBase.Load

    Dim Manager As New Employee("ALFKI", "John Smith")
    Dim Sales As New Department("Sales", Manager)

End Sub
```

Đặt một điểm ngắt trên End Sub cuối cùng và chạy ứng dụng. Khi sự thực thi ngưng ở dòng cuối cùng, hãy tựa con trỏ chuột trên biến Sales. Một ToolTip mở rộng sẽ xuất hiện, liệt kê mọi thành viên chung và riêng của đối tượng đó.

Thậm chí tốt hơn, nếu một đối tượng tham chiếu một đối tượng khác,

bạn có thể đi sâu vào các chi tiết của cả hai đối tượng. Để thử điều này, bạn nhấp dấu cộng (+) kế bên thuộc tính Manager để xem đối tượng Employee đã liên kết. Hình 1.5 minh họa DataTip mà bạn sẽ nhìn thấy.



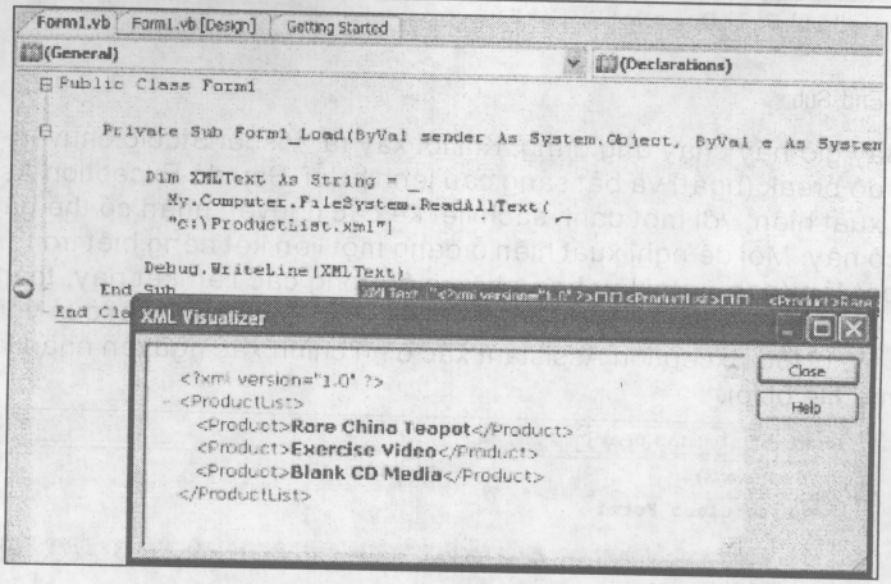
Hình 1.5. Nhìn vào một đối tượng

Nhờ sử dụng trình gỡ rối DataTips, bạn cũng có thể hiệu chỉnh các kiểu dữ liệu đơn giản. Chỉ cần nhấp double vào thuộc tính hay biến và hiệu chỉnh trị. Trong hình 1.5, biến riêng _Name đang được hiệu chỉnh.

1.3.2. Làm việc với các kiểu dữ liệu ngoại lai

Bằng cách sử dụng .NET Framework, bạn có thể tạo các lớp thời gian thiết kế (design-time classes) giúp mường tượng về các kiểu dữ liệu nhất định. Tuy chủ điểm này vượt quá phạm vi của sách, nhưng bạn có thể thấy nó hoạt động với ba lớp mường tượng cài sẵn dành cho text, HTML, và dữ liệu XML.

Chẳng hạn, hãy tưởng tượng bạn có một biến chuỗi chứa nội dung của một tài liệu XML. Bạn không thể dễ dàng nhìn thấy toàn bộ tài liệu trong phần hiển thị ToolTip chỉ có một dòng. Tuy nhiên, nếu bạn nhấp vào biểu tượng kính lúp kế bên nội dung trong ToolTip, bạn sẽ thấy một danh sách tất cả các trình visualizer (mường tượng hóa) mà bạn có thể sử dụng. Chọn XML Visualizer và một hộp thoại mới sẽ xuất hiện cùng với phần hiển thị đã định dạng, tạo mã màu, có thể cuộn và thay đổi kích cỡ của toàn bộ nội dung tài liệu, như minh họa ở hình 1.6.



Hình 1.6. Xem nội dung XML trong khi gỡ rối

1.4. Chẩn đoán và sửa lỗi

Visual Studio thực hiện tốt chức năng bắt các ngoại lệ, nhưng việc xử lý chúng không phải lúc nào cũng hữu ích. Exception Assistant mới được cài vào Visual Studio 2005 sẽ cho bạn một sự khởi đầu tốt.

Ghi chú

Visual Studio 2005 cho bạn một sự khởi đầu tốt để xử lý các vấn đề thông thường với Exception Assistant của nó.

1.4.1. Bạn làm điều đó như thế nào?

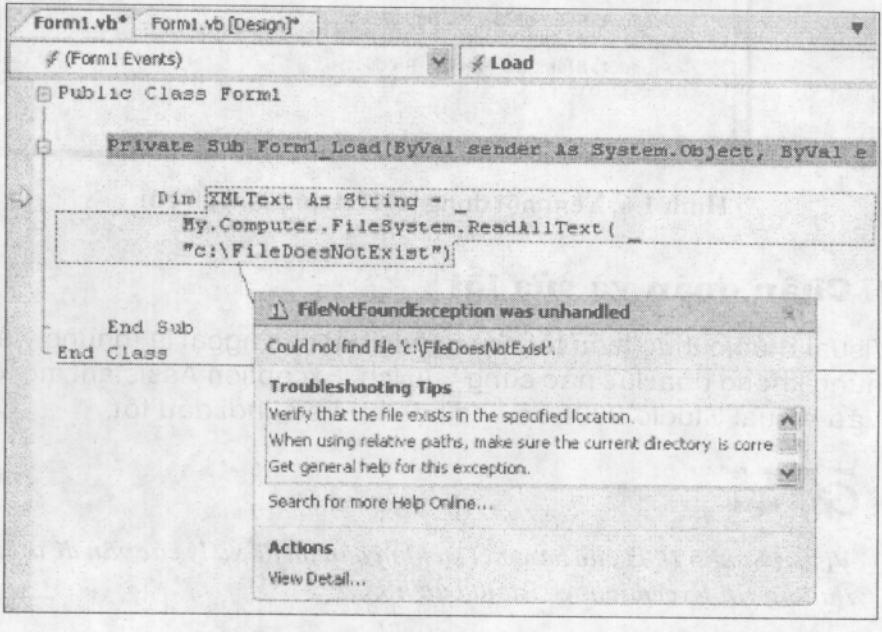
Bạn không cần thực hiện bất kỳ bước nào để kích hoạt Exception Assistant. Thay vào đó, nó sẽ tự hoạt động ngay khi chương trình của bạn bắt gặp một ngoại lệ chưa được xử lý.

Để xem cách nó hoạt động, bạn cần tạo một mã lỗi. Cách thử nghiệm tốt là thêm event handler sau đây vào bất kỳ form, vốn cố mở một file không tồn tại:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load

    Dim XMLText As String = My.Computer.FileSystem.ReadAllText( _
```

```
        "c:\FileDoesNotExist")

    End Sub
```

Bây giờ hãy chạy ứng dụng. Khi lỗi xảy ra, Visual Studio chuyển vào chế độ break (ngắt) và bật sáng câu lệnh bị lỗi. Sau đó Exception Assistant xuất hiện, với một danh sách liệt kê các nguyên nhân có thể gây ra sự cố này. Mỗi đề nghị xuất hiện ở dạng một liên kết riêng biệt trong cửa sổ bật lên (pop-up). Nếu bạn nhấp một trong các liên kết này, toàn bộ chủ điểm trợ giúp MSDN sẽ xuất hiện. Hình 1.7 minh họa kết quả với mã đọc file bị lỗi; Exception Assistant xác định chính xác nguyên nhân là do cố mở file bị lỗi.



Hình 1.7. Nhận sự trợ giúp với một ngoại lệ

Ghi chú

Ví dụ này sử dụng đặc điểm ngôn ngữ VB mới: đối tượng My. Bạn sẽ tìm hiểu thêm về các đối tượng My trong chương kế tiếp.

Nếu bạn muốn xem thông tin ngoại lệ ở cấp độ thấp, hãy nhấp liên kết View Detai ở cuối cửa sổ. Thao tác này sẽ hiển thị một hộp thoại với một PropertyGrid cho thấy mọi thông tin của đối tượng ngoại lệ liên quan. Chỉ riêng sự thay đổi này đã là một bước tuyệt vời tiến triển từ Visual Studio .NET 2003, trong đó bạn cần viết một trình điều hoãn ngoại

lệ Catch và xác lập một điểm ngắt để xem xét đối tượng ngoại lệ nền tảng.

1.4.2. Thế còn...

.. giải quyết các vấn đề phức tạp thì sao? Exception Assistant không được thiết kế để giúp bạn giải quyết các vấn đề phức tạp. Nó chỉ hoạt động tốt nhất khi xác định các lỗi thông thường, như cố sử dụng một phần tham chiếu null (thường là do quên sử dụng từ khóa New) và không thành công khi chuyển đổi một kiểu dữ liệu (thường do sự ép kiểu do vô ý).

1.5. Đổi tên mọi thể hiện của bất kỳ phần tử trong chương trình

Đổi tên tượng trưng (symbolic rename) cho phép bạn đổi tên mọi thể hiện của bất kỳ phần tử mà bạn khai báo trong chương trình chỉ một bước, từ các lớp và giao diện đến các thuộc tính và phương thức. Kỹ thuật này, chắc chắn không phải là một tính năng tìm kiếm và thay thế text đơn giản theo sự nhận biết cú pháp chương trình, giải quyết nhiều vấn đề được tìm thấy trong các ấn bản trước của Visual Studio. Chẳng hạn, hãy tưởng tượng bạn muốn đổi tên của thuộc tính chung FirstName. Nếu bạn sử dụng tính năng tìm kiếm và thay thế, bạn cũng sẽ vô tình làm ảnh hưởng đến một hộp text có tên txtFirstName, một event handler có tên cmdFirstName_Click, một trường cơ sở dữ liệu được truy cập qua row ("FirstName"), và cả các lời bình chú mã của bạn. Với sự thay đổi tên tượng trưng, IDE chỉ chú ý đổi tên của những gì bạn muốn và hoàn thành mọi công việc trong chỉ một bước.

Ghi chú

Nếu bạn cần thay đổi tên một phương thức, thuộc tính hay biến mà không làm ảnh hưởng đến các tên tương tự khác trong cùng file? Visual Studio 2005 có phương pháp hoàn hảo hơn tính năng tìm kiếm và thay thế.

1.5.1. Bạn làm điều đó như thế nào?

Bạn có thể sử dụng phương pháp đổi tên tượng trưng từ bất kỳ cửa sổ mã. Để hiểu cách hoạt động của nó, bạn tạo một form có chứa một hộp text TextBox1 và một nút cmdText. Sau cùng, thêm mã form ở ví dụ 1.2.

Ví dụ 1.2. Một form đơn giản sử dụng từ "Text" ở nhiều nơi

```
Public Class TextTest
```

```
    Private Sub TextTest_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles MyBase.Load
        ' Get the text from the text box.
        Dim Text As String = TextBox1.Text

        ' Convert and display the text.
        Text = ConvertText(Text)
        MessageBox.Show("Uppercase Text is: " & Text)
    End Sub

    Public Function ConvertText(ByVal Text As String) As String
        Return Text.ToUpper( )
    End Function

End Class
```

Mã này thực hiện một tác vụ tương đối tẻ nhạt: chuyển đổi chuỗi do người dùng cung cấp sang chữ hoa và hiển thị nó trong một hộp thông báo. Điều đáng lưu ý là nó sử dụng từ "Text" ở bao nhiêu chỗ. Bây giờ, hãy xem xét điều gì xảy ra nếu bạn cần đổi tên biến cục bộ Text trong event handler dành cho event Form.Load. Rõ ràng, điều này đủ làm bối rối bất kỳ thuật toán tìm kiếm và thay thế. Đó là nơi áp dụng phương pháp đổi tên tượng trưng.

Để sử dụng phương pháp đổi tên tượng trưng (symbolic rename), bạn chỉ cần nhấp phải vào biến Text cục bộ, và chọn Rename từ menu ngữ cảnh. Trong hộp thoại Rename, nhập tên biến mới LocalText và nhấp OK. Mọi trường hợp thể hiện thích hợp trong mã sẽ được thay đổi tự động mà không làm ảnh hưởng đến các phần tử khác trong mã (chẳng hạn như hộp text, lời bình chú, chuỗi text trực diện, tên lớp form, thông số Text trong hàm ConvertText,...). Sau đây là mã được tạo:

```
Public Class TextTest

    Private Sub cmdTest_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles cmdText.Click
        ' Get the text from the text box.
        Dim LocalText As String = TextBox1.Text

        ' Convert and display the text.
        LocalText = ConvertText(LocalText)
```

```
        MessageBox.Show("Uppercase Text is: " & LocalText)
    End Sub

    Public Function ConvertText(ByVal Text As String) As String
        Return Text.ToUpper( )
    End Function

End Class
```

Đổi tên tượng trưng hoạt động với bất kỳ tên thuộc tính, lớp hay phương thức mà bạn muốn thay đổi. Sau đây là một vài điểm quan trọng cần lưu ý về cách đổi tên tượng trưng:

- Nếu bạn đổi tên một lớp (class), tất cả các câu lệnh tạo thể hiện của lớp đó cũng được thay đổi.
- Nếu bạn đổi tên một phương thức (method), tất cả các câu lệnh gọi phương thức đó cũng được thay đổi.
- Nếu bạn đổi tên một biến giống như tên phương thức, chỉ biến được thay đổi (và ngược lại).
- Nếu bạn đổi tên một biến cục giống như tên biến cục bộ có phạm vi khác (chẳng hạn, trong một phương thức khác), chỉ biến đầu tiên là bị ảnh hưởng.

Tính năng đổi tên tượng trưng không gây ấn tượng ngay nhưng nó thật sự hữu ích. Đáng lưu ý là cách nó quan sát chính xác phạm vi của hạng mục mà bạn muốn đổi tên. Chẳng hạn, khi bạn đổi tên một biến cục bộ, các thay đổi của bạn không vượt ra ngoài phạm vi của thủ tục hiện tại. Mặt khác, việc đổi tên một lớp có thể ảnh hưởng đến mọi file trong project của bạn.

Lưu ý rằng nếu bạn thay đổi tên của một biến điều khiển (control variable), mã của bạn cũng được cập nhật theo. Tuy nhiên, có một ngoại lệ là tên của các event handler không bao giờ được sửa đổi tự động. Chẳng hạn, nếu bạn thay đổi Button1 sang Button2, tất cả mã tương tác với Button1 sẽ được cập nhật, nhưng thường trình con event handler Button1_Click sẽ không bị ảnh hưởng. (Ghi nhớ rằng tên của event handler không ảnh hưởng đến cách hoạt động của nó trong ứng dụng của bạn, miễn sao nó được nối với mệnh đề Handles).

Thủ thuật

Trong Visual Studio 2005, khi bạn đổi tên một file .vb trong Solution Explorer, tên của lớp trong file cũng được thay đổi, miễn sao file có chứa một

lớp có tên cũ. Chẳng hạn, nếu bạn đổi tên Form1.vb sang Form2.vb và file có chứa một lớp tên Form1, lớp đó sẽ được đổi tên thành Form2. Bất kỳ câu lệnh mã tạo một thể hiện của Form1 cũng sẽ được cập nhật, cho dù chúng nằm ở đâu trong project. Tuy nhiên, nếu bạn đã thay đổi tên lớp sang một tên khác (như MyForm), tên lớp sẽ không bị ảnh hưởng khi bạn đổi tên file. Trong Visual Studio 2002 và 2003, cùng một hoạt động đổi tên một file form sẽ không ảnh hưởng đến mã của bạn, vì vậy bạn nên lưu ý đến điều này.

1.5.2. Thế còn...

... sự hỗ trợ trong Visual Basic 2005 cho tính năng refactoring C# thì sao? Thật không may, nhiều tính năng refactoring bổ sung mà Visual Studio cung cấp cho các lập trình viên C# không hoàn toàn có trong Visual Basic. Đổi tên tượng trưng (symbolic rename) là một trong số ít các tính năng refactoring mới còn hoạt động tốt đối với các lập trình viên VB trong ấn bản này.

1.6. Sử dụng IntelliSense Filtering và AutoCorrect

IntelliSense là một trong những tiện nghi của Visual Studio và nó liên tục cải tiến trong Visual Studio 2005, với hai tính năng mới làm cho nó trở nên hữu ích hơn: IntelliSense filtering và AutoCorrect. IntelliSense filtering hạn chế số lượng tùy chọn mà bạn nhìn thấy trong những tùy chọn có ý nghĩa của ngữ cảnh hiện tại. AutoCorrect đưa bạn đi xa thêm một bước nữa bằng cách gợi ra các cách xử lý lỗi cú pháp thay vì chỉ báo cáo chúng.

Ghi chú

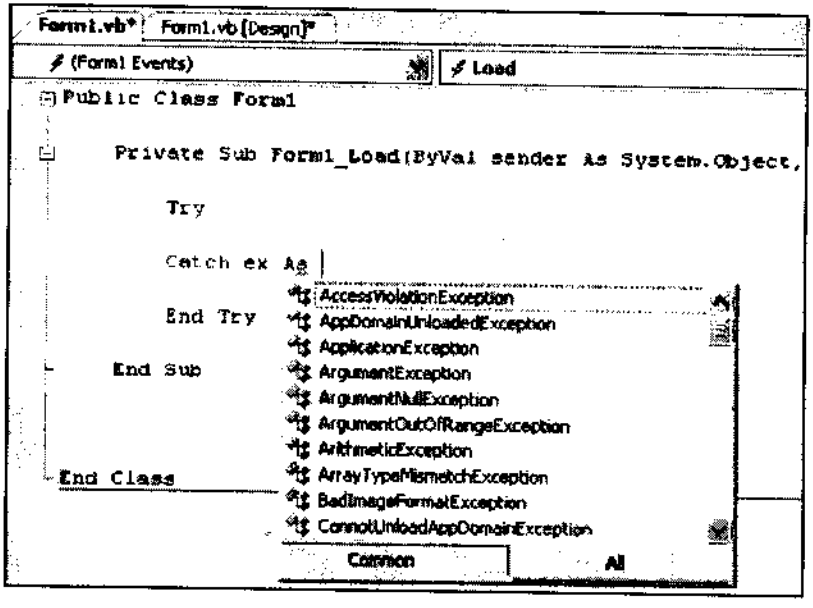
Visual Studio 2005 làm cho IntelliSense trở nên thông minh hơn bằng cách hạn chế các tên lớp không liên quan và gợi ý các phần chỉnh sửa mà bạn có thể áp dụng để sửa các lỗi cú pháp.

1.6.1. Bạn làm điều đó như thế nào?

Không cần thực hiện thêm bất kỳ bước nào khi bạn sử dụng IntelliSense filtering, nó hoạt động tự động. Khi bạn nhập mã, IntelliSense nhắc bạn với các danh sách của các lớp, các thuộc tính, các event và nhiều thứ khác. Trong Visual Studio 2005, danh sách này được sửa đổi theo nhu cầu tức thì của bạn, dựa vào nhiều chi tiết ngữ cảnh. Chẳng hạn, nếu bạn đang chọn một thuộc tính để áp dụng cho một phương thức, danh sách IntelliSense sẽ hiển thị chỉ các lớp phái sinh từ lớp Attribute cơ sở.

Để xem cách hoạt động của IntelliSense mới, bạn bắt đầu gõ một khối xử lý ngoại lệ. Khi bạn nhập khối Catch, danh sách IntelliSense sẽ

hiển thị chỉ các lớp phái sinh từ lớp Exception cơ sở (như minh họa ở hình 1.8). Chọn tab Common or All ở cuối danh sách, tùy thuộc vào việc bạn muốn xem các lớp thường được sử dụng nhiều nhất hay mọi khả năng.



Hình 1.8. Lọc chỉ các lớp Exception

AutoCorrect là một cải tiến IntelliSense nhắm đến các lỗi cú pháp. Mỗi lần Visual Studio phát hiện một sự cố, nó gạch dưới mã bị lỗi bằng màu xanh lục. Bạn có thể tựa con trỏ trên sự cố để xem một ToolTip với thông tin lỗi. Với AutoCorrect, Visual Studio cũng thêm một biểu tượng lỗi màu đỏ mà khi được nhấp, nó sẽ hiển thị một cửa sổ với phần chỉnh sửa đề nghị.

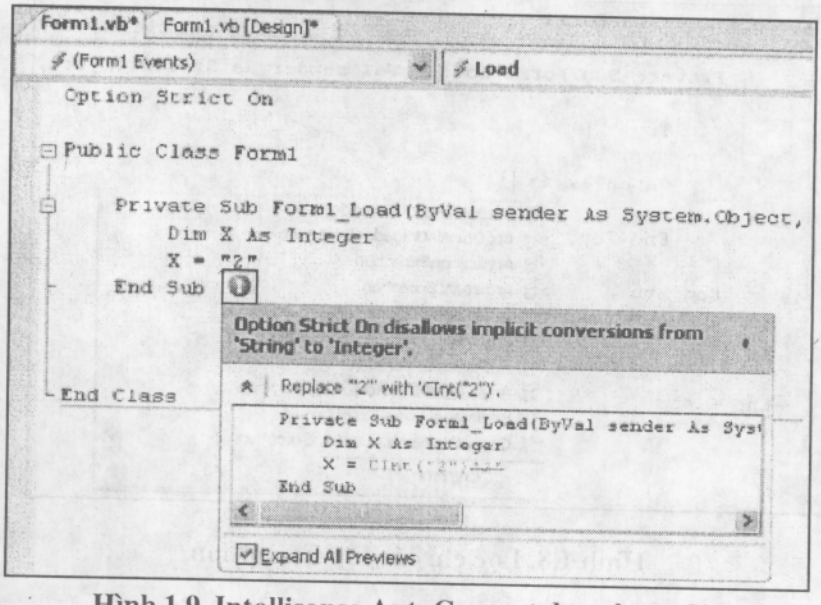
Để xem cách hoạt động của AutoCorrect, bạn nhập mã sau đây (cố gắng gán một chuỗi cho một số nguyên mà không ép kiểu thích hợp cho mã):

```
Dim X As Integer

X = "2"
```

Thủ thuật

Option Strict bắt các loại dữ liệu vào lúc biên dịch. Để mở tính năng này, bạn nhấp đúp vào My Project trong Solution Explorer, nhấp tab Compile, và tìm hộp danh sách xổ xuống Option Strict.

Giả sử Option Strict đã được mở, bạn sẽ nhìn thấy một biểu tượng lỗi màu đỏ khi bạn tựa con trỏ trên dòng này. Nhấp vào biểu tượng lỗi màu đỏ. Cửa sổ AutoCorrect xuất hiện cho bạn thấy mã ở dạng màu xanh dương, mã được thêm vào ở dạng màu đỏ, và mã bị loại bỏ nằm vắt ngang qua đường thẳng nét liền. Hình 1.9 minh họa phần chỉnh sửa đề nghị cho đoạn mã này.



Hình 1.9. Intellisense AutoCorrect đang hoạt động

Các sự cố khác mà AutoCorrect có thể xử lý bao gồm các tên lớp được định tính không đầy đủ, các từ khóa sai chính tả và các dòng bị bỏ sót trong một cấu trúc khối. Trong một số trường hợp, nó còn cho thấy nhiều khả năng chỉnh sửa hơn.

1.6.2. Thế còn...

... những thứ khác thì sao? Vẫn còn nhiều trí thông minh bổ sung mà IntelliSense có thể cung cấp, nhưng không. Chẳng hạn, khi gán một thuộc tính từ một lớp cho một biến chuỗi, tại sao không hiển thị chỉ những thuộc tính trả về các kiểu dữ liệu chuỗi? Hoặc khi áp dụng một thuộc tính cho một phương thức, tại sao không hiển thị các lớp thuộc tính được áp dụng cho chỉ các phương thức? Khi các bộ xử lý máy tính trở nên nhanh hơn và có nhiều chu trình nghỉ hơn, hãy mong đợi các cấp độ mới của trí thông minh nhân tạo xuất hiện trong IDE của bạn.

1.7. Hiệu chỉnh các thuộc tính Control tại chỗ

Cửa sổ Properties trong Visual Studio giúp hiệu chỉnh control dễ dàng hơn, nhưng không phải lúc nào cũng nhanh chóng. Chẳng hạn, hãy

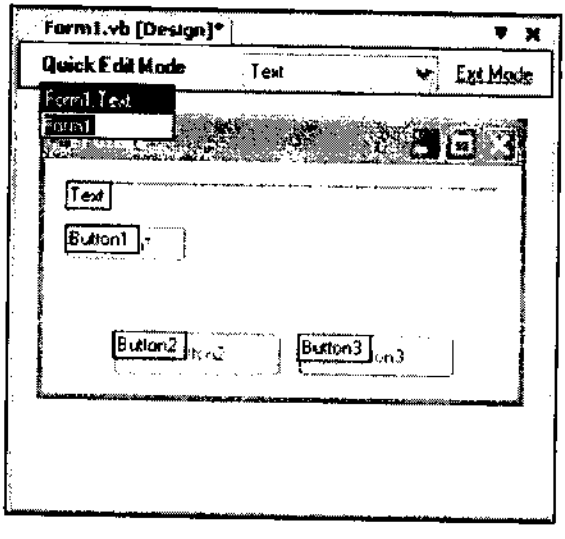
tưởng tượng bạn muốn điều chỉnh toàn bộ text trên một form. Trong các phiên bản trước của Visual Studio, tùy chọn duy nhất là chọn lần lượt mỗi control và chỉnh sửa thuộc tính Text trong cửa sổ Properties theo từng cái một. Mặc dù phương pháp này không nhất thiết khó thực hiện, nhưng chắc chắn nó không dễ như trước đây. Trong Visual Studio 2005, bạn có thể điều chỉnh một thuộc tính đơn cho một chuỗi control ngay trên form.

Ghi chú

Khi bạn cần cập nhật một thuộc tính cho nhiều control khác nhau, phương pháp hiệu chỉnh thuộc tính tại chỗ sẽ giúp bạn dễ dàng thực hiện điều này.

1.7.1. Bạn làm điều đó như thế nào?

Để thử phương pháp hiệu chỉnh thuộc tính tại chỗ, bạn tạo một form mới và thêm một tập hợp control. (Các control thật sự mà bạn sử dụng không quan trọng lắm, nhưng bạn nên đưa vào một số hộp text, các nút và các nhãn). Sau đó, chọn View > Property Editing View. Sau cùng, chọn thuộc tính bạn muốn thay đổi từ danh sách xổ xuống phía trên bề mặt thiết kế form. Theo mặc định, thuộc tính Name được chọn, nhưng hình 1.10 minh họa một ví dụ với thuộc tính Text.



Hình 1.10. Hiệu chỉnh một thuộc tính cho nhiều control

Trong khung xem hiệu chỉnh thuộc tính, một hộp hiệu chỉnh xuất hiện qua mỗi control trên form với các nội dung của thuộc tính được chọn. Bạn có thể hiệu chỉnh trị của thuộc tính đó bằng cách nhấp vào hộp hiệu

chỉnh và nhập trị mới. Bạn cũng có thể nhảy từ control này sang control kế tiếp bằng cách nhấn phím Tab.

Khi bạn đã hoàn thành công việc của mình, một lần nữa hãy chọn View > Property Editing View, hoặc nhấp liên kết Exit Mode kế bên danh sách xổ xuống thuộc tính.

1.7.2. Hiệu chỉnh thứ tự tab

Visual Studio cho phép bạn dễ dàng hiệu chỉnh thứ tự tab bằng cách nhấp vào các control theo thứ tự bạn muốn người dùng có thể định hướng qua chúng. Chọn một form với ít nhất một control và chọn View > Tab Order để kích hoạt chế độ này, vốn hoạt động tương tự như trong Visual Studio 2003.

1.8. Gọi các phương thức lúc thiết kế

Mặc dù Visual Studio .NET 2003 có cửa sổ Immediate, nhưng bạn không thể sử dụng nó để thực thi mã vào lúc thiết kế. Trong một thời gian dài, những người tạo mã VB đã bỏ qua tính năng này. Trong Visual Studio 2005, tính năng này trở lại cùng với sự trở lại của một trình biên dịch nền.

Ghi chú

Bạn cần thử một thường trình mã được viết hoàn toàn mới? Visual Studio 2005 cho phép bạn chạy nó mà không cần khởi động project của bạn.

1.8.1. Bạn làm điều đó như thế nào?

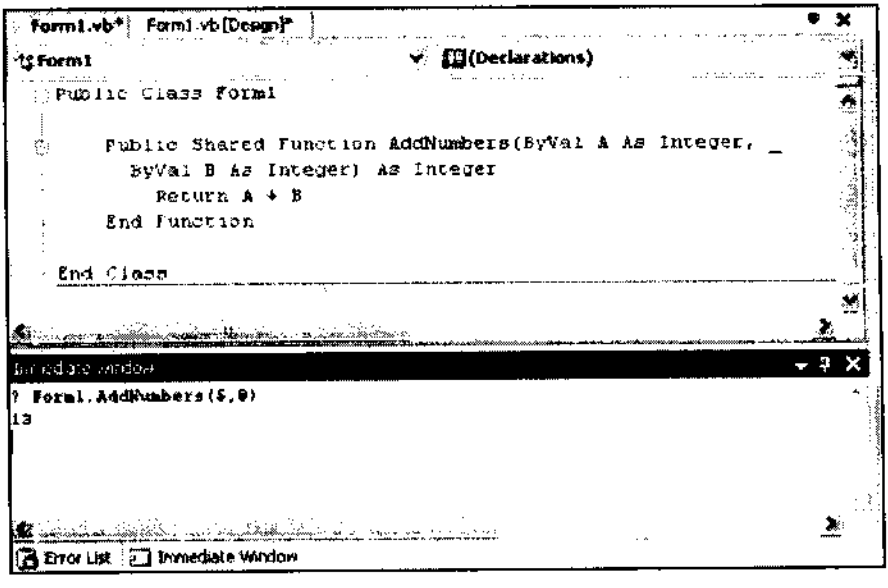
Bạn có thể sử dụng cửa sổ Immediate để lượng giá các biểu thức đơn giản và thậm chí chạy các thường trình con trong mã. Để thử kỹ thuật này, hãy thêm phương thức chia sẻ sau đây vào một lớp:

```
Public Shared Function AddNumbers(ByVal A As Integer, _
  ByVal B As Integer) As Integer

    Return A + B

End Sub
```

Bằng cách làm cho kỹ thuật này trở thành một phương thức chia sẻ, bạn chắc chắn nó có sẵn ngay cả khi không tạo một trường hợp thể hiện

của lớp. Bây giờ, bạn có thể gọi nó một cách dễ dàng trong môi trường thiết kế.

Theo mặc định, cửa sổ Immediate không hiển thị vào lúc thiết kế. Để hiển thị nó, bạn chọn Debug > Windows > Command từ menu. Các câu lệnh bên trong cửa sổ Immediate thường bắt đầu với ? (một dạng tắt của Print, yêu cầu Visual Studio hiển thị kết quả). Bạn có thể nhập phần còn lại của câu lệnh như bất kỳ dòng mã khác, với sự tiện ích của IntelliSense. Hình 1.11 minh họa một ví dụ mà trong đó cửa sổ Command được sử dụng để chạy phương thức chia sẻ đã trình bày ở trên.



Hình 1.11. Thực thi mã vào lúc thiết kế

Khi bạn thực thi một câu lệnh như minh họa ở hình 1.11, sẽ có một sự tạm ngưng ngắn trong khi trình biên dịch nền (background compiler) hoạt động (và bạn sẽ thấy thông báo "Build started" trong thanh trạng thái). Sau đó kết quả sẽ xuất hiện.

Ghi chú

Sự lượng giá biểu thức trong phiên bản beta của Visual Studio 2005 hơi kỳ quặc. Một số phần lượng giá không xử lý và chỉ khởi động trình ứng dụng của bạn mà không cho ra kết quả nào. Hãy tìm điểm này để cải tiến trong các phần tạo sau này.

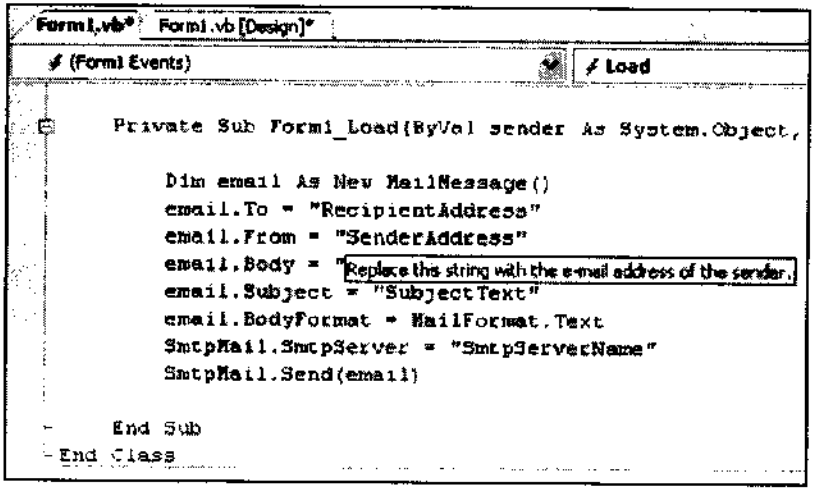
1.9. Chèn mã nhờ dùng Snippet

Một số mã thì phổ biến và chung đủ để các lập trình viên ở bất kỳ nơi nào viết lại nó vào mỗi ngày. Mặc dù các nhà phát triển có sự trợ giúp của tài liệu trực tuyến, các mẫu và các sách tham khảo như cuốn sách này, mã hữu dụng chưa bao giờ có vẻ có sẵn khi bạn cần nó. Visual Studio 2005 có một tính năng code snippet mới cho phép bạn chèn mã thường được sử dụng và nhanh chóng điều chỉnh nó cho phù hợp với các mục đích của bạn. Các phần tạo beta đầu tiên của Visual Studio 2005 đã đưa vào một công cụ để tạo các snippet (đoạn mã) riêng của bạn. Tuy tính năng này chưa có trong các ấn bản mới nhất nhưng Microsoft đã gợi ý rằng nó có thể xuất hiện ở dạng công cụ add-on riêng biệt trong một thời điểm sau này.

1.9.1. Bạn làm điều đó như thế nào?

Bạn có thể chèn một đoạn mã vào bất kỳ nơi nào trong mã của bạn. Chỉ cần chuyển đến vị trí thích hợp, nhấp phải chuột trên dòng hiện hành và chọn Insert Snippet. Một menu bật lên với một danh sách các hạng mục snippet, chẳng hạn như Math, Connectivity and Networking, và Working with XML. Ngay sau khi bạn chọn một hạng mục, một menu sẽ xuất hiện cùng với một danh sách các snippet. Ngay khi bạn chọn một snippet, mã sẽ được chèn vào.

Chẳng hạn, giả sử bạn muốn thêm khả năng gởi và nhận các thông báo email vào ứng dụng của bạn. Bạn chỉ cần tạo một event handler mới hay một phương thức độc lập và nhấp phải bên trong nó. Sau đó, chọn Insert Snippet và chọn Connectivity and Networking Create an Email Message. Hình 1.12 minh họa mã được chèn vào.



Hình 1.12. Tùy biến một đoạn mã

Các phần được tạo bóng của mã là các trị trực kiện (như các đường dẫn file và các phần tham chiếu control) mà bạn cần tạo tùy biến để điều chỉnh mã phù hợp với nhu cầu của bạn. Bằng cách nhấn phím Tab, bạn có thể chuyển từ miền được tạo bóng sang miền kế tiếp. Ngoài ra, nếu bạn tựa con trỏ chuột trên một miền được tạo bóng, một ToolTip sẽ xuất hiện với phần mô tả về những nội dung nào bạn cần chèn.

1.9.2. Thế còn...

... việc lấy thêm các snippet thì sao? Các đoạn mã cơ bản đi kèm với Visual Studio .NET thì khá khiêm tốn. Nó chỉ bao gồm các snippet thật sự hữu ích (ví dụ, "Find a Node in XML Data") và một số snippet thử (ví dụ, "Add a Comment to Your Code"). Tuy nhiên, nhiều chủ điểm chẳng hạn như sự mã hóa không hoàn toàn liên quan đến điều này.

1.10. Tạo tài liệu XML cho mã

Việc bình chú và tạo tài liệu chính xác cho mã đòi hỏi phải có thời gian. Thật không may, không có cách dễ dàng nào để tăng cường các bình chú mà bạn đặt vào trong mã khi đến lúc tạo tài liệu và các phần tham chiếu API chi tiết hơn. Thay vào đó, bạn phải tạo các tài liệu này ngay từ đầu.

Ghi chú

Sử dụng các lời bình chú XML để dễ dàng tạo các tham chiếu mã chi tiết, một tính năng mà các lập trình viên C# đã có kể từ .NET 1.0.

Visual Studio 2005 thay đổi tất cả những điều này bằng cách đưa ra một tính năng đã được chấp nhận bởi các lập trình viên C# kể từ các bình chú .NET 1.0XML. Với các bình chú XML, bạn giải thích mã của mình nhờ dùng một dạng đã được ấn định sẵn. Sau đó, bạn có thể sử dụng các công cụ khác để tương xứng các bình chú này và sử dụng chúng để tạo các tài liệu khác. Các tài liệu này có thể bao gồm tài liệu trợ giúp đến các báo cáo mã chuyên biệt (ví dụ, một danh sách các vấn đề chưa được giải quyết, mã kế thừa hay ngày tháng xem lại mã).

1.10.1. Bạn làm điều đó như thế nào?

Các bình chú XML được phân biệt với các bình chú thông thường qua dạng của chúng. Trước tiên, các bình chú XML bắt đầu với ba dấu apostrophe (') (thay vì chỉ một). Sau đây là một ví dụ:

```
''' <summary>This is the summary.</sumary>
```

Như bạn có thể thấy, các bình chú XML cũng có một đặc điểm

khác, đó là chúng sử dụng các tên thẻ gán (tag). Thẻ gán cho biết loại bình chú. Các thẻ gán này giúp bạn phân biệt giữa thông tin tóm tắt, thông tin về một phương thức cụ thể, các phần tham chiếu đến các mục tài liệu khác,...

Các thẻ gán bình chú XML thường được sử dụng nhất là:

<summary>

Mô tả một lớp hay một loại khác. Đây là thông tin ở cấp độ cao nhất cho mã của bạn.

<remarks>

Cho phép bạn bổ sung thông tin tóm tắt. Thẻ gán này thường được sử dụng nhất để cung cấp một mô tả ở cấp độ cao của mỗi thành viên kiểu (ví dụ, các phương thức và các thuộc tính riêng lẻ).

<param>

Mô tả các thông số được chấp nhận bởi một phương thức. Thêm một thẻ gán <param> cho mỗi thông số.

<returns>

Mô tả trị trả về của một phương thức.

<exception>

Cho phép bạn chỉ định một lớp có thể tung ra những ngoại lệ nào.

<example>

Cho phép bạn chỉ định một ví dụ về cách sử dụng một phương thức hay thành viên khác.

<see>

Cho phép bạn tạo một liên kết với một thành phần tài liệu khác.

Ngoài ra, có các thẻ gán thường được sử dụng chỉ để định dạng hay tạo cấu trúc cho các khối text. Bạn sử dụng các thẻ gán này bên trong các thẻ gán khác. Chúng bao gồm:

<para>

Cho phép bạn thêm cấu trúc vào một thẻ gán (chẳng hạn như thẻ gán <remarks>) bằng cách tách nội dung của nó ra thành các đoạn.

<list>

Bắt đầu một danh sách đánh dấu bullet. Bạn phải tạo mỗi mục danh sách riêng lẻ bằng thẻ gán <item>.

<c>

Biểu thị text bên trong một phần mô tả sẽ được đánh dấu là mã. Sử dụng thẻ gán <code> để biểu thị nhiều dòng ở dạng mã.

<code>

Cho phép bạn nhúng nhiều dòng của mã, như trong ví dụ về cách sử dụng. Ví dụ, bạn thường đặt một thẻ gán <code> bên trong một thẻ gán <example>.

Ngoài ra, bạn có thể định nghĩa các thẻ gán tùy ý mà sau đó bạn có thể sử dụng cho các mục đích riêng của bạn.

Visual Studio giúp bạn điều này bằng việc tự động thêm các thẻ gán XML nhưng chỉ khi nào bạn muốn chúng. Chẳng hạn, hãy xem xét thường trình mã được minh họa ở đây, thường trình này kiểm tra hai file có hoàn toàn giống nhau hay không nhờ dùng một mã hash (băng). Để sử dụng mẫu này như được viết trong ví dụ, bạn cần nhập các namespace System.IO và System.Security.Cryptography:

Bây giờ, đặt con trỏ ngay trước phần khai báo hàm và chèn ba dấu apostrophe ('). Visual Studio sẽ tự động thêm một tập hợp phác thảo của các thẻ gán XML, như minh họa dưới đây:

```
Public Function TestIfTwoFilesMatch(ByVal fileA As String, _
  ByVal fileB As String) As Boolean

    ' Create the hashing object.
    Dim Hash As HashAlgorithm = HashAlgorithm.Create( )

    ' Calculate the hash for the first file.
    Dim fsA As New FileStream(fileA, FileMode.Open)
    Dim HashA( ) As Byte = Hash.ComputeHash(fsA)
    fsA.Close( )

    ' Calculate the hash for the second file.
    Dim fsB As New FileStream(fileB, FileMode.Open)
    Dim HashB( ) As Byte = Hash.ComputeHash(fsB)
    fsB.Close( )
```

```
    ' Compare the hashes.
    Return (Convert.ToString(HashA) = Convert.ToString(HashB))

End Function
```

Bây giờ, bạn chỉ cần điền vào một số nội dung mẫu trong các thẻ gán:

```
''' <summary>
'''
''' </summary>
''' <param name="fileA"></param>
''' <param name="fileB"></param>
''' <returns></returns>
''' <remarks></remarks>
Public Function TestIfTwoFilesMatch(ByVal fileA As String, _
  ByVal fileB As String) As Boolean

...
```

Để có được một ví dụ hiện thực hơn, hãy đặt phương thức TestIfTwoFilesMatch ( ) vào một lớp và thêm các thẻ gán tài liệu XML vào phần khai báo lớn (class). Sau đây là một ví dụ tiêu biểu, ví dụ này sử dụng các phần tham chiếu chéo chỉ đến các phương thức có sẵn trong một danh sách:

```
''' <summary>
''' This function tests whether two files
''' contain the exact same content.
''' </summary>
''' <param name="fileA">Contains the full path to the first file.</param>
''' <param name="fileB">Contains the full path to the second file.</param>
''' <returns>True if the files match, false if they don't.</returns>
''' <remarks>
''' The implementation of this method uses cryptographic classes
''' to compute a hash value. This may not be the most performant
''' approach, but it is sensitive to the minutest differences,
''' and can't be practically fooled.
''' </remarks>
```

Không giống như các bình chú khác, các bình chú XML được thêm vào metadata của hợp ngữ được biên dịch. Chúng sẽ tự động xuất hiện trong Object Browser khi bạn xem xét kiểu.

Ngoài ra, ngay sau khi bạn tạo các thẻ gán XML, bạn có thể xuất chúng sang một tài liệu XML. Chỉ cần nhấp double vào nút My Project trong Solution Explorer, chọn tab Compile và bảo đảm tùy chọn "Generate XML documentation file" được chọn. Tài liệu XML được lưu tự động vào một file XML với tên giống như tên project và phần mở rộng .xml (trong thư mục bin).

Tài liệu được tạo sẽ bao gồm mọi bình chú XML, không có mã nào. Sau đó, bạn có thể nạp tài liệu này vào một loại ứng dụng khác. Chẳng hạn, bạn tạo ứng dụng riêng của bạn để quét các file bình chú XML và tạo các báo cáo chuyên biệt.

1.10. 2. Thế còn...

... việc tạo tài liệu trợ giúp thì sao? Mặc dù Visual Studio 2005 không có bất kỳ công cụ nào của riêng nó, nhưng ứng dụng NDoc nguồn mở có cung cấp một giải quyết (http://ndoc.sourceforge.net). NDoc lướt qua mã và sử dụng các thẻ gán tài liệu XML mà nó tìm thấy để tạo các trang web kiểu MSDN hay tài liệu kiểu Visual Studio (MS Help 2.0). Hiện tại NDoc chưa hỗ trợ .NET 2.0.

Chương 2

Ngôn ngữ Visual Basic 2005

Khi Visual Basic .NET lần đầu tiên xuất hiện, các nhà phát triển VB trung thành đã bị sốc khi phát hiện có nhiều sự thay đổi trong ngôn ngữ mà họ ưa chuộng. Đột nhiên, các tác vụ thông thường như thể hiện một đối tượng và khai báo một cấu trúc lại đòi hỏi cú pháp mới, và cả những kiểu dữ liệu cơ bản như mảng (array) được biến đổi thành một điều gì đó mới. Thật may, Visual Basic 2005 không bị trường hợp này. Sự thay đổi ngôn ngữ trong phiên bản mới nhất của VB là những phần tin chỉnh giúp đơn giản các tác vụ mà họ không làm cho bất kỳ mã hiện có nào bị lỗi thời. Nhiều thay đổi là những đặc điểm ngôn ngữ được nhập vào từ C# (ví dụ, quá tải toán tử), trong khi các thay đổi khác là các thành phần hoàn toàn mới được cài vào phiên bản mới nhất của thường chạy ngôn ngữ phổ biến. Trong chương này, bạn sẽ học về mọi sự thay đổi hữu ích nhất đối với ngôn ngữ VB.

2.1. Sử dụng các đối tượng My để lập trình các tác vụ thông thường

Các đối tượng My mới cung cấp sự truy cập dễ dàng vào nhiều tính năng mà các nhà phát triển thường cần nhưng không nhất thiết phải biết nơi tìm trong thư viện lớp .NET. Về cơ bản, các đối tượng My cung cấp one-stop shopping (mua sắm một chặn), với sự truy cập vào mọi thứ từ Windows registry đến nối kết mạng hiện hành. Trên hết, hệ thống phân cấp đối tượng My được sắp xếp theo công dụng và dễ định hướng nhờ dùng Visual Studio IntelliSense.

2.1.1. Bạn làm điều đó như thế nào?

Có bảy đối tượng My ở cấp độ đầu tiên. Trong số đó, ba đối tượng cốt lõi tập trung hóa tính chức năng từ .NET Framework và cung cấp thông tin máy tính. Ba đối tượng này bao gồm:

Ghi chú

Bạn thấy mệt mỏi khi phải lướt qua thư viện lớp .NET bao quát để tìm những gì bạn cần? Với các đối tượng My mới, bạn có thể nhanh chóng tìm thấy một số tính năng hữu dụng nhất mà .NET cung cấp.

My.Computer

Đối tượng này cung cấp thông tin về máy tính hiện hành, bao gồm nối kết mạng, trạng thái chuột và bàn phím, máy in và màn hình và đồng hồ. Bạn cũng có thể sử dụng đối tượng này làm điểm nhảy để mở âm thanh, tìm một file, truy cập registry, hay sử dụng Windows clipboard.

My.Application

Đối tượng này cung cấp thông tin về ứng dụng hiện hành và ngữ cảnh của nó, bao gồm hợp ngữ và phiên bản của nó, folder mà trình ứng dụng đang chạy, văn hóa và các đối số dòng lệnh đã được sử dụng để khởi động trình ứng dụng. Bạn cũng có thể sử dụng đối tượng này để ghi lại một sự kiện ứng dụng.

My.User

Đối tượng này cung cấp thông tin về người dùng hiện hành. Bạn có thể sử dụng đối tượng này để kiểm tra account Windows của người dùng và kiểm tra người dùng là thành viên của nhóm nào.

Ngoài ba đối tượng trên, còn có hai đối tượng khác cung cấp các phần thể hiện mặc định. Các phần thể hiện mặc định là các đối tượng mà .NET tự động tạo ra cho các loại lớp nhất định được xác định trong ứng dụng của bạn. Chúng bao gồm:

My.Forms

Đối tượng này cung cấp một dạng thể hiện mặc định của mỗi form Windows trong ứng dụng của bạn. Bạn có thể sử dụng đối tượng này để giao tiếp giữa các form mà không cần theo dõi các tham chiếu form trong một lớp khác.

My.WebServices

Đối tượng này cung cấp một dạng thể hiện proxy-class mặc định cho mọi dịch vụ web. Chẳng hạn, nếu project của bạn sử dụng hai phần tham chiếu web, bạn có thể truy cập một lớp proxy đã được tạo sẵn cho mỗi phần tham chiếu qua đối tượng này.

Hai đối tượng My sau cùng giúp dễ dàng truy cập vào các xác lập cấu hình và các nguồn tài nguyên:

My.Settings

Đối tượng này cho phép bạn truy xuất các xác lập tùy ý từ file cấu hình XML của ứng dụng.

My.Resources

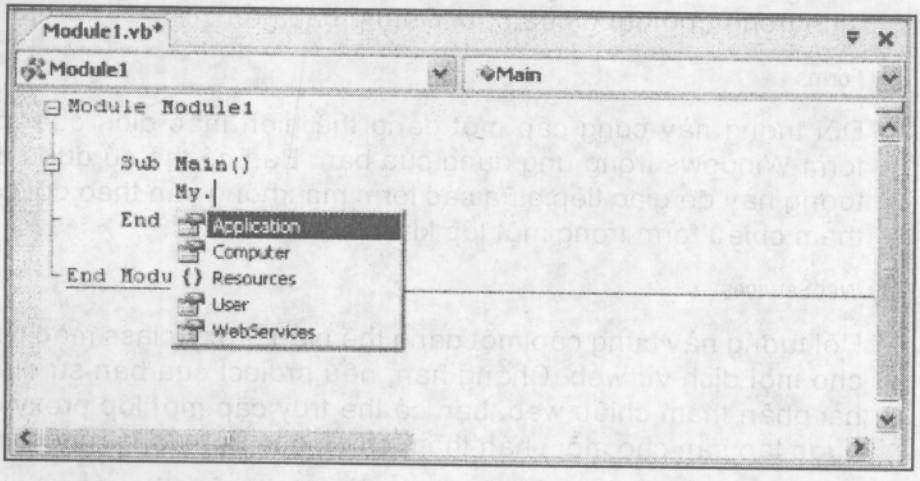
Đối tượng này cho phép bạn truy xuất các khối tài nguyên nhị phân hay dữ liệu text được biên dịch thành hợp ngữ ứng dụng của bạn. Các nguồn tài nguyên thường được sử dụng để chứa các chuỗi cục bộ hóa, các hình ảnh và các file audio.

Lưu ý

Lưu ý rằng các đối tượng My chịu ảnh hưởng bởi loại project. Chẳng hạn, khi tạo một ứng dụng web hay console (bàn điều khiển), bạn sẽ không thể sử dụng My.Forms.

Một số lớp My được định nghĩa trong namespace Microsoft.VisualBasic.MyServices, trong khi các lớp khác, chẳng hạn như các lớp được sử dụng cho các đối tượng My.Settings và My.Resources, được tạo động bởi Visual Studio 2005 khi bạn chỉnh sửa các xác lập ứng dụng và thêm tài nguyên vào project hiện hành.

Để thử đối tượng My, bạn có thể sử dụng Visual Studio IntelliSense. Chỉ cần gõ My, theo sau là một dấu chấm và xem xét các đối tượng có sẵn, như minh họa ở hình 2.1. Bạn có thể chọn một đối tượng và nhấn lại dấu chấm đ chuyển xuống một cấp độ khác.



Hình 2.1. Trình duyệt các đối tượng My

Để thử một ví dụ đơn giản hiển thị một số thông tin cơ bản nhờ dùng đối tượng My, chúng ta tạo một project console mới. Sau đó thêm mã này vào thường trình Main ( ):

```
Console.WriteLine(My.Computer.Name)
Console.WriteLine(My.Computer.Clock.LocalTime)
```

```
Console.WriteLine(My.Application.CurrentDirectory)

Console.WriteLine(My.User.Identity.Name)
```

Khi bạn chạy mã này, bạn sẽ thấy một số kết quả xuất trong cửa sổ console cho biết tên máy tính, thời gian hiện hành, thư mục ứng dụng và người dùng:

```
SALESSERVER

2005-10-1 8:08:52 PM

C:\Code\VBNotebook\1.07\MyTest\bin

MATTHEW
```

Đối tượng My cũng có một "phía tối". Việc sử dụng đối tượng My làm bạn khó chia sẻ giải pháp của mình với các nhà phát triển không sử dụng VB, bởi vì các ngôn ngữ khác, chẳng hạn như C#, không có cùng tính năng này.

2.1.2. Tìm hiểu thêm?

Bạn có thể tìm hiểu về đối tượng My và xem các ví dụ bằng cách tra mục index "My Object" trong MSDN Help. Bạn cũng có thể tìm hiểu thêm bằng việc xem qua một số mục khác trong cuốn sách này có đề cập đến đối tượng My. Một số ví dụ bao gồm:

- Sử dụng My.Application để truy xuất các chi tiết của chương trình, chẳng hạn như phiên bản hiện hành và các thông số dòng lệnh được sử dụng để khởi động nó (xem mục "Lấy thông tin ứng dụng" trong chương này).
- Sử dụng My.Resources để tải các hình ảnh và các tài nguyên khác từ hợp ngữ ứng dụng (xem mục "Sử dụng các tài nguyên được tạo kiểu mạnh" trong chương này).
- Sử dụng My.Settings để truy xuất các xác lập ứng dụng và người dùng (xem mục "Sử dụng các xác lập cấu hình được tạo kiểu mạnh" trong chương này).
- Sử dụng My.Forms để tương tác giữa các cửa sổ ứng dụng (xem mục "Giao tiếp giữa các Form" trong chương 3).
- Sử dụng My.Computer để thực hiện sự thao tác file và các tác vụ mạng trong các chương 5 và 6.

- Sử dụng My.User để xác thực người dùng hiện hành (xem mục "Kiểm tra tư cách thành viên trong nhóm của người dùng hiện hành" ở chương 6).

2.2. Lấy thông tin ứng dụng

Đối tượng My.Application cung cấp sẵn một kho tàng thông tin. Việc lấy thông tin này dễ như truy xuất một thuộc tính.

Ghi chú

Nhờ sử dụng đối tượng My.Application, bạn có thể lấy thông tin về phiên bản hiện hành của ứng dụng, nó nằm ở nơi nào và các thông số nào đã được sử dụng để khởi động nó.

2.2.1. Bạn làm điều đó như thế nào?

Thông tin trong đối tượng My.Application trở nên tiện lợi trong nhiều tình huống. Sau đây là hai ví dụ:

- Bạn muốn lấy số phiên bản chính xác. Điều này có thể hữu dụng nếu bạn muốn tạo một hộp About động hoặc kiểm tra với một dịch vụ web để bảo đảm bạn có phiên bản mới nhất của một hợp ngữ.
- Bạn muốn ghi một số chi tiết chẩn đoán. Điều này trở nên quan trọng nếu một sự cố xảy ra tại một site client và bạn cần ghi một số thông tin chung về ứng dụng đang chạy.

Để tạo một ví dụ đơn giản, bạn có thể sử dụng mã ở ví dụ 2.1 trong một ứng dụng console. Nó truy xuất tất cả các chi tiết này và hiển thị một báo cáo hoàn chỉnh trong cửa sổ console.

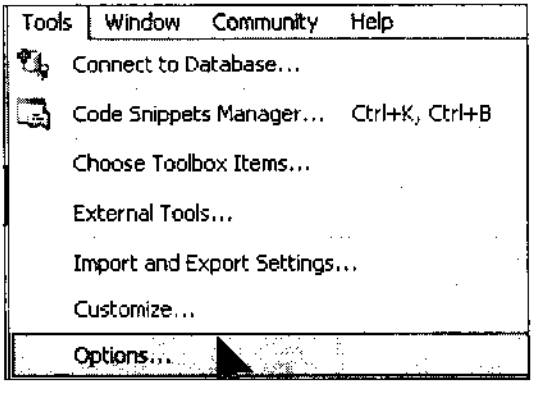
Ví dụ 2.1. Truy xuất thông tin từ My.Application

```
' ' Find out what parameters were used to start the application.
Console.Write("Command line parameters: ")
For Each Arg As String In My.Application.CommandLineArgs
    Console.Write(Arg & " ")
Next
Console.WriteLine( )
Console.WriteLine( )

' Find out some information about the assembly where this code is located.
' This information comes from metadata (attributes in your code).
```

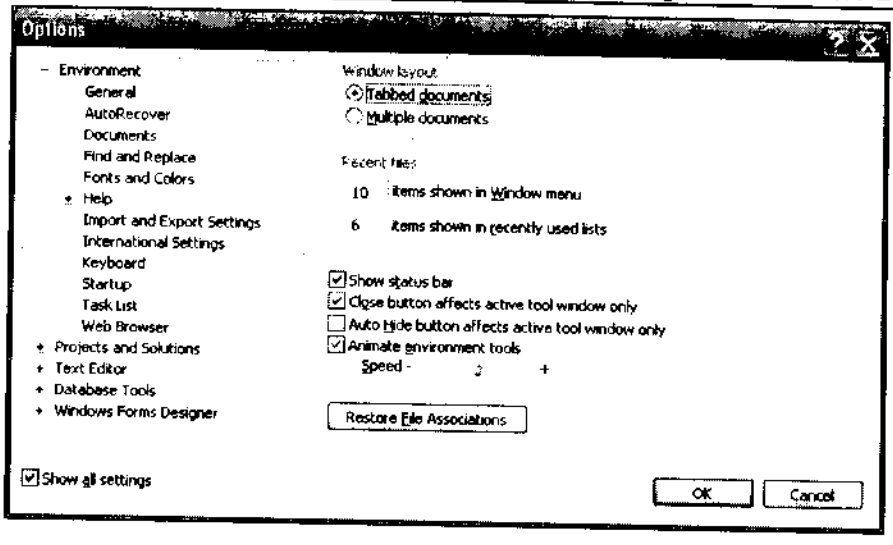
```
Console.WriteLine("Company: " & My.Application.Info.CompanyName)
Console.WriteLine("Description: " & My.Application.Info.Description)
Console.WriteLine("Located in: " & My.Application.Info.DirectoryPath)
Console.WriteLine("Copyright: " & My.Application.Info.Copyright)
Console.WriteLine("Trademark: " & My.Application.Info.Trademark)
Console.WriteLine("Name: " & My.Application.Info.AssemblyName)
Console.WriteLine("Product: " & My.Application.Info.ProductName)
Console.WriteLine("Title: " & My.Application.Info.Title)
Console.WriteLine("Version: " & My.Application.Info.Version.ToString( ))
Console.WriteLine( )
```

Thủ thuật

Visual Studio 2005 có một cửa sổ Quick Console hoạt động như là một phiên bản nhẹ của cửa sổ dòng lệnh bình thường. Trong một số trường hợp, cửa sổ này bị một lỗi nhỏ. Nếu bạn gặp rắc rối khi chạy một ứng dụng console mẫu và thấy kết xuất của nó, bạn chỉ cần tắt tính năng này. Để làm như vậy, bạn chọn Tools > Options.



Bảo đảm hộp kiểm "Show all settings" được chọn, và chọn tab Debugging General. Sau đó tắt "Redirect all console output to the Quich Console window".



Trước khi bạn kiểm tra mã này, nhớ cài đặt môi trường để bảo đảm bạn sẽ nhìn thấy dữ liệu có ý nghĩa. Chẳng hạn, bạn có thể muốn yêu cầu Visual Studio cung cấp một số thông số dòng lệnh khi nó khởi động ứng dụng. Để làm điều này, bạn nhấp double vào biểu tượng My Project trong Solution Explorer. Sau đó, chọn tab Debug và tìm hộp text "Command line parameters". Chẳng hạn, bạn có thể thêm ba thông số bằng cách chỉ định dòng lệnh /a /b /c.

Nếu bạn muốn xác lập thông tin như tác giả hợp ngữ, sản phẩm, phiên bản,... bạn cần thêm các thuộc tính đặc biệt vào file AssemblyInfo.vb, file này không hiển thị trong Solution Explorer. Để truy cập nó, bạn cần chọn Solution > Show All Files. Bạn sẽ tìm thấy file AssemblyInfo.vb bên dưới nút My Project. Sau đây là một bộ thẻ gán tiêu biểu mà bạn có thể nhập vào:

```
<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyCompany("Prosetech")>
<Assembly: AssemblyDescription("Utility that tests My.Application")>
<Assembly: AssemblyCopyright("(C) Matthew MacDonald")>
<Assembly: AssemblyTrademark("(R) Prosetech")>
<Assembly: AssemblyTitle("Test App")>
<Assembly: AssemblyProduct("Test App")>
```

Tất cả những thông tin này được nhúng trong hợp ngữ đã biên dịch của bạn ở dạng metadata.

Bây giờ bạn có thể chạy ứng dụng kiểm tra. Sau đây là một ví dụ về kết xuất mà bạn sẽ thấy:

Ghi chú

Điểm mới trong VB 2005 là khả năng thêm thông tin ứng dụng trong một hộp thoại đặc biệt. Để sử dụng tính năng này, bạn nhấp double vào mục My Project trong Solution Explorer, chọn tab Assembly và nhấp nút Assembly Information.

```
Command line parameters: /a /b /c

Company: Prosetech
Description: Utility that tests My.Application
Located in: C:\Code\VBNotebook\1.08\ApplicationInfo\bin
Copyright: (C) Matthew MacDonald
Trademark: (R) Prosetech
Name: ApplicationInfo.exe
Product: Test App
Title: Test App
Version: 1.0.0.0
```

2.2.2. Lấy thông tin chẩn đoán chi tiết hơn

Đối tượng My.Computer.Info cũng cung cấp các chi tiết chẩn đoán với hai thuộc tính hữu ích. LoadedAssemblies cung cấp một tập hợp với mọi hợp ngữ đang được tải (và có sẵn đối với ứng dụng của bạn). Bạn cũng có thể kiểm tra thông tin phiên bản và nhà xuất bản. StackTrace cung cấp một snapshot của stack (ngăn xếp) hiện hành, phản ánh bạn đang ở nơi nào trong mã. Chẳng hạn, nếu phương thức Main ( ) gọi một phương thức có tên là A ( ) mà sau đó phương thức này sẽ gọi phương thức B ( ), bạn sẽ thấy ba phương thức của bạn trên stackB ( ), A ( ), và Main ( ) theo thứ tự ngược lại.

Sau đây là mã mà bạn có thể thêm vào để bắt đầu xem thông tin này:

```
Console.WriteLine("Currently loaded assemblies")
For Each Assm As System.Reflection.Assembly In _
  My.Application.Info.LoadedAssemblies
    Console.WriteLine(Assm.GetName( ).Name)
Next
Console.WriteLine( )
```

```
Console.WriteLine("Current stack trace: " & My.Application.Info.StackTrace)
Console.WriteLine( )
```

2.3. Sử dụng các tài nguyên được tạo kiểu mạnh

Ngoài mã, các hợp ngữ .NET cũng có chứa các tài nguyên được nhúng dữ liệu nhị phân chẳng hạn như các hình ảnh và các chuỗi được tạo mã cứng. Cho dù .NET đã hỗ trợ một hệ thống các tài nguyên kể từ Version 1.0, Visual Studio đã không đưa vào phần hỗ trợ thời gian thiết kế đã hợp nhất. Do vậy, các nhà phát triển cần chứa dữ liệu ảnh thường thêm nó vào một control hỗ trợ nó vào lúc thiết kế, chẳng hạn như PictureBox hay ImageList. Các control này chèn dữ liệu ảnh vào file tài nguyên ứng dụng một cách tự động.

Ghi chú

Các tài nguyên được tạo kiểu mạnh cho phép bạn nhúng dữ liệu tĩnh chẳng hạn như các hình ảnh vào các hợp ngữ đã được biên dịch và truy cập nó dễ dàng trong mã của bạn.

Trong Visual Studio 2005, bạn có thể dễ dàng thêm thông tin vào file tài nguyên (resources) và cập nhật nó sau đó. Thậm chí tốt hơn, bạn có thể truy cập thông tin này theo một dạng được tạo kiểu mạnh từ bất kỳ nơi nào trong mã của bạn.

2.3.1. Bạn làm điều đó như thế nào?

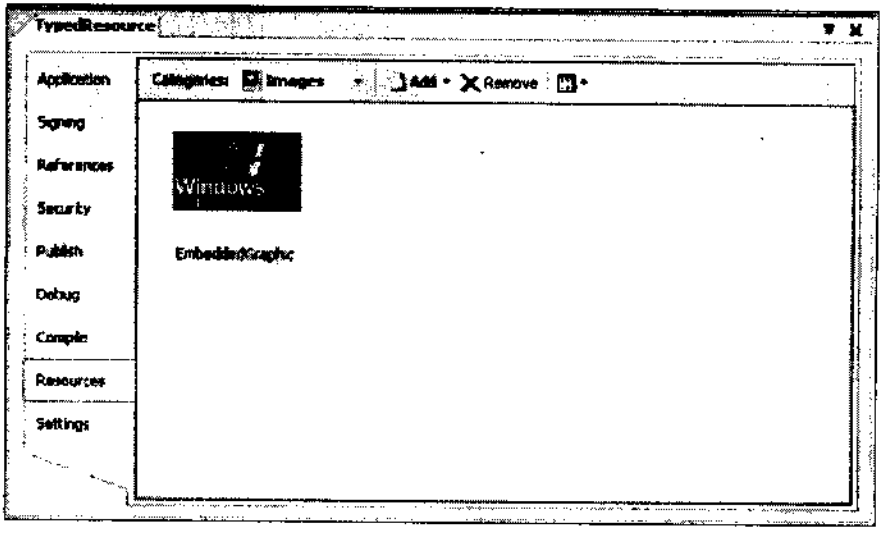
Để thử sử dụng một tài nguyên được tạo kiểu mạnh của một hình ảnh trong phần thực hành này, bạn cần tạo một ứng dụng Windows mới trước khi tiếp tục.

Để thêm một tài nguyên, hãy bắt đầu bằng cách nhấp nút vào nút My Project trong Solution Explorer. Thao tác này sẽ mở trình thiết kế ứng dụng, nơi bạn có thể cấu hình một loạt các xác lập liên quan đến ứng dụng. Kế tiếp, nhấp tab Resources. Trong hộp danh sách xổ xuống Categories, chọn loại tài nguyên bạn muốn xem (chuỗi, hình ảnh, audio,...). Khung xem chuỗi hiển thị một lưới các xác lập. Khung xem ảnh thì hơi khác một chút theo mặc định, nó hiển thị thumbnail của mỗi ảnh.

Để thêm một hình ảnh mới, bạn chọn hạng mục Images từ danh sách xổ xuống rồi chọn Add > Existing File từ thanh công cụ. Trình duyệt đến một file ảnh, chọn nó và nhấp OK. Nếu bạn không có sẵn một file ảnh, hãy thử sử dụng một file từ thư mục Windows, chẳng hạn như winnt256.bmp (đi kèm với hầu hết phiên bản của Windows).

Theo mặc định, tên tài nguyên có cùng tên với file nhưng bạn có thể

đổi tên nó sau khi thêm nó vào. Trong ví dụ ở đây, hãy đổi tên ảnh sang EmbeddedGraphic (như minh họa ở hình 2.2).



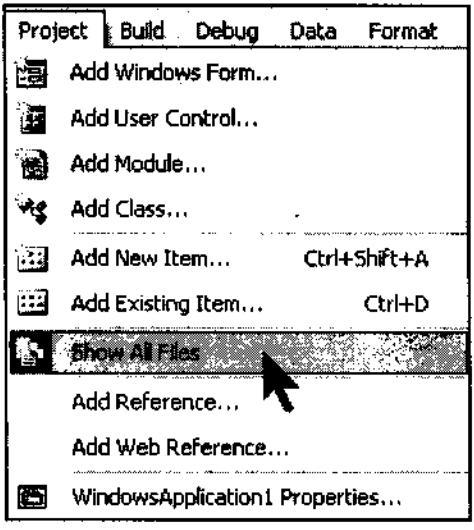
Hình 2.2. Thêm một hình ảnh ở dạng một tài nguyên được tạo kiểu mạnh

Sử dụng tài nguyên thì dễ. Mọi tài nguyên được biên dịch động thành một lớp tài nguyên được tạo kiểu mạnh mà bạn có thể truy cập qua My.Resources. Để thử tài nguyên này, bạn thêm một control PictureBox vào form Windows (và giữ lại tên mặc định PictureBox1). Sau đó thêm mã dưới đây để hiển thị hình ảnh khi form tải:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

  PictureBox1.Image = My.Resources.EmbeddedGraphic

End Sub
```

Ghi chú

Lớp tài nguyên (resources) được thêm vào thư mục My Project và được đặt tên Resources.Designer.vb. Để xem lớp này, bạn cần chọn Project > Show All Files. Dĩ nhiên, bạn đừng bao giờ thay đổi file này bằng tay.



Nếu bạn chạy mã, bạn sẽ nhìn thấy ảnh xuất hiện trên form. Để bảo đảm ảnh đang được trích từ hợp ngữ, thử biên dịch ứng dụng và sau đó xóa file ảnh (mã vẫn sẽ hoạt động liên tục).

Khi bạn thêm một tài nguyên theo cách này, Visual Studio sao chép tài nguyên đó vào thư mục con Resources của ứng dụng. Bạn có thể thấy thư mục này, cùng với tất cả các tài nguyên chứa trong nó, trong Solution Explorer. Khi bạn biên dịch ứng dụng, mọi tài nguyên được nhúng trong hợp ngữ. Tuy nhiên, có một ưu điểm rõ ràng là bảo lưu chúng trong một thư mục riêng. Bằng cách này, bạn có thể dễ dàng cập nhật một tài nguyên bằng cách thay thế file và biên dịch lại ứng dụng. Bạn không cần chỉnh sửa bất kỳ mã nào. Nó rất có lợi nếu bạn cần cập nhật nhiều hình ảnh hay nhiều nguồn tài nguyên khác cùng một lần.

Ghi chú

Một ưu điểm khác của các tài nguyên là bạn có thể sử dụng cùng hình ảnh trong nhiều control trên nhiều form khác nhau mà không cần thêm nhiều bản sao của cùng một file.

Bạn cũng có thể đính kèm một tài nguyên với nhiều console nhờ dùng cửa sổ Properties. Chẳng hạn, khi bạn nhấp dấu ellipsis (...) trong cửa sổ Properties kế bên thuộc tính Image dành cho control PictureBox, một trình thiết kế xuất hiện liệt kê tất cả những hình ảnh có sẵn trong các nguồn tài nguyên của ứng dụng.

2.3.2. Thế còn...

... ImageList? Nếu bạn là người phát triển Windows, bạn có lẽ đã

quen thuộc với control ImageList, vốn nhóm nhiều hình ảnh lại với nhau (thường là các bitmap nhỏ) để sử dụng trong các control khác, chẳng hạn như các menu, các thanh công cụ, các cây và các danh sách. ImageList không sử dụng các tài nguyên đã được tạo kiểu. Thay vào đó, nó sử dụng một kiểu chuỗi hóa tùy ý. Bạn sẽ nhận thấy tuy ImageList cung cấp sự hỗ trợ thời gian thiết kế và sự truy cập bằng lập trình vào các hình ảnh chứa trong nó, nhưng sự truy cập này không được tạo kiểu mạnh.

2.4. Sử dụng các xác lập cấu hình được tạo kiểu mạnh

Các ứng dụng thường cần các xác lập cấu hình để xác định các chi tiết như các vị trí file, các chuỗi nối kết cơ sở dữ liệu và các sở thích của người dùng. Thay vì tạo mã cứng các xác lập này (hay sáng tạo cơ chế riêng của bạn để lưu trữ chúng), .NET cho phép bạn thêm chúng vào một file cấu hình của riêng ứng dụng. File này cho phép bạn điều chỉnh các giá trị bằng cách hiệu chỉnh một file text mà không cần biên dịch lại ứng dụng của bạn.

Ghi chú

Sử dụng các xác lập cấu hình không dễ bị lỗi của trình thiết kế ứng dụng.

Trong Visual Studio 2005, các xác lập cấu hình còn dễ sử dụng hơn. Đó là vì chúng được biên dịch tự động thành một lớp tùy ý nhằm cung cấp sự truy cập được tạo kiểu mạnh vào chúng. Điều này có nghĩa là bạn có thể truy xuất các xác lập nhờ dùng các thuộc tính, với sự trợ giúp của IntelliSense, thay vì trông cậy vào các tiến trình tìm kiếm (lookup) dựa vào chuỗi. Thậm chí tốt hơn, .NET còn cải tiến mô hình này với khả năng sử dụng các xác lập có thể cập nhật được của người dùng để theo dõi sở thích và những thông tin khác. Bạn sẽ xem cách hoạt động của cả hai kỹ thuật này trong phần thực hành ở đây.

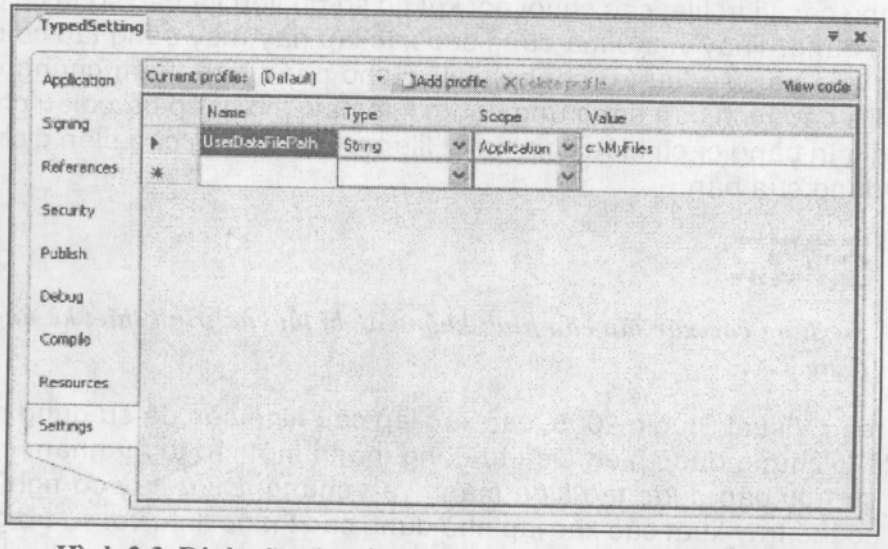
2.4.1. Bạn làm điều đó như thế nào?

Mọi xác lập cấu hình tùy ý được xác định với một tên chuỗi duy nhất. Trong các phiên bản trước của .NET, bạn có thể truy xuất giá trị của một xác lập cấu hình bằng cách tìm giá trị đó qua tên chuỗi của nó trong một tập hợp (collection). Tuy nhiên, nếu bạn sử dụng sai tên, bạn sẽ không nhận ra lỗi của bạn cho đến khi bạn chạy mã và nó thất bại với một ngoại lệ runtime (thời gian chạy).

Trong Visual Studio 2005, vấn đề này đã được cải tiến khá nhiều. Để thêm một xác lập cấu hình mới, bạn nhấp double vào nút My Project

trong Solution Explorer. Thao tác này sẽ mở trình thiết kế ứng dụng (application designer) nơi bạn có thể cấu hình một loạt các xác lập liên quan đến ứng dụng. Kế tiếp, nhấp tab Setting, tab này hiển thị một danh sách các xác lập cấu hình tùy ý mà bạn có thể định rõ các xác lập mới cũng như các giá trị của chúng.

Để thêm một xác lập cấu hình tùy ý vào ứng dụng của bạn, hãy nhập một tên xác lập mới ở cuối danh sách. Sau đó chỉ định kiểu dữ liệu, phạm vi và nội dung thật sự của xác lập. Chẳng hạn, để thêm một xác lập với một đường dẫn file, bạn có thể sử dụng tên UserDataFilePath, kiểu String, phạm vi Application (bạn sẽ tìm hiểu thêm về điều này một cách ngắn gọn), và trị c:\MyFiles. Hình 2.3 minh họa xác lập này.



Hình 2.3. Định rõ một xác lập ứng dụng được tạo kiểu mạnh

Ghi chú

Trong một ứng dụng web, các xác lập cấu hình được đặt trong file web.config. Trong các ứng dụng khác, các xác lập cấu hình được ghi vào một file cấu hình dùng tên của ứng dụng, cộng với phần mở rộng .config, như trong MyApp.exe.config.

Khi bạn thêm xác lập (setting), Visual Studio .NET chèn thông tin sau đây vào file cấu hình ứng dụng:

```
<configuration>
  <!— Other settings are defined here. —>
  <applicationSettings>
    <WindowsApplication1.MySettings>
```

```
        <setting name="UserDataFilePath" serializeAs="String">
          <value>c:\MyFiles</value>
        </setting>
      </WindowsApplication1.MySettings>
    </applicationSettings>
</configuration>
```

Cùng lúc này trong hậu cảnh, Visual Studio biên dịch một lớp có chứa thông tin về xác lập cấu hình tùy ý của bạn. Sau đó, bạn có thể truy cập xác lập theo tên ở bất kỳ nơi nào trong mã của bạn thông qua đối tượng My.Settings. Chẳng hạn, sau đây là mã truy xuất xác lập có tên UserDataFilePath:

```
Dim path As String

path = My.Settings.UserDataFilePath
```

Trong .NET 2.0, các xác lập cấu hình không cần thiết phải là các chuỗi. Bạn cũng có thể sử dụng các kiểu dữ liệu có thể chuỗi hóa khác, bao gồm các số nguyên, các số thập phân, ngày tháng và thời gian (chỉ cần chọn kiểu dữ liệu thích hợp từ danh sách xổ xuống Type). Các kiểu dữ liệu này được chuỗi hóa sang text trong file cấu hình, nhưng bạn có thể truy xuất chúng thông qua My.Settings ở dạng kiểu dữ liệu gốc, không cần phân ngữ.

Ghi chú

Lớp application settings (các xác lập ứng dụng) được thêm vào thư mục My Project và được đặt tên Settings.Designer.vb. Để xem lớp này, bạn chọn Project > Show All Files.

2.4.2. Cập nhật các xác lập

Ví dụ UserDataFilePath sử dụng một xác lập thuộc phạm vi ứng dụng, vốn có thể được đọc vào thời gian chạy nhưng không thể được chỉnh sửa. Nếu bạn cần thay đổi một xác lập thuộc phạm vi ứng dụng, bạn phải chỉnh sửa file cấu hình bằng tay (hoặc sử dụng danh sách settings trong Visual Studio).

Một lựa chọn khác là tạo các xác lập thuộc phạm vi người dùng. Để làm điều này, bạn chỉ cần chọn User từ danh sách xổ xuống Scope trong danh sách settings. Với một xác lập thuộc phạm vi người dùng, trị bạn xác lập trong Visual Studio được lưu trữ ở dạng mặc định trong file cấu hình trong thư mục ứng dụng. Tuy nhiên, khi bạn thay đổi các xác lập

này, một file user.config mới được tạo cho người dùng hiện hành và được lưu trong một thư mục của người dùng (với một tên ở dạng c:\Documents and Settings/[UserName]\Local Settings\Application Data\[ApplicationName]\[UniqueDirectory]).

Mẹo duy nhất liên quan đến các xác lập của người dùng là bạn phải gọi My.Settings.Save ( ) để chứa các thay đổi của bạn. Nếu không, các thay đổi sẽ chỉ tồn tại cho đến khi ứng dụng được đóng. Thông thường, bạn sẽ gọi My.Settings.Save ( ) khi ứng dụng kết thúc.

Để thử một xác lập thuộc phạm vi người dùng, bạn thay đổi phạm vi (scope) của xác lập UserDataFilePath từ Application sang User. Sau đó, tạo một form có một hộp text (được gọi là txtFilePath) và hai nút, một dùng để truy xuất dữ liệu người dùng (cmdRefresh) và một dùng để thay đổi nó (cmdUpdate). Sau đây là các event handler mà bạn sẽ sử dụng:

```
Private Sub cmdRefresh_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles cmdRefresh.Click
    txtFilePath.Text = My.Settings.UserDataFilePath
End Sub

Private Sub cmdUpdate_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles cmdUpdate.Click
    My.Settings.UserDataFilePath = txtFilePath.Text
End Sub
```

Sau cùng, để bảo đảm các thay đổi của bạn có ở đó vào lần kế tiếp bạn chạy ứng dụng, hãy yêu cầu .NET tạo hay cập nhật file user.config khi form đóng với mã này:

```
Private Sub Form1_FormClosed(ByVal sender As Object, _
  ByVal e As System.Windows.Forms.FormClosedEventArgs) _
  Handles Me.FormClosed
    My.Settings.Save( )
End Sub
```

Đây là một dạng form kiểm tra đơn giản. Bạn có thể chạy ứng dụng này và thử truy xuất xác lập hiện hành theo cách khác và chứa một xác lập mới. Nếu bạn quan tâm, bạn có thể theo dõi file user.config vốn chứa các xác lập đã thay đổi cho người dùng hiện hành.

2.5 Tạo các lớp chung an toàn kiểu

Các lập trình viên thường đối mặt với một lựa chọn khó khăn. Một mặt, cần phải tạo các giải pháp càng chung càng tốt để chúng có thể được sử dụng lại trong các tình huống khác. Chẳng hạn, tại sao phải tạo một lớp CustomerCollection chấp nhận chỉ các đối tượng thuộc kiểu Customer trong khi bạn có thể tạo một lớp Collection chung vốn có thể được cấu hình để chấp nhận các đối tượng thuộc bất kỳ kiểu nào? Mặt khác, phải xem xét các vấn đề thực thi và an toàn kiểu để đưa ra một giải pháp chung ít được mong đợi hơn. Nếu bạn sử dụng một lớp .NET Collection chung để chứa các đối tượng Customer, làm thế nào để chắc chắn một người nào đó không vô tình chèn một kiểu khác của đối tượng vào tập hợp, gây ra một lỗi sau đó?

Ghi chú

Bạn cần tạo một lớp đủ linh hoạt để làm việc với bất kỳ kiểu nào của đối tượng, nhưng có thể hạn chế các đối tượng mà nó chấp nhận trong bất kỳ trường hợp đã cho? Với các lớp chung (generic), VB có giải pháp hoàn hảo.

Visual Basic 2005 và .NET 2.0 cung cấp một giải pháp được gọi là generics. Generics là các lớp được thông số hóa theo kiểu. Nói cách khác, generics cho phép bạn tạo một mẫu lớp hỗ trợ bất kỳ kiểu nào. Khi bạn thể hiện lớp đó, bạn chỉ định kiểu mà bạn muốn sử dụng và từ điểm đó trở đi, đối tượng của bạn được "khóa vào" kiểu mà bạn đã chọn.

2.5.1. Bạn làm điều đó như thế nào?

Một ví dụ về nơi sử dụng generics có ý nghĩa là lớp System.Collections.ArrayList. ArrayList là một tập hợp tự định kích cỡ động đa năng. Nó có thể chứa các đối tượng .NET bình thường hoặc các đối tượng tùy ý của riêng bạn. Để hỗ trợ điều này, ArrayList xem mọi thứ như là kiểu Object gốc.

Vấn đề là không có cách nào để đặt ra các giới hạn về cách hoạt động của ArrayList. Chẳng hạn, nếu bạn muốn sử dụng ArrayList để chứa một tập hợp các đối tượng Customer, bạn không có cách nào để chắc chắn rằng một đoạn mã lỗi sẽ không vô tình chèn các chuỗi, các số nguyên hay một kiểu dữ liệu khác của đối tượng, gây ra các sự cố sau này. Vì lý do này, các nhà phát triển thường tạo các lớp tập hợp được tạo kiểu mạnh của riêng mình. Thật vậy, thư viện lớp .NET có hàng chục lớp tập hợp này.

Generics có thể giúp giải quyết vấn đề này. Chẳng hạn, nhờ sử dụng generics bạn có thể khai báo một lớp hoạt động với bất kỳ kiểu nào nhờ dùng từ khóa Of:

```
Public Class GenericList(Of ItemType)

    ' (Code goes here)

End Class
```

Trong trường hợp này, bạn đang tạo một lớp mới được tạo là GenericList có thể hoạt động với bất kỳ kiểu dữ liệu nào của đối tượng. Tuy nhiên, client cần chỉ định rõ kiểu nào nên được sử dụng. Trong mã lớp của bạn, bạn quy chiếu đến kiểu đó là ItemType. Dĩ nhiên, ItemType thật ra không phải là một kiểu dữ liệu mà nó chỉ là một placeholder dành cho kiểu mà bạn sẽ chọn khi bạn thể hiện một đối tượng GenericList.

Ví dụ 2.2 minh họa mã hoàn chỉnh cho một ArrayList an toàn kiểu đơn giản.

Ví dụ 2.2. Một tập hợp an toàn kiểu (typesafe) nhờ dùng generics

```
Public Class GenericList(Of ItemType)
    Inherits CollectionBase

    Public Function Add(ByVal value As ItemType) As Integer
        Return List.Add(value)
    End Function

    Public Sub Remove(ByVal value As ItemType)
        List.Remove(value)
    End Sub

    Public ReadOnly Property Item(ByVal index As Integer) As ItemType
        Get
            ' The appropriate item is retrieved from the List object and
            ' explicitly cast to the appropriate type, and then returned.
            Return CType(List.Item(index), ItemType)
        End Get
    End Property
End Class
```

Lớp GenericList bao bọc một ArrayList bình thường, được cung cấp qua thuộc tính List của lớp CollectionBase mà nó kế thừa từ đó. Tuy nhiên, lớp GenericList hoạt động khác với ArrayList qua việc cung cấp các phương thức Add ( ) và Remove ( ) được tạo kiểu mạnh vốn sử dụng placeholder ItemType.

Sau đây là một ví dụ về cách bạn có thể sử dụng lớp GenericList để tạo một tập hợp ArrayList chỉ hỗ trợ các chuỗi:

```
' Create the GenericList instance, and choose a type (in this case, string).
Dim List As New GenericList(Of String)

' Add two strings.
List.Add("blue")
List.Add("green")

' The next statement will fail because it has the wrong type.
' There is no automatic way to convert a GUID to a string.
' In fact, this line won't ever run, because the compiler
' notices the problem and refuses to build the application.
List.Add(Guid.NewGuid( ))
```

Không có giới hạn đối với số cách để thông số hóa một lớp. Trong ví dụ GenericList, chỉ có một thông số kiểu (type). Tuy nhiên, bạn có thể dễ dàng tạo một lớp hoạt động với hai hay ba kiểu của đối tượng và cho phép bạn làm cho mọi kiểu này trở thành kiểu chung (generic). Để sử dụng phương pháp này, bạn chỉ cần tách mỗi kiểu tham số bằng một dấu phẩy (giữa các dấu ngoặc ở đầu một lớp).

Ví dụ, hãy xem xét lớp GenericHashtable sau đây, lớp này cho phép bạn xác định kiểu của các mục mà tập hợp sẽ chứa (ItemType), cũng như kiểu của các khóa mà bạn sẽ sử dụng để tạo chỉ số cho các mục đó (KeyType):

```
Public Class GenericHashTable(Of ItemType, KeyType)

    Inherits DictionaryBase

    ' (Code goes here.)

End Class
```

Một đặc điểm quan trọng khác trong generics là khả năng áp dụng các ràng buộc (constraint) cho các thông số. Các ràng buộc hạn chế các kiểu được phép cho một lớp chung đã cho. Chẳng hạn, giả sử bạn muốn

tạo một lớp hỗ trợ chỉ các kiểu thực thi một giao diện cụ thể. Để làm điều này, trước tiên hãy khai báo kiểu hay các kiểu mà lớp chấp nhận và sau đó sử dụng từ khóa As để chỉ định lớp cơ sở mà kiểu phải phái sinh từ đó, hoặc giao diện mà kiểu phải thực thi.

Sau đây là một ví dụ hạn chế các mục chứa trong GenericList trong chỉ các mục có thể chuỗi hóa. Tính năng này sẽ hữu ích nếu bạn muốn thêm một phương thức vào GenericList đòi hỏi sự chuỗi hóa, chẳng hạn như một phương thức viết tất cả các mục trong danh sách vào một dòng:

```
Public Class SerializableList(Of ItemType As ISerializable)

  Inherits CollectionBase

  ' (Code goes here.)

End Class
```

Tương tự, sau đây là một tập hợp có thể chứa bất kỳ kiểu nào của đối tượng, miễn sao nó được phái sinh từ lớp System.Windows.Forms.Control. Kết quả cuối cùng là một tập hợp bị hạn chế trong các control, giống như tập hợp được hiển thị bởi thuộc tính Forms.Controls trên một cửa sổ:

```
Public Class ControlCollection(Of ItemType As Control)

  Inherits CollectionBase

  ' (Code goes here.)

End Class
```

Đôi khi, lớp chung (generic) của bạn có thể cần khả năng tạo lớp (thông số) parameter. Chẳng hạn, ví dụ GenericList có thể cần khả năng tạo một thể hiện của mục mà bạn muốn chứa trong tập hợp (collection). Trong trường hợp này, bạn cần sử dụng ràng buộc New. Ràng buộc New cho phép chỉ các kiểu thông số có hàm tạo đối số zero chung và không được đánh dấu là MustInherit. Điều này bảo đảm mã của bạn có thể tạo các phần thể hiện của kiểu thông số. Sau đây là một tập hợp đặt ra ràng buộc New:

```
Public Class GenericList(Of ItemType As New)

  Inherits CollectionBase

  ' (Code goes here.)

End Class
```

Cũng cần lưu ý rằng bạn có thể chỉ định nhiều ràng buộc như bạn

muốn, miễn sao bạn nhóm danh sách các ràng buộc trong các dấu ngoặc nhọn, như minh họa dưới đây:

```
Public Class GenericList(Of ItemType As {ISerializable, New})
  Inherits CollectionBase
  ' (Code goes here.)
End Class
```

Các ràng buộc được tuân theo bởi trình biên dịch, vì vậy nếu bạn vi phạm một quy tắc ràng buộc khi sử dụng một lớp generic, bạn sẽ không thể biên dịch ứng dụng của bạn.

Ghi chú

Generics được cài vào Common Language Runtime. Điều đó có nghĩa rằng chúng được hỗ trợ trong các ngôn ngữ .NET thuộc lớp đầu tiên, bao gồm cả C#.

2.5.2. Thế còn...

... việc sử dụng generics với các cấu trúc mã khác thì sao? Generics không chỉ hoạt động với các lớp. Chúng cũng có thể được sử dụng trong các cấu trúc, các giao diện và cả các phương thức (method).

Các chuyên viên thiết kế .NET Framework biết rất rõ tính hữu dụng của các tập hợp generic và họ đã tạo sẵn một vài tập hợp để bạn sử dụng. Bạn sẽ tìm thấy chúng trong namespace Systems.Collections_ Generic. Chúng bao gồm:

- List (một tập hợp cơ bản như ví dụ GenericList)
- Dictionary (một tập hợp tên - trị vốn tạo trị số cho mỗi mục bằng một khóa)
- LinkedList (một danh sách được liên kết, trong đó mỗi mục chỉ đến mục kế tiếp trong chuỗi)
- Queue (một tập hợp vào trước ra trước)
- Stack (vào sau ra trước)
- SortedList (một tập hợp tên - trị vốn được giữ lại theo thứ tự phân loại liên tục)

Đa số các kiểu này sao lặp một trong các kiểu namespace System.Collections. Các tập hợp cũ vẫn còn để tương thích ngược.

2.6. Tạo các kiểu dữ liệu Nullable đơn giản

Với sự hỗ trợ mới cho generics trong .NET Framework, nhiều tính năng mới trở nên khả thi. Một trong những tính năng này là các tập hợp generic được tạo kiểu mạnh đã minh họa trong phần thực hành trước "Tạo các lớp chung an toàn kiểu". Bây giờ, bạn sẽ thấy một cách khác mà generics có thể giải quyết các vấn đề thông thường, lần này sử dụng các kiểu dữ liệu nullable mới.

Ghi chú

Bạn cần trình bày dữ liệu có thể có hoặc không? Các kiểu nullable mới của VB .NET sẽ lấp khoảng hở này.

2.6.1. Bạn làm điều đó như thế nào?

Một trị null (được nhận biết trong Visual Basic qua từ khóa Nothing), là một cờ đặc biệt biểu thị hiện không có dữ liệu nào. Đa số các nhà phát triển đều quen thuộc với các tham chiếu đối tượng null, vốn biểu thị đối tượng đã được xác định nhưng chưa được tạo. Chẳng hạn, trong mã dưới đây, FileStream chứa một tham chiếu null bởi vì nó chưa được thể hiện với từ khóa New:

```
Dim fs As FileStream
If fs Is Nothing
    ' This is always true because the FileStream hasn't
    ' been created yet.
    Console.WriteLine("Object contains a null reference.")
End If
```

Các kiểu dữ liệu cốt lõi như integer (số nguyên) và string (chuỗi) không thể chứa các trị null. Các biến số được khởi tạo tự động với 0. Các biến Boolean là False. Các biến string được xác lập tự động sang một chuỗi rỗng (""). Thật vậy, ngay cả khi bạn trực xác lập một biến kiểu dữ liệu đơn giản theo cách tường minh sang Nothing trong mã của bạn, nó sẽ tự động chuyển trở lại trị rỗng (0, False, hay ""), như mã dưới đây minh họa:

```
Dim j As Integer = Nothing
If j = 0 Then
    ' This is always true because there is an
    ' implicit conversion between Nothing and 0 for integers.
    Console.WriteLine("Non-nullable integer j = " & j)
End If
```

Kiểu thiết kế này đôi khi gây ra sự cố, bởi vì không có cách nào để phân biệt giữa một trị rỗng và một trị chưa bao giờ được cung cấp ở nơi đầu tiên. Chẳng hạn, hãy tưởng tượng bạn tạo mã cần thiết để truy xuất số lần mà người dùng đã đặt hàng từ một file text. Sau đó, bạn kiểm tra trị này. Sự cố xảy ra nếu trị này bằng 0. Hoàn toàn đơn giản, bạn không có cách nào để biết đây có phải là dữ liệu hợp lệ hay không (người dùng chưa đặt đơn đặt hàng nào), hoặc nó thể hiện thông tin bị thiếu (xác lập này không thể được truy xuất hay người dùng hiện hành không phải là một khách hàng đã đăng ký).

Nhờ vào generics, .NET 2.0 có một giải pháp đó là lớp System.Nullable có thể bao bọc bất kỳ kiểu dữ liệu khác. Khi bạn tạo một trường hợp thể hiện của Nullable, bạn chỉ định kiểu dữ liệu. Nếu bạn không xác lập một trị thì phần thể hiện này có chứa một tham chiếu null. Bạn có thể kiểm tra điều này có đúng hay không bằng cách kiểm tra phương thức Nullable.HasType ( ) và bạn có thể truy xuất đối tượng nền tảng thông qua thuộc tính Nullable.Value.

Sau đây là mã mẫu mà bạn cần để tạo một kiểu số nguyên nullable:

```
Dim i As Nullable(Of Integer)

If Not i.HasValue Then
    ' This is true, because no value has been assigned.
    Console.WriteLine("i is a null value")
End If


' Assign a value. Note that you must assign directly to i, not i.Value.
' The i.Value property is read-only, and it always reflects the
' currently assigned object, if it is not Nothing.
i = 100
If i.HasValue Then
    ' This is true, because a value (100) is now present.
    Console.WriteLine("Nullable integer i = " & i.Value)
End If
```

2.6.2. Thế còn...

.. việc sử dụng Nullable với các đối tượng tham chiếu đầy đủ đặc tính thì sao? Mặc dù bạn không cần khả năng này (bởi vì các kiểu tham chiếu có thể chứa một tham chiếu null), nhưng nó vẫn đem lại cho bạn một số tiện lợi. Nghĩa là, bạn có thể sử dụng phương thức HasValue ( ) dễ đọc hơn thay vì kiểm tra để tìm Nothing. Trên hết, bạn có thể thực hiện sự

thay đổi này một cách liên tục, bởi vì lớp Nullable có khả năng đáng lưu ý là cho phép chuyển đổi ngầm giữa Nullable và kiểu mà nó bao bọc.

2.7. Sử dụng các toán tử với các đối tượng tùy biến

Tất cả các lập trình viên VB đều quen thuộc với các toán tử số học để cộng (+), trừ (-), chia (/), và nhân (*). Thông thường, các toán tử này được dành riêng cho các kiểu dữ liệu số .NET và không có ý nghĩa khi được sử dụng với các đối tượng khác. Tuy nhiên, trong VB.NET 2.0, bạn có thể tạo các đối tượng hỗ trợ tất cả những toán tử này, cũng như những toán tử được sử dụng cho các phép toán logic và phép chuyển đổi ngầm. Kỹ thuật này sẽ không có ý nghĩa đối với các đối tượng thương mại, nhưng nó cực kỳ tiện lợi nếu bạn cần tạo mô hình cho các cấu trúc toán học chẳng hạn như vector, ma trận, số phức, hay như được minh họa trong các phần ví dụ dưới đây.

Ghi chú

Bạn không muốn sử dụng cú pháp cồng kềnh như ObjA.Subtract(ObjB) để thực hiện các phép toán đơn giản trên các đối tượng tùy biến của bạn? Với sự hỗ trợ của VB cho sự quá tải toán tử, bạn có thể xử lý các đối tượng của bạn một cách dễ dàng như các số bình thường.

2.7.1. Bạn làm điều đó như thế nào?

Để quá tải một toán tử trong Visual Basic 2005, bạn cần tạo một phương thức toán tử đặc biệt trong lớp của bạn (hay trong cấu trúc). Phương thức này phải được khai báo với các từ khóa Public Shared Operator, theo sau là ký hiệu dành cho toán tử (ví dụ, +).

Thủ thuật

Việc quá tải một toán tử chỉ đơn giản là định nghĩa những gì mà một toán tử sẽ thực hiện khi được sử dụng với một kiểu nhất định của đối tượng. Nói cách khác, khi bạn quá tải toán tử + trong một lớp Fraction, bạn cho .NET biết những gì cần làm khi mã của bạn cộng hai đối tượng Fraction lại với nhau.

Ví dụ, đây là một phương thức toán tử tăng thêm sự hỗ trợ cho toán tử (+):

```
Public Shared Operator+(objA As MyClass, objB as MyClass) As MyClass

    (Code goes here.)
```

```
End Operator
```

Mọi phương thức toán tử (operator method) chấp nhận hai thông số, các thông số này tượng trưng cho các trị ở mỗi phía của toán tử. Tùy thuộc vào lớp và toán tử, thứ tự có thể rất quan trọng (như đối với phép chia).

Ngay sau khi bạn đã định nghĩa một toán tử, trình biên dịch VB sẽ gọi mã của bạn khi nó thực thi một câu lệnh vốn sử dụng toán tử với lớp. Ví dụ, trình biên dịch thay đổi mã như sau:

```
ObjC = ObjA + ObjB
```

thành:

```
ObjC = MyClass.Operator+ (ObjA, ObjA)
```

Ví dụ 2.3 minh họa cách bạn có thể quá tải các toán tử số học Visual Basic được sử dụng để xử lý các đối tượng Fraction. Một Fraction gồm có hai phần: một tử số và một mẫu số (gọi theo cách thông tục là "phần trên và phần dưới"). Mã Fraction quá tải các toán tử +, -, * và /, cho phép bạn thực hiện các phép tính phân số mà không phải đổi các số của bạn ra các số thập phân và làm mất đi độ chính xác.

Ví dụ 2.3. Quá tải các toán tử toán học trong lớp Fraction

```
Public Structure Fraction

    ' The two parts of a fraction.
    Public Denominator As Integer
    Public Numerator As Integer

    Public Sub New(ByVal numerator As Integer, ByVal denominator As Integer)
        Me.Numerator = numerator
        Me.Denominator = denominator
    End Sub

    Public Shared Operator +(ByVal x As Fraction, ByVal y As Fraction) _
      As Fraction
        Return Normalize(x.Numerator * y.Denominator + _
          y.Numerator * x.Denominator, x.Denominator * y.Denominator)
    End Operator

    Public Shared Operator -(ByVal x As Fraction, ByVal y As Fraction) _
```

```
  As Fraction
    Return Normalize(x.Numerator * y.Denominator - _
      y.Numerator * x.Denominator, x.Denominator * y.Denominator)
End Operator

Public Shared Operator *(ByVal x As Fraction, ByVal y As Fraction) _
  As Fraction
    Return Normalize(x.Numerator * y.Numerator, _
      x.Denominator * y.Denominator)
End Operator

Public Shared Operator /(ByVal x As Fraction, ByVal y As Fraction) _
  As Fraction
    Return Normalize(x.Numerator * y.Denominator, _
      x.Denominator * y.Numerator)
End Operator

' Reduce a fraction.
Private Shared Function Normalize(ByVal numerator As Integer, _
  ByVal denominator As Integer) As Fraction
    If (numerator <> 0) And (denominator <> 0) Then
        ' Fix signs.
        If denominator < 0 Then
            denominator *= -1
            numerator *= -1
        End If

        Dim divisor As Integer = GCD(numerator, denominator)
        numerator \= divisor
        denominator \= divisor
    End If

    Return New Fraction(numerator, denominator)
End Function

' Return the greatest common divisor using Euclid's algorithm.
```

```
    Private Shared Function GCD(ByVal x As Integer, ByVal y As Integer) _
     As Integer
      Dim temp As Integer

      x = Math.Abs(x)
      y = Math.Abs(y)
      Do While (y <> 0)
         temp = x Mod y
         x = y
         y = temp
      Loop

      Return x
    End Function

    ' Convert the fraction to decimal form.
    Public Function GetDouble( ) As Double
       Return CType(Me.Numerator, Double) / _
         CType(Me.Denominator, Double)
    End Function

    ' Get a string representation of the fraction.
    Public Overrides Function ToString( ) As String
       Return Me.Numerator.ToString & "/" & Me.Denominator.ToString
    End Function

  End Structure
```

Mã console được minh họa ở hình 2.4 đặt lớp fraction (phân số) qua một phép thử. Nhờ vào sự quá tải toán tử nên số vẫn ở dạng phân số và độ chính xác không bị mất đi.

Ví dụ 2.4. Kiểm tra lớp Fraction

```
  Module FractionTest

    Sub Main( )
       Dim f1 As New Fraction(2, 3)
       Dim f2 As New Fraction(1, 4)
```

```
        Console.WriteLine("f1 = " & f1.ToString( ))
        Console.WriteLine("f2 = " & f2.ToString( ))

        Dim f3 As Fraction
        f3 = f1 + f2      ' f3 is now 11/12
        Console.WriteLine("f1 + f2 = " & f3.ToString( ))

        f3 = f1 / f2      ' f3 is now 8/3
        Console.WriteLine("f1 / f2 = " & f3.ToString( ))

        f3 = f1 - f2      ' f3 is now 5/12
        Console.WriteLine("f1 - f2 = " & f3.ToString( ))

        f3 = f1 * f2      ' f2 is now 1/6
        Console.WriteLine("f1 * f2 = " & f3.ToString( ))
    End Sub

End Module
```

Khi bạn chạy ứng dụng này, bạn sẽ thấy kết xuất sau:

```
f1 = 2/3
f2 = 1/4
f1 + f2 = 11/12
f1 / f2 = 8/3
f1 - f2 = 5/12
f1 * f2 = 1/6
```

Thông thường, các thông số và trị trả về của một phương thức toán tử sử dụng cùng kiểu dữ liệu. Tuy nhiên, không có lý do nào khiến bạn không thể tạo hơn một phiên bản của một phương thức toán tử vì đối tượng của bạn có thể được sử dụng trong các biểu thức với các kiểu dữ liệu khác nhau.

2.7.2. Quá tải toán tử với các kiểu dữ liệu khác

Có nhiều lớp là những ứng cử tự nhiên cho sự quá tải toán tử. Sau đây là một số ví dụ:

- Các lớp toán học vốn tạo mẫu cho các vector, ma trận, số phức, hay tensor.

- Các lớp money vốn làm tròn các phép tính đến số gần nhất và hỗ trợ các loại tiền tệ khác nhau.
- Các lớp measurement có các đơn vị không theo quy cách như inches và feet.

2.8. Phân chia một lớp ra thành nhiều File

Nếu bạn đã từng phá vỡ một lớp .NET 2.0 Windows Forms, bạn sẽ nhận thấy toàn bộ mã được tạo tự động sẽ bị thiếu! Để hiểu nó đã được đưa đến nơi nào, bạn cần tìm hiểu một tính năng mới được gọi là các lớp partial, các lớp này cho phép bạn phân chia các lớp ra thành nhiều phần.

Ghi chú

Các lớp của bạn phát triển quá lớn đến nỗi không thể quản lý trong một file? Với từ khóa Partial mới, bạn có thể phân chia một lớp ra thành nhiều file riêng biệt.

2.8.1. Bạn làm điều đó như thế nào?

Bằng cách sử dụng từ khóa Partial mới, bạn có thể phân chia một lớp ra thành nhiều phần như bạn muốn. Bạn chỉ cần định nghĩa cùng một lớp ở nhiều nơi. Sau đây là một ví dụ định nghĩa một lớp có tên là SampleClass trong hai phần:

```
Partial Public Class SampleClass
    Public Sub MethodA( )
        Console.WriteLine("Method A called.")
    End Sub
End Class

Partial Public Class SampleClass
    Public Sub MethodB( )
        Console.WriteLine("Method B called.")
    End Sub
End Class
```

Trong ví dụ này, hai phần khai báo nằm trong cùng một file, phần này tiếp sau phần kia. Tuy nhiên, không có lý do nào khiến bạn không thể đặt hai phần SampleClass trong các file mã nguồn khác nhau trong cùng project. (Hạn chế duy nhất là bạn không thể định nghĩa hai phần này trong các hợp ngữ riêng biệt hay trong các namespace riêng biệt).

Khi bạn tạo ứng dụng có chứa mã trước, Visual Studio sẽ theo dõi mỗi lần của SampleClass và ráp nó vào một lớp đã được biên dịch hoàn chỉnh với hai phương thức MethodA ( ) và MethodB ( ). Bạn có thể sử dụng cả hai phương thức này, như minh họa ở đây:

```
Dim Obj As New SampleClass( )

Obj.MethodA( )

Obj.MethodB( )
```

Các lớp partial không giúp ích nhiều cho bạn trong việc giải quyết các vấn đề lập trình, nhưng chúng có thể hữu dụng trong việc phân chia các lớp cực kỳ lớn. Dĩ nhiên, sự tồn tại của các lớp lớn trong ứng dụng của bạn có thể là một dấu hiệu cho thấy bạn chưa phân tích chính xác vấn đề của bạn, trong trường hợp này bạn nên thật sự phân chia lớp của mình ra thành các lớp riêng biệt, không phải theo từng phần (partial). Một trong những vai trò chính của các lớp partial trong .NET là che giấu mã thiết kế vốn được tạo tự động bởi Visual Studio, sự hiển thị của nó trong các phiên bản trước là nguyên nhân gây ra sự phiền toái cho một số lập trình viên VB.

Chẳng hạn, khi bạn tạo một form Windows .NET trong Visual Basic 2005, mã xử lý sự kiện (event) của bạn đặt trong file mã nguồn dành cho form, nhưng mã thiết kế để tạo và cấu hình mỗi control và nối kết các event handler của nó thì không thấy ở bất kỳ nơi nào. Để xem mã này, bạn cần chọn Project > Show All Files từ menu Visual Studio. Khi bạn chọn, file chứa nửa bị thiếu của lớp xuất hiện trong Solution Explorer ở dạng một file riêng biệt. Đặt tên cho một form là Form1, bạn sẽ thật sự có được một file Form1.vb có chứa mã của bạn và một file Form1.Designer.vb chứa phần được tạo tự động.

2.8.2. Thế còn...

... việc sử dụng từ khóa Partial với các cấu trúc thì sao? Điều đó cũng xảy ra nhưng bạn không thể tạo các giao diện từng phần, các phép liệt kê hay bất kỳ cấu trúc lập trình .NET khác.

2.9. Mở rộng Namespace My

Các đối tượng My không được định nghĩa ở một nơi duy nhất. Một số đến từ các lớp được định nghĩa trong namespace Microsoft.Visual Basic.MyServices, trong khi một số khác được tạo động khi bạn thêm các form, các dịch vụ web, các xác lập cấu hình và các nguồn tài nguyên được nhúng vào project của bạn. Tuy nhiên, với tư cách là người phát triển bạn có thể gia nhập vào namespace My và mở rộng nó bằng các

thành phần riêng của bạn (ví dụ, các phép tính hữu ích và các tác vụ của riêng ứng dụng).

Ghi chú

Bạn sử dụng các đối tượng My quá nhiều và bạn muốn tự mình tùy biến chúng? VB 2005 cho phép bạn cài vào các lớp riêng của bạn.

2.9.1. Bạn làm điều đó như thế nào?

Để cài một lớp mới vào hệ thống phân cấp đối tượng My, bạn chỉ cần sử dụng một khối Namespace với tên My. Chẳng hạn, bạn có thể thêm mã này để tạo một lớp BusinessFunctions mới có chứa một hàm thuộc về công ty để tạo các tên định danh tùy ý (bằng cách nối tên khách hàng với một GUID mới):

```
Namespace My


  Public Class BusinessFunctions
    Public Shared Function GenerateNewCustomerID( _
      ByVal name As String) As String
        Return name & "_" & Guid.NewGuid.ToString( )
    End Function
  End Class


End Namespace
```

Ngay sau khi bạn đã tạo đối tượng BusinessFunctions ở đúng chỗ, bạn có thể sử dụng nó trong ứng dụng của mình giống như bất kỳ đối tượng My khác. Ví dụ, để hiển thị một customer ID mới:

```
Console.WriteLine(My.BusinessFunctions.GenerateNewCustomerID("matthew"))
```

Lưu ý rằng các lớp My mà bạn thêm vào cần sử dụng các phương thức và thuộc tính chia sẻ. Đó là bởi vì đối tượng My sẽ không được thể hiện tự động. Do vậy, nếu bạn sử dụng các thành viên thể hiện bình thường, bạn sẽ cần tự mình tạo đối tượng My và bạn sẽ không thể xử lý nó với cùng một cú pháp. Một giải pháp khác là tạo một module trong namespace My, bởi vì tất cả các phương thức và thuộc tính trong một module luôn luôn được chia sẻ.

Bạn cũng có thể mở rộng một số đối tượng My hiện có nhờ vào các lớp partial. Chẳng hạn, nhờ sử dụng tính năng này bạn có thể thêm

thông tin mới vào đối tượng My.Computer hay các thường mới vào đối tượng My.Application. Trong trường hợp này, phương pháp hơi khác một chút. My.Computer bộc lộ một thể hiện của đối tượng MyComputer. My.Application bộc lộ một thể hiện của đối tượng MyApplication. Do đó để thêm vào một trong hai lớp này, bạn cần tạo một lớp partial với tên thích hợp và thêm các thành viên thể hiện mà bạn cần. Bạn cũng nên khai báo lớp này với từ khóa truy cập Friend cho phù hợp với lớp hiện tại.

Ghi chú

Các thành viên chia sẻ là các thành viên luôn có sẵn qua tên lớp, cho dù bạn chưa tạo một đối tượng. Nếu bạn sử dụng các biến chia sẻ, sẽ có một bản sao của biến đó, tổng thể của toàn bộ ứng dụng.

Sau đây là một ví dụ mà bạn có thể sử dụng để mở rộng My.Application với một phương thức kiểm tra các phiên bản cập nhật:

```
Namespace My

    Partial Friend Class MyApplication
        Public Function IsNewVersionAvailable( ) As Boolean
            ' Usually, you would read the latest available version number
            ' from a web service or some other resource.
            ' Here, it's hardcoded.
            Dim LatestVersion As New Version(1, 2, 1, 1)
            Return Application.Info.Version.CompareTo(LatestVersion)
        End Function
    End Class

End Namespace
```

Và bây giờ bạn có thể sử dụng phương thức này:

```
If My.Application.IsNewVersionAvailable( )
    Console.WriteLine("A newer version is available.")
Else
    Console.WriteLine("This is the latest version.")
End If
```

2.9.2. Thế còn...

... Việc sử dụng các phần mở rộng My của bạn trong nhiều ứng dụng thì sao? Không có lý do nào khiến bạn không thể xử lý các lớp My giống như cách bạn xử lý bất kỳ lớp hữu khác mà bạn muốn sử dụng lại trong nhiều ứng dụng. Nói cách khác, bạn có thể tạo một project thư viện lớp (class library), thêm một số phần mở rộng My và biên dịch nó thành một DLL. Sau đó, bạn có thể tham chiếu DLL đó trong các ứng dụng khác.

Dĩ nhiên, việc mở rộng namespace My theo cách đó có hai nhược điểm lớn:

- Việc chia sẻ thành phần của bạn với các ngôn ngữ khác trở nên khó khăn hơn. Chẳng hạn, C# không cung cấp một tính năng My. Mặc dù, bạn vẫn có thể sử dụng một đối tượng My tùy ý trong một ứng dụng C# nhưng nó không cài vào một cách chặt chẽ.
- Khi bạn sử dụng namespace My, bạn lảng tránh một trong hai ích lợi lớn của các namespace và tránh các xung đột tên. Chẳng hạn, xem xét hai công ty tạo ra các thành phần để ghi nhật ký. Nếu bạn sử dụng chuẩn namespace .NET được đề nghị (CompanyName.Application.ClassName), ít có khả năng hai thành phần này sẽ có các tên được định tính đầy đủ như nhau. Một có thể là Acme.SuperLogger.Logger trong khi thành phần còn lại là ComponentTech.LogMagic.Logger. Tuy nhiên, nếu cả hai đều mở rộng một đối tượng My thì hoàn toàn có khả năng chúng sẽ sử dụng cùng một tên (như My.Application.Logger). Do vậy, bạn sẽ không thể sử dụng cả hai trong cùng một ứng dụng.

2.10. Chuyển sang lần lặp kế tiếp của một vòng lặp

Ngôn ngữ Visual Basic có cung cấp một vài câu lệnh điều khiển dòng lưu thông, các câu lệnh này cho phép bạn điều khiển trực tiếp sự thực thi của mã. Ví dụ, bạn có thể sử dụng Return để ra khỏi một hàm hay Exit để trở lại từ một vòng lặp. Tuy nhiên, trước VB 2005, không có bất kỳ cách nào để chuyển sang lần lặp kế tiếp của một vòng lặp.

Ghi chú

Từ khóa Continue mới của VB cho bạn một cách nhanh để thoát khỏi một khối mã bị làm rối trong một vòng lặp và đưa thẳng vào lần lặp kế tiếp.

2.10.1. Bạn làm điều đó như thế nào?

Câu lệnh Continue là một trong những chi tiết ngôn ngữ nhanh chóng chứng tỏ sự tiện lợi. Câu lệnh Continue tồn tại trong ba phiên bản: Continue For, Continue Do, và Continue While, mỗi phiên bản được sử dụng với một kiểu vòng lặp khác (For ... Next, Do ... Loop, hay While ... End While).

Để xem cách hoạt động của câu lệnh Continue, hãy xem xét mã dưới đây:

```
For i = 1 to 1000
    If i Mod 5 = 0 Then
        ' (Task A code.)
        Continue For
    End If
    ' (Task B code.)
Next
```

Mã này lặp 1.000 lần, gia số một bộ đếm i. Bất kỳ khi nào i có thể chia hết cho năm, mã task A thực thi. Sau đó, câu lệnh Continue For được thực thi, bộ đếm (counter) được gia tăng và sự thực thi tiếp tục ở phần bắt đầu của vòng lặp, bỏ qua mã trong task B.

Trong ví dụ này, câu lệnh continue không thật sự bị bắt buộc, bởi vì bạn có thể viết lại mã một cách dễ dàng như sau:

```
For i = 1 to 1000
    If i Mod 5 = 0 Then
        ' (Task A code.)
    Else
        ' (Task B code.)
    End If
Next
```

Tuy nhiên, điều này hầu như không khả thi như vậy nếu bạn cần thực hiện một vài phép thử khác nhau. Để xem lợi ích thật sự của câu lệnh Continue, bạn cần xem xét một ví dụ phức tạp hơn (và hiện thực).

Ví dụ 2.5 minh họa một vòng lặp có thể lướt qua một mảng các từ. Mỗi từ được phân tích và chương trình quyết định từ được tạo thành từ các chữ, các ký tự số hay ký tự khoảng cách. Nếu chương trình trùng khớp với một phép thử (ví dụ, phép thử letter), nó cần tiếp tục đến từ kế tiếp mà không thực hiện phép thử kế tiếp. Để thực hiện điều này mà không cần sử dụng câu lệnh Continue, bạn cần sử dụng các vòng lặp lồng nhau, một phương pháp tạo mã khó sử dụng.

Ví dụ 2.5. Phân tích một chuỗi mà không sử dụng câu lệnh Continue

```
' Define a sentence.
Dim Sentence As String = "The final number is 433."

' Split the sentence into an array of words.
Dim Delimiters( ) As Char = {" ", ".", ","}
Dim Words( ) As String = Sentence.Split(Delimiters)

' Examine each word.
For Each Word As String In Words
    ' Check if the word is blank.
    If Word <> "" Then
        Console.Write("'" + Word + "'" & vbTab & "= ")

        ' Check if the word is made up of letters.
        Dim AllLetters As Boolean = True
        For Each Character As Char In Word
            If Not Char.IsLetter(Character) Then
                AllLetters = False
            End If
        Next
        If AllLetters Then
            Console.WriteLine("word")
        Else
            ' If the word isn't made up of letters,
            ' check if the word is made up of numbers.
            Dim AllNumbers As Boolean = True
            For Each Character As Char In Word
                If Not Char.IsDigit(Character) Then
                    AllNumbers = False
                End If
            Next
            If AllNumbers Then
                Console.WriteLine("number")
            Else
                ' If the word isn't made up of letters or numbers,
```

```
            ' assume it's something else.
            Console.WriteLine("mixed")
        End If
    End If
End If
Next
```

Bây giờ, hãy xem xét phiên bản đã được viết lại được minh họa trong ví dụ 2.6, sử dụng câu lệnh Continue để làm rõ những gì đang diễn ra.

Ví dụ 2.6. Phân tích một chuỗi nhờ sử dụng câu lệnh Continue

```
' Examine each word.
For Each Word As String In Words
    ' Check if the word is blank.
    If Word = "" Then Continue For
    Console.Write("'" + Word + "'" & vbTab & "= ")

    ' Check if the word is made up of letters.
    Dim AllLetters As Boolean = True
    For Each Character As Char In Word
        If Not Char.IsLetter(Character) Then
            AllLetters = False
        End If
    Next
    If AllLetters Then
        Console.WriteLine("word")
        Continue For
    End If

    ' If the word isn't made up of letters,
    ' check if the word is made up of numbers.
    Dim AllNumbers As Boolean = True
    For Each Character As Char In Word
        If Not Char.IsDigit(Character) Then
            AllNumbers = False
        End If
    Next
```

```
        If AllNumbers Then
            Console.WriteLine("number")
            Continue For
        End If

        ' If the word isn't made up of letters or numbers,
        ' assume it's something else.
        Console.WriteLine("mixed")
    Next
```

2.10.2. Thế còn...

... việc sử dụng Continue trong một vòng lặp xếp lồng thì sao? Điều đó là khả thi. Nếu bạn xếp lồng một vòng lặp For bên trong một vòng lặp Do, bạn có thể sử dụng Continue For để chuyển sang lần lặp kế tiếp của vòng lặp bên trong, hay Continue Do để chuyển sang lần lặp kế tiếp của vòng lặp ngoài. Kỹ thuật này cũng có tác dụng theo thứ tự ngược lại (với một vòng lặp Do bên trong một vòng lặp For), nhưng nó không có tác dụng nếu bạn xếp lồng một vòng lặp bên trong một vòng lặp khác cùng kiểu. Trong trường hợp này, không có cách nào để tham chiếu vòng lặp ngoài, vì vậy câu lệnh Continue của bạn luôn luôn tham chiếu vòng lặp trong.

2.11. Loại bỏ các đối tượng tự động

Trong .NET, cần bảo đảm các đối tượng vốn sử dụng các nguồn tài nguyên chưa được quản lý (ví dụ, các file handle, các nối kết cơ sở dữ liệu và các ngữ cảnh đồ họa) giải phóng các nguồn tài nguyên này càng sớm càng tốt. Về vấn đề này, các đối tượng như vậy luôn luôn thực thi giao diện IDisposable và cung cấp một phương thức Dispose ( ) mỗi trường bạn có thể gọi để giải phóng ngay các nguồn tài nguyên của chúng.

Vấn đề duy nhất với kỹ thuật này là bạn phải luôn luôn nhớ gọi phương thức Dispose ( ) (hoặc một phương thức khác gọi Dispose ( ), chẳng hạn như phương thức Close ( )). VB 2005 cung cấp một phương pháp bảo vệ mới mà bạn có thể áp dụng để bảo đảm Dispose ( ) luôn được gọi: câu lệnh Using.

2.11.1. Bạn làm điều đó như thế nào?

Bạn sử dụng câu lệnh Using trong một cấu trúc khối. Ở dòng đầu tiên, khi bạn khai báo khối Using, bạn chỉ định đối tượng có thể được loại bỏ mà bạn đang sử dụng. Thông thường, bạn cũng có thể tạo đối tượng

đồng thời nhờ dùng từ khóa New. Sau đó bạn viết mã vốn sử dụng đối tượng có thể loại bỏ (disposable) bên trong khối Using. Sau đây là một ví dụ với một đoạn mã tạo một file mới và viết một vài dữ liệu vào file:

```
Using NewFile As New System.IO.StreamWriter("c:\MyFile.txt")
    NewFile.WriteLine("This is line 1")
    NewFile.WriteLine("This is line 2")
End Using

' The file is closed automatically.

' The NewFile object is no longer available here.
```

Trong ví dụ này, ngay khi sự thực thi rời khỏi khối Using, phương thức Dispose ( ) được gọi trên đối tượng NewFile, giải phóng file handle.

2.11.2. Thế còn...

... các lỗi xảy ra bên trong một khối Using thì sao? Thật may mắn, .NET bảo đảm nó loại bỏ tài nguyên cho dù bạn thoát khối using bằng cách nào, ngay cả khi một ngoại lệ chưa được xử lý xảy ra.

Câu lệnh Using có ý nghĩa với tất cả các loại đối tượng có thể được loại bỏ, chẳng hạn như:

- File (bao gồm FileStream, StreamReader, và StreamWriter)
- Các nối kết cơ sở dữ liệu (bao gồm SqlConnection, OracleConnection, và OleDbConnection)
- Các nối kết mạng (bao gồm TcpClient, UdpClient, NetworkStream, FtpWebresponse, HttpWebResponse)
- Đồ họa (bao gồm Image, Bitmap, Metafile, Graphics)

2.12. Bảo vệ các thuộc tính với khả năng truy cập phân chia

Đa số các thuộc tính đều có một thủ tục property get (cho phép bạn truy xuất trị thuộc tính) và một thủ tục property set (cho phép bạn xác lập một trị mới cho thuộc tính). Trong các phiên bản trước của Visual Basic, cấp độ truy cập được khai báo của cả hai thủ tục cần phải giống nhau. Trong VB 2005, bạn có thể bảo vệ một thuộc tính bằng cách gán cho thủ tục set một cấp độ truy cập thấp hơn bạn gán cho thủ tục get.

Ghi chú

Trước đây, không có cách nào để tạo một thuộc tính mà mọi người có

thể đọc nhưng chỉ ứng dụng của bạn là có thể cập nhật. Sau cùng VB 2005 đã nới lỏng các quy tắc này và cho bạn nhiều sự linh hoạt hơn.

2.12.1. Bạn làm điều đó như thế nào?

VB nhận biết ba cấp độ truy cập. Chúng bao gồm:

- Public (có sẵn cho mọi lớp trong mọi hợp ngữ)
- Friend (có sẵn cho toàn bộ mã trong tất cả các lớp trong hợp ngữ hiện hành)
- Private (chỉ có sẵn cho mã trong cùng lớp)

Hãy tưởng tượng bạn đang tạo một thành phần DLL sẽ được sử dụng bởi một ứng dụng khác. Bạn có thể quyết định tạo một thuộc tính được gọi là Status mà ứng dụng client cần đọc, vì vậy bạn khai báo thuộc tính Public:

```
Public Class ComponetClass

    Private _Status As Integer
    Public Property Status( ) As Integer
        Get
            Return _Status
        End Get
        Set(ByVal value As Integer)
            _Status = value
        End Set
    End Property

End Class
```

Vấn đề ở đây là cấp độ truy cập được gán cho thuộc tính Status cho phép client thay đổi nó, mà điều này lại chẳng có ý nghĩa. Bạn có thể làm cho Status trở thành một thuộc tính read-only nghĩa là chỉ đọc (nói cách khác, bỏ qua thủ tục property set), nhưng điều đó sẽ không cho phép các lớp khác vốn là thành phần của các ứng dụng và nằm trong tập hợp thành phần để thay đổi nó.

Giải pháp là cung cấp cho thủ tục property set cấp độ truy cập Friend. Sau đây là diện mạo mã, với sự thay đổi duy nhất được tô nổi:

```
Public Property Status( ) As Integer
    Get
        Return _Status
    End Get
    Friend Set(ByVal value As Integer)
        _Status = value
    End Set
End Property
```

2.12.2. Thế còn...

... các thuộc tính read-only (chỉ đọc) và write-only (chỉ viết) thì sao? Khả năng truy cập phân chia (split accessibility) không giúp ích cho bạn nếu bạn cần tạo một thuộc tính read-only (chẳng hạn như một trị đã được tính) hay một trị write-only (chẳng hạn như một password vẫn không được truy cập). Để tạo một thuộc tính read-only, hãy thêm từ khóa ReadOnly vào phần khai báo thuộc tính (ngay sau từ khóa truy cập), và loại bỏ thủ tục property set. Để tạo một thuộc tính write-only, bạn loại bỏ thủ tục property get và thêm từ khóa WriteOnly. Các từ khóa này chẳng có gì mới bởi vì chúng đã có sẵn kể từ Visual Basic .NET 1.0.

2.13. Lượng giá các điều khiển riêng biệt với Logic mạch ngắn (Short-Circuit Logic)

Trong các phiên bản trước của VB, có hai toán tử logic: And và Or, Visual Basic 2005 giới thiệu hai toán tử mới bổ sung cho hai toán tử này: AndAlso và OrElse. Các toán tử này hoạt động theo cùng cách với And và Or, ngoại trừ chúng hỗ trợ cho sự tạo mạch ngắn (short-circuiting), vốn cho phép bạn lượng giá chỉ một phần của một câu lệnh điều kiện dài.

Ghi chú

Với short-circuiting, bạn có thể kết hợp nhiều điều kiện để viết mã súc tích hơn.

2.13.1. Bạn làm điều đó như thế nào?

Một tình huống lập trình phổ biến là nhu cầu lượng giá nhiều điều kiện liên tiếp. Thông thường, điều này liên quan đến việc kiểm tra một đối tượng không phải là null và sau đó xem xét một trong các thuộc tính của

nó. Để xử lý tình huống này, bạn cần sử dụng các khối If xếp lồng, như minh họa dưới đây:

```
If MyObject Is Nothing Then
    If MyObject.Value > 10 Then
        ' (Do something.)
    End If
End If
```

Sẽ rất tốt nếu kết hợp cả hai điều kiện này vào thành một dòng, như sau:

```
If MyObject Is Nothing And MyObject.Value > 10 Then
    ' (Do something.)
End If
```

Thật không may, điều này sẽ không có tác dụng bởi vì VB luôn lượng giá cả hai điều kiện. Nói cách khác, cho dù MyObject là Nothing thì VB sẽ lượng giá điều kiện thứ hai và cố truy xuất thuộc tính MyObject.Value, vốn sẽ gây ra một NullReferenceException.

Visual Basic 2005 giải quyết vấn đề này với các từ khóa AndAlso và OrElese. Khi bạn sử dụng các từ khóa này, Visual Basic sẽ không lượng giá điều kiện thứ hai nếu điều kiện đầu tiên là sai. Sau đây là mã đã chỉnh sửa:

```
If MyObject Is Nothing AndAlso MyObject.Value > 10 Then
    ' (Do something.)
End If
```

2.13.2. Thế còn...

... các phần tinh chỉnh ngôn ngữ khác thì sao? Trong chương này, chúng ta đã khảo sát các điểm đổi mới ngôn ngữ VB quan trọng nhất. Tuy nhiên, cần chỉ ra một vài điểm ít quan trọng hơn nhưng chưa được đưa vào chương nào:

- Từ khóa IsNot cho phép bạn đơn giản cú pháp. Nhờ sử dụng nó, bạn có thể thay thế cú pháp như If Not x Is Nothing bằng câu lệnh tương đương If x IsNot Nothing.
- Hàm tryCast ( ) cho phép bạn lượt bớt mã ép kiểu. Nó hoạt động giống như CType ( ) hoặc DirectCast ( ), với một ngoại

lệ là nếu đối tượng không thể được chuyển đổi sang kiểu yêu cầu thì phần tham chiếu null được trả về. Do vậy, thay vì kiểm tra kiểu của một đối tượng và sau đó ép kiểu nó, bạn có thể sử dụng ngay tryCast ( ) và sau đó kiểm tra xem bạn có một thể hiện đối tượng thật sự hay không.

- Các kiểm nguyên không có dấu cho phép bạn chứa các trị số không âm. Hạn chế đó nằm ở phần lưu trữ bộ nhớ, cho phép bạn chứa các số lớn hơn. Các số không dấu đã luôn có trong .NET Framework, nhưng bây giờ VB 2005 đưa các từ khóa vào cho chúng (UInteger, Ulong và UShort).

Chương 3

Các ứng dụng Windows

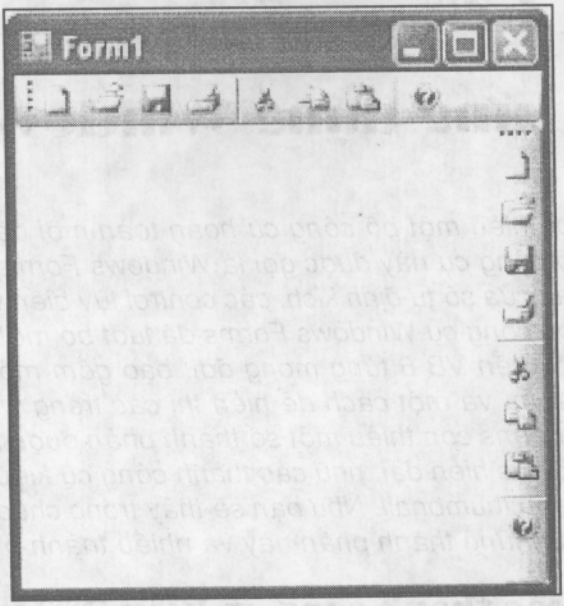
.NET 1.0 đã giới thiệu một bộ công cụ hoàn toàn mới để viết các ứng dụng Windows. Bộ công cụ này được gọi là Windows Forms, nó có nhiều tính năng để tạo các cửa sổ tự định kích, các control tùy biến và các đồ họa động. Tuy nhiên, bộ công cụ Windows Forms đã lượt bỏ một vài tính năng mà nhiều nhà phát triển VB 6 từng mong đợi, bao gồm một control hiệu chỉnh được tạo mặt nạ và một cách để hiển thị các trang web HTML. Bộ công cụ Windows Forms còn thiếu một số thành phần được tìm thấy trong các ứng dụng Windows hiện đại, như các thanh công cụ kiểu Office XP và các menu với các ảnh thumbnail. Như bạn sẽ thấy trong chương này, .NET 2.0 bao gồm tất cả những thành phần này và nhiều thành phần khác.

3.1. Sử dụng các thanh công cụ kiểu Office

Với .NET 1.0 và 1.1, các nhà phát triển VB đã tự hài lòng với control ToolBar mà nay đã lỗi thời, hoặc vẽ các thanh công cụ tùy ý riêng của họ bằng tay. Trong .NET 2.0, điều này đã được cải tiến với một control ToolStrip mới, hiện đại, xử lý chính xác các đề tài (theme) Windows XP và hỗ trợ nhiều thành phần đồ họa như các nút, các nhãn, các danh sách xổ xuống, các menu xổ xuống, các hộp text và nhiều thứ khác.

3.1.1. Bạn làm điều đó như thế nào?

Để sử dụng control System.Windows.Forms.ToolStrip, bạn chỉ cần rê ToolStrip từ mục Menu & Toolbars của hộp công cụ Visual Studio lên trên một form. Để điều khiển ToolStrip gióng thẳng với phía nào của form, hãy xác lập thuộc tính Docking. Chẳng hạn, hình 3.1 minh họa một form, Form1, với hai control ToolStrip, một được neo cố định vào đỉnh form và control kia được neo vào cạnh phải.

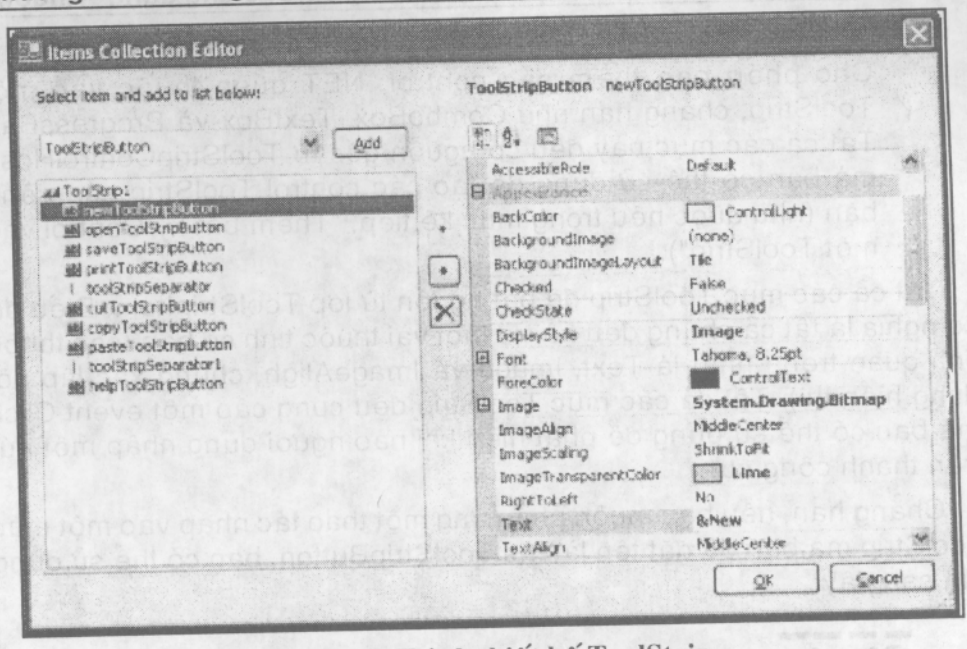


Hình 3.1. Ba đối tượng ToolStrip trong một RaftingContainer

Ghi chú

Sau cùng, một control ToolStrip đã có diện mạo xứng đáng với một ứng dụng Windows hiện đại.

Để thêm các nút vào ToolStrip, bạn có thể sử dụng trình thiết kế Visual Studio. Chỉ cần nhấp vào thẻ thông minh ToolStrip và chọn Edit Items. Bạn có thể chọn các mục mới từ một danh sách xổ xuống và cấu hình các thuộc tính của chúng trong một cửa sổ giống như cửa sổ minh họa ở hình 3.2. Hoặc chọn Insert Standard Items để tạo các nút ToolStrip chuẩn cho việc quản lý tài liệu (new, open, save, close) và hiệu chỉnh (cut, copy, paste).



Hình 3.2. Trình thiết kế ToolStrip

Chìa khóa để nắm vững control ToolStrip là nắm vững mọi thành phần khác nhau mà bạn có thể đặt bên trong nó. Chúng bao gồm:

ToolStripButton

Tiêu biểu cho một mục trên thanh công cụ mà người dùng có thể nhấp vào. Nó có thể chứa text hay một hình ảnh (hoặc cả hai). Đây là mục ToolStrip phổ biến nhất.

ToolStripLabel

Tiêu biểu cho một mục không thể được chọn trên ToolStrip. Nó có thể chứa text hay một hình ảnh (hoặc cả hai).

ToolStripSeparator

Phân chia các mục kề nhau trong một ToolStrip bằng một đường khắc mãnh.

ToolStripDropDownButton và ToolStripSplitButton

Tiêu biểu cho một menu xổ xuống với các mục. Điểm khác biệt duy nhất là cách vẽ danh sách xổ xuống. ToolStripDropDown Button hiển thị các mục của nó ở dạng một menu, với một lề thumbnail và khả năng kiểm tra các mục. Trong cả hai trường hợp, các mục menu là những đối tượng ToolStripMenuItem được thêm vào tập hợp do thuộc tính DropDownItems hiển thị.

ToolStripComboBox, ToolStripTextBox và ToolStripProgressBar

Cho phép bạn thêm các control .NET quen thuộc vào một ToolStrip, chẳng hạn như ComboBox, TextBox và ProgressBar. Tất cả các mục này đều có nguồn gốc từ ToolStripControlHost, mà bạn có thể sử dụng để tạo các control ToolStrip của riêng bạn (như được nêu trong mục kế tiếp, "Thêm bất kỳ Control vào một ToolStrip").

Tất cả các mục ToolStrip để bắt nguồn từ lớp ToolStripItem. Điều đó có nghĩa là tất cả chúng đều hỗ trợ một vài thuộc tính cơ bản (các thuộc tính quan trọng nhất là Text, Image và ImageAligh, chúng xác lập nội dung hiển thị). Tất cả các mục ToolStrip đều cung cấp một event Click mà bạn có thể sử dụng để phát hiện khi nào người dùng nhấp một nút trên thanh công cụ.

Chẳng hạn, nếu bạn muốn phản ứng một thao tác nhấp vào một mục ToolStrip mà bạn đã đặt tên là TestToolStripButton, bạn có thể sử dụng mã sau đây:

Ghi chú

Khi người dùng nhấp một nút trên ToolStrip, event Click của nút đó kích khởi. Điều này khác với control ToolBar thừa kế, vốn kích khởi một event Click chung bất kể nút nào đã được nhấp.

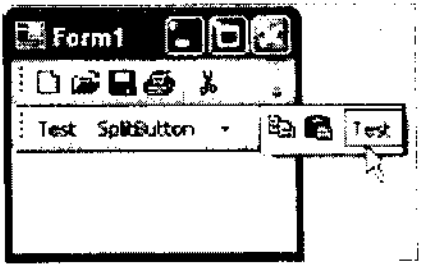
```
Private Sub TestToolStripButton_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles TestToolStripButton.Click

    MessageBox.Show("You clicked " & CType(sender, ToolStripItem).Name)
End Sub
```

Ngay sau khi bạn đã tạo một ToolStrip và đã thêm ít nhất một mục, bạn có thể sử dụng một kiểu định dạng có ý nghĩa. Sau đây chỉ là một vài tính năng ấn tượng mà ToolStrip cung cấp:

- Nó phù hợp với diện mạo thanh công cụ Office XP, với một nền gradient màu xanh dương, các kẹp định kích và sự theo dõi nóng (bật sáng một mục khi chuột di chuyển qua nó).
- Nó hỗ trợ chính xác các theme Windows XP. Điều đó có nghĩa là nếu bạn thay đổi kiểu phối màu sang Olive Green hay Silver, mọi control ToolStrip sẽ tự động cập nhật chính chúng, làm cho ứng dụng của bạn có vẻ hài hòa với cảnh.
- Nó cho phép người dùng tạo tùy biến. Nếu bạn kích hoạt thuộc tính ToolStrip.AllowReorder, người dùng có thể sắp xếp

lại thứ tự của các nút trong một ToolStrip bằng cách nhấn giữ phím Alt và rê các mục từ nơi này sang nơi khác, hoặc rê một nút từ một ToolStrip này sang một ToolStrip khác.

- Nó hỗ trợ các menu tràn qua. Nếu bạn kích hoạt tính năng này (bằng cách xác lập ToolStrip.CanOverflow sang true) và thu hẹp cửa sổ sao cho toàn bộ ToolStrip không còn nằm vừa trong đó nữa, một menu xổ xuống đặc biệt xuất hiện ở bên phải với tất cả các nút bổ sung, như minh họa ở hình 3.3.



Hình 3.3. Một menu tràn qua

Trong ví dụ trên, ToolStrip được đặt cố định tại chỗ. Nếu bạn muốn, bạn có thể cho người dùng khả năng rê một ToolStrip, hoặc để neo nó ở một nơi khác hoặc để sắp xếp lại những thành phần xuất hiện cùng nhau. Để làm được điều này, bạn cần thêm một ToolStripContainer vào form của bạn, vốn hiển thị ở dạng một hộp với nền gradient màu xanh dương (giống như nền của ToolStrip). Mặc dù bạn có thể sử dụng hơn một ToolStripContainer, nhưng thông thường bạn sẽ chỉ sử dụng một và neo nó để lấp đầy hoàn toàn hoặc một phần của cửa sổ.

Ghi chú

Để thêm một ToolStripContainer và đặt một ToolStrip trong nó trong chỉ một bước, nhấp thẻ thông minh ToolStrip rồi nhấp liên kết "Embed in ToolStripContainer".

ToolStripContainer thường bao bọc bốn đối tượng ToolStripPanel, một đối tượng cho mỗi phía. Các đối tượng này được hiển thị qua các thuộc tính như ToolStripContainer.LeftToolStripPanel, ToolStripContainer.TopToolStripPanel,... Mỗi panel có thể chứa vô số đối tượng ToolStrip, các đối tượng này sau đó được neo vào phía thích hợp. Phần hướng dẫn là ngay sau khi bạn đặt một ToolStrip trong một ToolStripContainer, người dùng sẽ có thể rê một ToolStrip về panel của nó vào lúc chạy. Thậm chí người dùng cũng có thể rê một ToolStrip từ ToolStripPanel này đến một ToolStripPanel kia để thay đổi phía mà nó được neo trên đó (hoặc sang một ToolStripContainer hoàn toàn riêng biệt trong cùng cửa sổ).

Thủ thuật

Nếu bạn muốn ngăn chặn người dùng neo ToolStrip vào phía bên trái của container, hãy xác lập thuộc tính ToolStripContainer.LeftToolStripPanelVisible. Bạn cũng có thể sử dụng các thuộc tính tương tự để ngăn chặn neo vào các phía phải, đỉnh hay đáy.

3.1.2. Thế còn...

... việc cập nhật phần còn lại của giao diện để có diện mạo đẹp như ToolStrip thì sao? .NET 2.0 thật sự cung cấp bốn control có diện mạo hiện đại của Windows XP và hỗ trợ chủ đề Windows XP. Chúng là ToolStrip, StatusStrip, MenuStrip và ContextMenuStrip, chúng thay thế ToolBar, StatusBar, MainMenu, và ContextMenu. Bạn có thể nhanh chóng làm mới giao diện của ứng dụng bằng cách cập nhật các standby cũ này vào các control mới.

Thủ thuật

Trong Visual Studio 2005, bạn sẽ không nhìn thấy các control thừa kế như ToolBar và StatusBar, bởi vì chúng bị loại ra khỏi hộp công cụ theo mặc định. Nếu bạn muốn sử dụng chúng, hãy nhấp phải vào hộp công cụ (toolbox), chọn Choose Items, và chọn các control này từ danh sách.

3.1.3. Tìm hiểu thêm?

Để tìm hiểu thêm về các lớp ToolStrip, bạn có thể xem thêm các mục sau đây trong chương này:

- "Thêm bất kỳ Control vào một ToolStrip", mục này giải thích cách thêm các control khác vào một ToolStrip.
- "Thêm các biểu tượng vào Menu của bạn", mục này giải thích cách sử dụng control MenuStrip mới.

3.2 Thêm bất kỳ Control vào một ToolStrip

ToolStrip hỗ trợ nhiều lớp ToolStripItem, cho phép bạn thêm mọi thứ từ các nút và các menu xổ xuống đến các hộp text và các nhãn. Tuy nhiên, trong một số trường hợp bạn có thể muốn vượt xa các tùy chọn chuẩn và sử dụng các control .NET khác, hoặc đặt các control tùy biến của riêng bạn trong ToolStrip. Để làm được điều này, bạn cần sử dụng ToolStripControlHost.

Ghi chú

Bạn muốn mở rộng một ToolStrip với một control tùy biến? Nhờ vào ToolStripControlHost, bạn có thể thêm vào hầu như bất kỳ điều gì.

3.2.1. Bạn làm điều đó như thế nào?

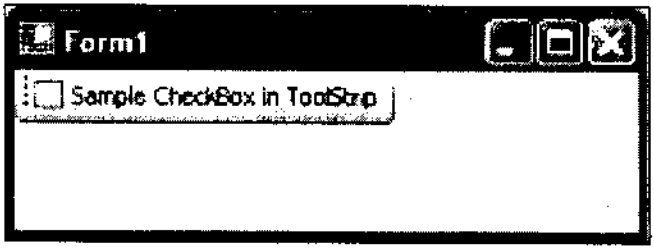
Không có cách nào để thêm các control chuẩn trực tiếp vào ToolStrip, bởi vì ToolStrip chỉ hỗ trợ các lớp phái sinh từ ToolStripItem. Bạn có thể tạo một lớp phái sinh từ ToolStripItem để thực thi một thành phần ToolStrip tùy biến, nhưng phương pháp này khá phức tạp và gây nhàm chán. Một phương pháp đơn giản hơn nhiều là sử dụng ToolStripControlHost, vốn có thể bao bọc hầu như bất kỳ control .NET.

Để sử dụng ToolStripControlHost với một control khác ToolStripItem, bạn chỉ cần chuyển đối tượng control ở dạng một đối hàm tạo (constructor) khi bạn tạo ToolStripControlHost. Sau đó, thêm đối tượng ToolStripControlHost vào ToolStrip. Bạn có thể sử dụng mã trong ví dụ 3.1 để thêm một control CheckBox vào tập hợp ToolStrip.Items. Hình 3.4 minh họa kết quả.

Ví dụ 3.1. Thêm một control Checkbox vào một tập hợp ToolStrip.Items

```
' Create a CheckBox.
Dim CheckStrip As New CheckBox( )

' Set the CheckBox so it takes the size of its text.
CheckStrip.AutoSize = True
CheckStrip.Text = "Sample CheckBox in ToolStrip"

' Make sure the CheckbBox is transparent (so the
' ToolStrip gradient background shows through).
CheckStrip.BackColor = Color.FromArgb(0, 255, 0, 0)

' Create the ToolStripControlHost that wraps the CheckBox.
Dim CheckStripHost As New ToolStripControlHost(CheckStrip)

' Set the ToolStripControlHost to take the full width
' of the control it wraps.
CheckStripHost.AutoSize = True
```

```
' Add the wrapped CheckBox.
ToolStrip1.Items.Add(CheckStripHost)
```



Hình 3.4. Một ToolStrip với một CheckBox

3.2.2. Tạo tiến trình ToolStripControlHost

Nếu bạn đang sử dụng ToolStripControlHost để điều khiển một control khác, bạn có thể muốn thêm các thuộc tính vào ToolStripControlHost để hiển thị dữ liệu từ control được điều khiển. Chẳng hạn, bạn có thể thêm một thuộc tính Checked vào ToolStripControlHost đã được sử dụng trong ví dụ này để bạn có thể dễ dàng xác lập hay truy xuất trạng thái đã kiểm tra (checked) của control CheckBox được bao bọc. Để sử dụng kỹ thuật này, bạn cần tạo một lớp tiến trình phái sinh từ ToolStripControlHost.

3.3. Thêm các biểu tượng vào Menu của bạn

Các ứng dụng Windows đã dần dần được cải tiến kể từ lần đầu tiên Windows XP và Office XP xuất hiện. Ngày nay, nhiều ứng dụng Windows hiện đại sử dụng một menu đã được tinh chỉnh với một lề bóng màu xanh dương ở phía bên trái, và một biểu tượng tùy ý cho mỗi lệnh menu. (Để thấy diện mạo này, xem minh họa ở hình 3.5).

Ghi chú

Bạn cũng có thể làm cho các menu của bạn trở nên nổi bật hơn với các ảnh thumbnail.

Nếu bạn muốn tạo một menu có vẻ sáng bóng với diện mạo này trong .NET 1.0 hay 1.1, bạn cần tự mình vẽ nó nhờ dùng mã GDI+. Tuy có sẵn nhiều ví dụ hay của kỹ thuật này trên Internet, nhưng nó hơi bề bộn. Trong .NET 2.0, điều này đã được cải tiến khá nhiều. Cho dù các control MainMenu và ContextMenu ban đầu không thay đổi nhưng hai control mới MenuStrip và ContextMenuStrip có chức năng như nhau ngoại trừ mô phỏng menu với diện mạo Office XP mới.

3.3.1. Bạn làm điều đó như thế nào?

Các lớp MenuStrip và ContextMenuStrip tăng cường mọi nỗ lực khi tạo lớp ToolStrip. Về cơ bản, một MenuStrip là một container đặc biệt cho các đối tượng ToolStripItem. Thuộc tính MenuStrip.Items chứa một tập hợp các tiêu đề menu ở cấp độ cao nhất (như File, Edit, View và Help), mỗi tiêu đề được biểu thị bởi một đối tượng ToolStripMenuItem. Mỗi ToolStripMenuItem có một DropDownItemsProperty, vốn hiển thị một tập hợp khác của các đối tượng ToolStripMenuItem, một dành cho mỗi mục menu được chứa.

Ví dụ 3.2 minh họa mã tạo menu Windows File quen thuộc.

Ví dụ 3.2. Tạo một menu Windows File

```
' Add the top-level items to the menu.
MenuStrip1.Items.AddRange(New ToolStripItem( ) _
  {fileToolStripMenuItem})

' Set the text for the File menu, and set "F" as the
' quick access key (so that Alt+F will open the menu.)
fileToolStripMenuItem.Text = "&File"

' Add the child items to the File menu.
fileToolStripMenuItem.DropDownItems.AddRange(New ToolStripItem( ) _
  {newToolStripMenuItem, openToolStripMenuItem, _
  toolStripSeparator, saveToolStripMenuItem, _
  saveAsToolStripMenuItem, toolStripSeparator1, _
  printToolStripMenuItem, printPreviewToolStripMenuItem, _
  toolStripSeparator2, exitToolStripMenuItem})

' Configure the File child items.
' Set the text and shortcut key for the New menu option.
newToolStripMenuItem.ShortcutKeys = CType((Keys.Control Or Keys.N), Keys)
newToolStripMenuItem.Text = "&New"

' Set the text and shortcut key for the Open menu option.
openToolStripMenuItem.ShortcutKeys = CType((Keys.Control Or Keys.O), Keys)
openToolStripMenuItem.Text = "&Open"
' (Code for configuring other omitted menu items.)
```

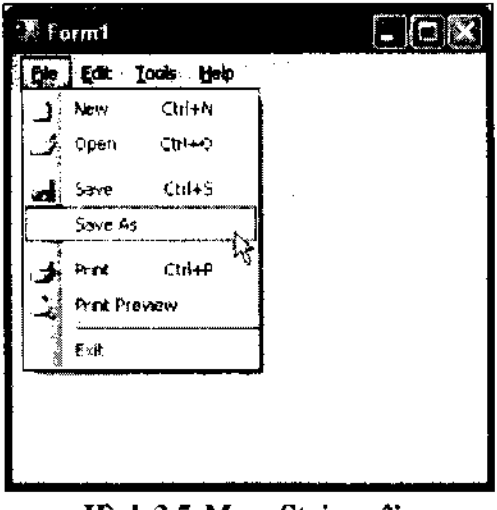
Thông thường, bạn sẽ không nhập thông tin này bằng tay, nó là thành phần của mã trình thiết kế (designer) mà Visual Studio tự động tạo khi bạn xác lập các thuộc tính trong cửa sổ Properties. Tuy nhiên, nó cho bạn thấy cách hoạt động của menu và bạn cần làm gì nếu bạn muốn thêm động các mục mới vào lúc chạy.

Như bạn đã thấy trong ví dụ 3.2, cấu trúc của một control MenuStrip giống như cấu trúc tiền thân của nó, control MainMenu với các đối tượng menu chứa các đối tượng menu khác. Điểm khác biệt duy nhất là ở loại đối tượng được sử dụng để biểu thị các mục menu (bây giờ nó là ToolStripMenuItem thay vì MenuItem) và tên của thuộc tính được sử dụng để chứa tập hợp các mục menu được chứa (ToolStripMenuItem.DropDownItems thay vì MenuItem.ChildItems).

Để tận dụng những lợi ích thật sự của ToolStripMenuItem mới, bạn cần sử dụng một thuộc tính mà nó chưa có sẵn với các đối tượng MenuItem bình thường: thuộc tính Image, thuộc tính này xác lập biểu tượng thumbnail xuất hiện bên trong lề menu.

```
newToolStripMenuItem.Image = CType( _
    resources.GetObject("newToolStripMenuItem.Image"), _
    System.Drawing.Image)
```

Hình 3.5 minh họa menu File chuẩn.



Hình 3.5. MenuStrip mới

Thông thường, bạn sẽ tải tất cả các hình ảnh của bạn nhờ dùng Visual Studio Properties Windows vào lúc thiết kế. Trong trường hợp đó, chúng sẽ được nhúng ở dạng một tài nguyên bên trong hợp ngữ của bạn. Một tùy chọn khác là tải chúng vào một ImageList và sau đó xác lập

ImageKey hoặc IndexProperties của ToolStripMenuItem để chỉ vào một hình ảnh trong ImageList.

Ghi chú

Để tạo nhanh một sườn menu cơ bản (bao gồm các lệnh menu chuẩn cho menu File, Edit, Tools, và Help), nhấp thẻ thông minh MenuStrip và chọn Insert Standard Items.

3.3.2. Thế còn...

... việc tô vẽ menu ngay từ đầu thì sao? Hy vọng bạn sẽ không cần thực hiện điều này. ToolStripMenuItem cho bạn nhiều sự linh hoạt hơn lớp MenuItem ban đầu, bạn không những có thể chèn các hình ảnh mà còn chọn một font không chuẩn bằng cách xác lập các thuộc tính ToolStripMenuItem.Font. Sau đây là một ví dụ:

```
fileToolStripMenuItem.Font = New Font("Verdana", 10, FontStyle.Regular)
```

Kỹ thuật này hữu dụng khi bạn muốn hiển thị một danh sách các font trong một loại ứng dụng hiệu chỉnh tài liệu nào đó và bạn muốn mô phỏng các tên font trong các mặt chữ tương ứng của chúng trong menu.

Nếu bạn cần thực hiện những thay đổi phức tạp hơn cho cách menu được vẽ, bạn sẽ cần sử dụng một bộ mô phỏng (renderer) khác. MenuStrip, giống như các control "strip", cung cấp một RenderMode và một thuộc tính Renderer. Thuộc tính RenderMode cho phép bạn sử dụng một trong các renderer cài sẵn bằng cách chọn một trị từ phép liệt kê ToolStripRenderMode (chẳng hạn như Professional, System, và Custom). Nếu bạn muốn sử dụng một renderer của riêng bạn, chọn Custom và sau đó cung cấp một đối tượng renderer mới trong thuộc tính Renderer. Renderer này có thể là một thể hiện của lớp thuộc nhóm thứ ba hoặc là một thể hiện của lớp mà bạn đã tạo (cũng phải sinh ToolStripRenderer và bỏ qua các phương thức để cung cấp logic vẽ chuyên biệt của bạn).

3.4. Đặt Web trong một cửa sổ

Không thiếu lý do khiến bạn muốn hợp nhất một cửa sổ trang web vào ứng dụng của mình. Có lẽ bạn muốn trưng bày web site công ty của mình, tạo một trình duyệt tùy biến, hay hiển thị tài liệu sản phẩm HTML. Trong .NET 1.0 và .NET 1.1, bạn có thể sử dụng một cửa sổ trình duyệt web thông qua COM interop, nhưng có nhiều tính năng kỳ lạ hay thiếu. Control WebBrowser mới trong .NET 2.0 giải quyết các vấn đề này bằng sự hợp nhất web một cách dễ dàng, hỗ trợ việc in và lưu các tài liệu, và khả năng ngăn chặn người dùng định hướng sai web site.

Ghi chú

Control WebBrowser được quản lý mới của .NET cho phép bạn trưng bày một trang HTML hoặc cho phép một người dùng trình duyệt một web site từ bên trong ứng dụng Windows của bạn mà không gặp phải bất kỳ vấn đề rắc rối nào.

3.4.1. Bạn làm điều đó như thế nào?

Control System.Windows.Forms.WebBrowser bao bọc một cửa sổ Internet Explorer. Bạn có thể thả control WebBrowser vào bất kỳ Windows form trực tiếp từ hộp công cụ Visual Studio .NET.

Để chỉ dẫn WebBrowser trưng bày hay hiển thị một trang, bạn chỉ cần xác lập thuộc tính Url sang trang web đích. Mọi sự định hướng trong WebBrowser đều không đồng bộ, nghĩa là mã của bạn tiếp tục chạy trong trang đang tải xuống. Để kiểm tra xem trang đã hoàn chỉnh hay chưa, hãy xác nhận thuộc tính ReadyState là Completed hay tốt hơn là nó phản ứng một event WebBrowser.

Ghi chú

Control WebBrowser hỗ trợ mọi thứ mà IE thực hiện, bao gồm JavaScript, các control ActiveX và các plug-in.

Các event WebBrowser được xếp theo thứ tự sau:

Ghi chú

WebBrowser cung cấp các phương thức sao lặp các hàm trình duyệt mà tất cả những người truy cập web đều quen thuộc, chẳng hạn như Stop( ), Refresh( ), GoBack( ), GoForward( ), GoHome( ), GoSearch( ), Print( ), ShowPrintDialog( ), và ShowSave-AsDialog( ).

1. Navigating kích khởi khi bạn xác lập một Url mới hay người dùng nhấp một liên kết. Đây là cơ hội để bạn hủy sự định hướng trước khi có bất kỳ điều gì xảy ra.
2. Navigated kích khởi sau Navigating, ngay trước khi trình duyệt web bắt đầu tải xuống trang.
3. Event ProgressChanged kích khởi định kỳ trong suốt tiến trình tải xuống (download) và cung cấp cho bạn thông tin về bao nhiêu byte đã được tải xuống và tổng mong đợi là bao nhiêu.
4. DocumentCompleted kích khởi khi trang đã được tải hoàn toàn. Đây là cơ hội để bạn xử lý trang.

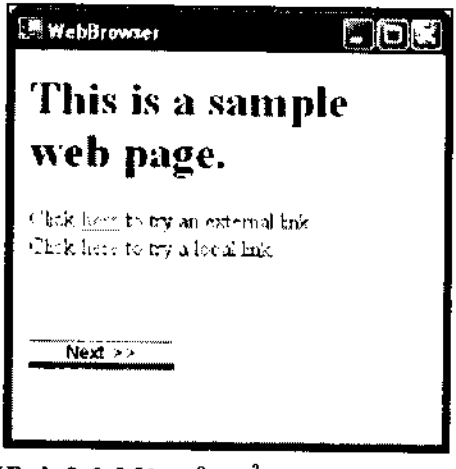
Hình 3.3 minh họa mã xử lý event cho một form, WebForm, vốn điều khiển một WebBrowser cùng với một thanh trạng thái đơn giản và thanh tiến độ. WebBrowser hiển thị một file HTML cục bộ (chú ý cách URL bắt đầu với file:///, không phải http://) và bảo đảm bất kỳ liên kết web ngoài được mở trong các cửa sổ Internet Explorer độc lập.

Ví dụ 3.3. Tạo một cửa sổ trình duyệt cơ bản

```
Public Class WebForm

    Private Sub WebForm_Load(ByVal sender As Object, ByVal e As EventArgs) _
      Handles MyBase.Load
        ' Prevent the user from dragging and dropping links onto this browser.
        Browser.AllowWebBrowserDrop = False

        ' Go to the local documentation page.
        Browser.Url = new Uri("file:///" & _
          My.Application.StartupPath & "\Doc.html")
    End Sub

    Private Sub Browser_Navigating(ByVal sender As Object, _
      ByVal e As WebBrowserNavigatingEventArgs) Handles Browser.Navigating
        If Not e.Url.IsFile Then
            ' Don't resolve this external link.
            ' Instead, use the Navigate( ) method to open a
            ' standalone IE window.
            e.Cancel = True
            Browser.Navigate(e.Url, True)
        End If
    End Sub

    Private Sub Browser_Navigated(ByVal sender As Object, _
      ByVal e As WebBrowserNavigatedEventArgs) Handles Browser.Navigated
        ' Show the progress bar.
        Progress.Visible = True
    End Sub

    Private Sub Browser_ProgressChanged(ByVal sender As Object, _
```

```
    ByVal e As WebBrowserProgressChangedEventArgs) _
    Handles Browser.ProgressChanged
      ' Update the progress bar.
      Progress.Maximum = e.MaximumProgress
      Progress.Value = e.CurrentProgress
  End Sub

  Private Sub Browser_DocumentCompleted(ByVal sender As Object, _
    ByVal e As WebBrowserDocumentCompletedEventArgs) _
    Handles Browser.DocumentCompleted
      ' Hide the progress bar.
      Progress.Visible = False
  End Sub

  Private Sub Browser_StatusTextChanged(ByVal sender As Object, _
    ByVal e As EventArgs) Handles Browser.StatusTextChanged
      ' Display the text that IE would ordinarily show
      ' in the status bar.
      Status.Text = Browser.StatusText
  End Sub

End Class
```

Hình 3.6 minh họa form với cửa sổ webBrowser đã tùy biến của nó. Cửa sổ này cũng có một StatusStrip để hiển thị trạng thái và một chỉ báo tiến độ khi các trang đang được tải.



Hình 3.6. Một cửa sổ web được nhúng

Ghi chú

Cửa sổ WebBrowser ở dạng cơ bản nhất và không có chứa một thanh công cụ, thanh địa chỉ hay thanh trạng thái (tuy bạn có thể thêm các control vào form của bạn).

3.4.2. Các thủ thuật truy cập web khác

WebBrowser cung cấp cho bạn hầu như mọi thế mạnh của IE để bạn có thể sử dụng trong các ứng dụng của mình. Sau đây là một vài thủ thuật khác mà bạn có thể muốn thử:

- Thay vì xác lập thuộc tính Url, hãy gọi phương thức Navigate ( ), vốn có hai overload hữu dụng. Overload đầu tiên (được minh họa trong ví dụ trước) cho phép bạn mở một cửa sổ trình duyệt độc lập. Overload thứ hai cho phép bạn tải một tài liệu vào một khung nhất định trong trang hiện hành.
- Thay vì sử dụng các URL, bạn có thể tải một tài liệu HTML trực tiếp từ một tài nguyên khác, sử dụng thuộc tính DocumentStream hay DocumentText. DocumentStream chấp nhận một tham chiếu đến bất kỳ đối tượng Stream, trong khi DocumentText chấp nhận một chuỗi có chứa dữ liệu HTML.
- Ngay sau khi bạn đã tải một tài liệu, bạn có thể khai thác nó nhờ dùng mô hình tài liệu HTML được cài sẵn trong .NET. Điểm nổi bật là thuộc tính Document, thuộc tính này trả về một đối tượng HtmlDocument vốn tạo mô hình cho tài liệu hiện hành, bao gồm cả các thẻ gán và nội dung của nó.
- Bạn có thể hướng WebBrowser đến một thư mục để cung cấp cho người dùng khả năng trình duyệt file nhanh. Tuy nhiên, lưu ý rằng bạn sẽ không thể ngăn chặn họ sao chép, di chuyển hay xóa các file.

3.5. Hợp chuẩn nhập liệu trong khi người dùng gõ nhập

Cả Visual Basic 6 và Access đều cung cấp cho các nhà phát triển các control hiệu chỉnh được tạo mặt nạ (masked editing control): text input control tự động định dạng dữ liệu nhập khi bạn gõ dựa vào một mặt nạ cụ thể. Chẳng hạn, nếu bạn gõ 1234567890 vào một input control được tạo mặt nạ vốn sử dụng một mặt nạ số điện thoại, số này sẽ được hiển thị ở dạng chuỗi (123) 456-7890.

Ghi chú

Các lập trình viên VB 6 đã quen với control ActiveX MaskedEdit bị thất vọng khi nhận thấy .NET không có phần thay thế. Trong .NET 2.0, MaskedTextBox mới đã lấp khoảng hở này.

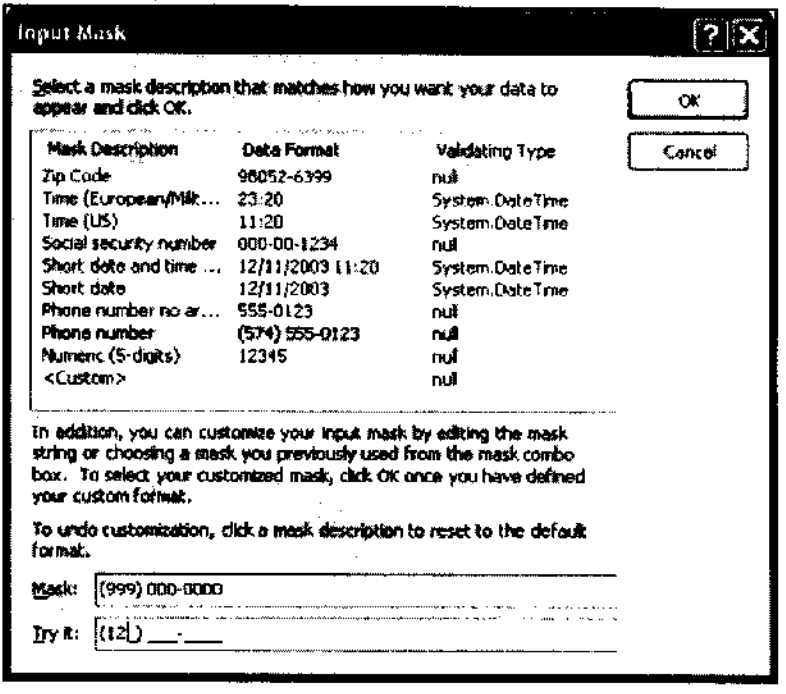
Các input control được tạo mặt nạ không chỉ cải tiến sự trình bày các trị nhất định mà còn ngăn chặn lỗi. Việc chọn đúng mặt nạ sẽ bảo đảm các ký tự nhất định bị loại ra ngay tức thì (ví dụ, một mặt nạ số điện thoại sẽ không chấp nhận các mẫu tự). Các input control được tạo mặt nạ cũng giúp tránh các lỗi, vốn xảy ra khi có trên một cách trình bày cùng một thông tin. Chẳng hạn, với mặt nạ số điện thoại (telephone number mask), người dùng sẽ lập tức nhận ra rằng mã vùng là bắt buộc, cho dù bạn không giải thích cụ thể yêu cầu này.

3.5.1. Bạn làm điều đó như thế nào?

.NET 2.0 có một control mới được gọi là MaskedTextBox vốn mở rộng control TextBox. Ngay sau khi bạn đã thêm một MaskedTextBox vào một form, bạn có thể thiết lập mặt nạ bằng hai cách:

- Bạn có thể chọn một trong các mặt nạ đã được tạo sẵn.
- Bạn có thể định nghĩa mặt nạ riêng của mình.

Để thiết lập một mặt nạ, bạn nhấp vào thẻ thông minh MaskedTextBox và chọn Set Mask. Hộp thoại Input Mask xuất hiện, với một danh sách các mặt nạ thường dùng, bao gồm các mặt nạ dành cho các số điện thoại, các mã vùng, ngày tháng,... Khi bạn chọn một mặt nạ từ danh sách, mặt nạ này sẽ được hiển thị trong hộp text Mask. Bây giờ bạn có thể tùy biến mặt nạ này. Bạn cũng có thể thử mặt nạ bằng cách sử dụng hộp text Try It, như minh họa ở hình 3.7.



Hình 3.7. Chọn một mặt nạ cho MaskedTextBox

Ghi chú

Nhờ vào các kỳ quan của COM Interop, control VB 6 MaskedEdit vẫn được sử dụng trong .NET. Tuy nhiên, control .NET MaskedTextBox cải tiến dựa trên nhiều hạn chế trong control MaskedEdit, vì vậy nó vẫn tốt hơn.

Mặt nạ mà bạn chọn sẽ được chứa trong thuộc tính MaskTextBox.Mask. Ngay sau khi bạn đã chọn một mặt nạ (mask), nó sẽ được áp dụng bất kỳ khi nào người dùng gõ vào MaskedTextBox. Nếu bạn muốn phản hồi lỗi của người dùng (như các ký tự không đúng) để cung cấp thêm thông tin, bạn có thể phản hồi event MaskInputRejected.

Nếu bạn muốn tạo một mặt nạ tùy biến, bạn cần tìm hiểu thêm về cách tạo mặt nạ. Về cơ bản, một mặt nạ được tạo từ hai loại ký tự: placeholder, chỉ định nơi người dùng phải cung cấp một ký tự; và trực kiện, được sử dụng để định dạng trị. Chẳng hạn, trong mặt nạ số điện thoại (999)-000-000, các dấu nối và các dấu ngoặc là những trực kiện. Các ký tự này luôn tồn tại và không thể bị xóa, chỉnh sửa hay di chuyển bởi người dùng. Số 0 là một placeholder tiêu biểu cho bất kỳ ký tự số, trong khi số 9 là một placeholder tiêu biểu cho một ký tự số tùy ý.

Bảng 3.1 liệt kê và giải thích tất cả các ký tự bạn có thể sử dụng để tạo một mặt nạ. Bạn có thể sử dụng bảng này để tham chiếu khi tạo các mặt nạ riêng của bạn.

Bảng 3.1. Các ký tự mặt nạ

Ký tự	Mô tả
0	Chữ số bắt buộc (0-9)
9	Chữ số tùy ý hay khoảng cách. Nếu được để trống, một khoảng cách được chèn vào tự động.
#	Chữ số tùy ý, khoảng cách hay dấu công/trừ. Nếu được để trống, một khoảng cách được chèn vào tự động.
L	Mẫu tự ASCII bắt buộc (a-z hay A-Z).
?	Mẫu tự ASCII tùy ý.
&	Ký tự Unicode bắt buộc. Cho phép bất kỳ điều gì không phải là một khóa điều khiển, bao gồm dấu chấm câu và các ký hiệu.
C	Ký tự Unicode tùy ý.
A	Ký tự chữ - số bắt buộc (cho phép chữ hay số nhưng không chấp nhận dấu chấm câu hay các ký hiệu).
A	Ký tự chữ - số tùy ý.
.	Dấu thập phân.
,	Dấu hàng ngàn.
:	Dấu phân cách giờ.
$	Ký hiệu tiền tệ.
<	Mọi ký tự theo sau sẽ được chuyển đổi tự động thành chữ thường khi người dùng gõ chúng vào. (Không có cách nào để chuyển đổi phần kế tiếp của text trở lại chế độ nhập kết hợp giữa chữ hoa và thường ngay khi bạn sử dụng ký tự này).
>	Mọi ký tự theo sau sẽ được chuyển đổi tự động sang chữ hoa khi người dùng gõ chúng vào.
\	Thoát một ký tự được tạo mặt nạ, biến nó thành một trực kiện. Do vậy, nếu bạn sử

	dụng \ & nó được hiểu là một ký tự trực kiện, vốn sẽ được chèn trong hộp text.
Tất cả những ký tự khác	Tất cả những ký tự khác được xem là trực kiện và được hiển thị trong hộp text.

Sau cùng, có vài thuộc tính khác mà MaskedTextBox cung cấp (và bạn có thể muốn sử dụng). Chúng bao gồm:

BeepOnError

Nếu người dùng nhập một ký tự không hợp lệ và biết BeepOnError is TRue, thì MaskedTextBox sẽ phát ra tiếng báo lỗi chuẩn.

PromptChar

Khi hộp text trống, mọi trị bắt buộc được thay bằng một ký tự nhắc. Theo mặc định, ký tự nhắc là dấu underscore ( ), vì vậy một mặt nạ cho số điện thoại sẽ hiển thị (_ _ _) -_ _ _ - _ _ _ _ trong khi trống.

MaskCompleted

Trả về TRue nếu không có ký tự rỗng trong hộp text (nghĩa là người dùng đã nhập vào trị bắt buộc).

InputText

InputText trả về dữ liệu trong MaskedTextBox không có bất kỳ ký tự trực kiện nào. Chẳng hạn, trong một MaskedTextBox vốn cho phép người dùng nhập một số điện thoại, thuộc tính Text sẽ trả về số đã được định dạng đầy đủ, như (123) - 456 - 7890, trong khi InputText trả về chỉ nội dung số, tức là 1234567890.

3.5.2. Thế còn...

... việc sử dụng kỹ thuật hiệu chỉnh được tạo mặt nạ trong các input control khác thì sao? Điều đó thì có thể thực hiện được nhưng không dễ. MaskedTextBox dựa vào một lớp MaskedEditProvider đặc biệt trong namespace System.ComponentModel.

Để tạo một loại khác của control được tạo mặt nạ, bạn cần tạo một control tùy biến sử dụng MaskedEditProvider trong nội bộ. Khi control của bạn nhận được một thao tác nhấn phím, bạn cần xác định hành động đã thực hiện và chuyển nó sang MaskedEditProvider nhờ dùng các phương thức như Add ( ), Insert ( ), Remove ( ), và Replace ( ). Sau đó, bạn có thể truy xuất trị hiển thị mới bằng cách gọi MaskedEditProvider.ToDisplayString ( ), và làm mới control tùy biến một

cách thích hợp. Phần khó là xử lý mọi việc hiệu chỉnh này ở cấp độ thấp mà không gây ra nhấp nháy hay làm mất vị trí của người dùng trong chuỗi dữ liệu nhập. Để tìm hiểu thêm thông tin, bạn có thể xem ví dụ đầy đủ đi kèm với mã có thể được tải xuống trong dự án MaskedEditing.

3.6. Tạo các hộp Text Auto-Complete

Trong nhiều chỗ của hệ điều hành Windows, bạn sẽ tìm thấy các hộp AutoComplete. Các hộp text này đề nghị một hoặc nhiều trị khi bạn gõ.

Ghi chú

Với các tính năng auto-complete mới của .NET, bạn có thể tạo các hộp text thông minh có khả năng đề nghị các trị có thể có dựa vào mục nhập gần đây hay một danh sách mặc định.

Thông thường, các trị AutoComplete được lấy từ history gần đây. Chẳng hạn, khi bạn gõ một URL vào thanh địa chỉ của Internet Explorer, bạn sẽ thấy một danh sách bao gồm các URL bạn đã truy cập trước đây. Bây giờ với .NET 2.0, bạn có thể giữ các tính năng AutoComplete này với các danh sách tùy biến của riêng bạn hay một trong các danh sách do hệ điều hành bảo lưu.

3.6.1. Bạn làm điều đó như thế nào?

Cả hai control TextBox và ComboBox đều hỗ trợ tính năng AutoComplete trong .NET 2.0. Để sử dụng AutoComplete, trước tiên hãy xác lập thuộc tính AutoCompleteMode của control sang một trong các trị sau:

Append

Trong chế độ này, trị AutoComplete được chèn tự động vào control khi bạn gõ. Tuy nhiên, phần được thêm vào sẽ được chọn để phần mới sẽ được thay thế nếu bạn tiếp tục gõ. (Bạn cũng có thể nhấp delete để xóa nó).

Suggest

Đây là chế độ thân thiện nhất. Khi bạn gõ, một danh sách xổ xuống gồm các trị AutoComplete phù hợp xuất hiện ngay bên dưới control. Nếu một trong các mục này phù hợp với những gì bạn muốn, bạn có thể chọn nó.

SuggestAppend

Chế độ này kết hợp Append và Suggest. Giống như với Suggest,

một danh sách các trị phù hợp được hiển thị trong một danh sách xổ xuống. Tuy nhiên, trị phù hợp đầu tiên cũng được chèn vào control và được chọn.

Sau khi chọn loại AutoComplete, bạn cần chỉ định danh sách nào sẽ được sử dụng cho các phần đề nghị. Bạn làm điều này bằng cách xác lập thuộc tính AutoCompleteSource sang một trong các trị sau:

FileSystem

Bao gồm các đường dẫn file đã được nhập gần đây. Sử dụng FileSystemDirectories để đưa vào chỉ các đường dẫn thư mục.

HistoryList

Bao gồm các Url từ danh sách history của Internet Explorer.

RecentlyUsedList

Bao gồm mọi tài liệu trong "danh sách được sử dụng gần đây nhất" của người dùng, xuất hiện trong menu Start (phụ thuộc vào các xác lập hệ thống).

AllUrl

Bao gồm URL của tất cả các site mà người dùng hiện hành đã vào gần đây, dù chúng được gõ bằng tay bởi người dùng hay được liên kết từ một trang web.

AllSystemSources

Bao gồm toàn bộ danh sách URL và đường dẫn file.

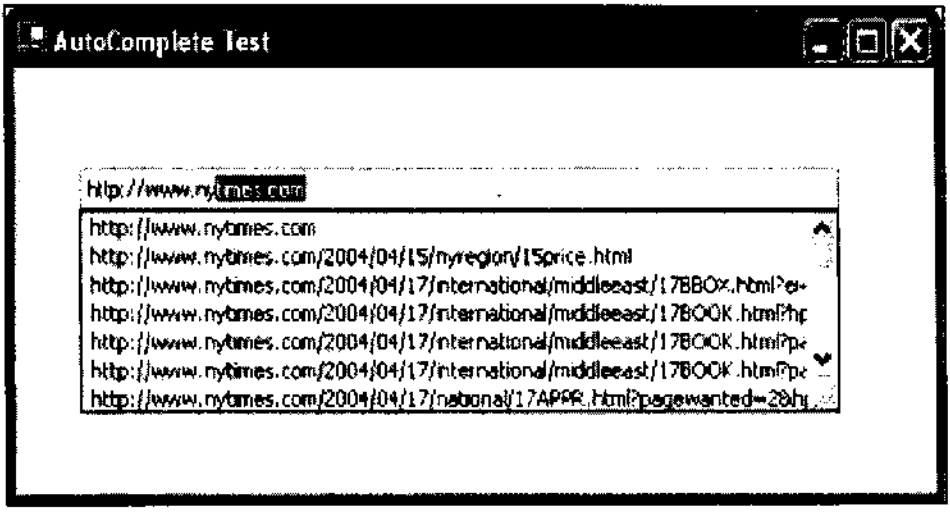
ListItems

Bao gồm các mục trong tập hợp ComboBox.Items. Lựa chọn này không hợp lệ với TextBox.

CustomSource

Bao gồm các mục trong tập hợp AutoCompleteCustomSource. Bạn cần tự mình thêm vào các mục này.

Hình 3.8 minh họa một hộp text AutoComplete nhờ dùng AutoSuggestAppend làm AutoCompleteMode và AllUrl làm AutoCompleteSource.



Hình 3.8. Một hộp text AutoComplete

Cả control TextBox lẫn ComboBox đều có cùng chức năng. Nếu bạn sử dụng AutoSuggest hay AutoSuggestAppend với một ComboBox, danh sách các mục tương kết được hiển thị trong một danh sách bên dưới control. Tuy nhiên, không được nhầm lẫn danh sách này với danh sách các mục mà bạn đã thêm vào thuộc tính ComboBox.Items. Khi bạn nhấp mũi tên xổ xuống cho ComboBox, bạn sẽ thấy danh sách các mục của bạn, không phải danh sách các đề nghị AutoComplete. Cả hai danh sách đều hoàn toàn riêng biệt và không có cách lập trình nào để bạn tương tác với danh sách AutoComplete. Ngoại lệ duy nhất là nếu bạn tạo một ComboBox với một AutoCompleteSource của CustomSource hay ListItems.

3.6.2. Thế còn...

... việc sử dụng AutoComplete trong các control khác thì sao? Thật không may, không có cách quản lý nào để thực hiện nó trong .NET. Tuy nhiên, bạn có thể truy xuất thông tin mà bạn cần trực tiếp từ registry. Chẳng hạn, nếu bạn tìm trong mục Software\Microsoft\Internet Explorer\TypedURLs của khóa registry HKEY_CURRENT_USER, bạn sẽ thấy danh sách các URL được gõ vào gần đây. Để truy xuất các mục này bằng lập trình, hãy tham khảo các lớp như RegistryKey trong namespace Microsoft.Win32.

3.7. Mở một âm thanh hệ thống Windows

Hệ điều hành Windows cảnh báo người dùng về những sự kiện hệ thống bằng cách biểu diễn chúng sang dạng âm thanh được ghi trong

các file audio cụ thể. Vấn đề là các file này được lưu trữ ở những nơi khác nhau trên các máy tính khác nhau. Trong .NET 1.0 và 1.1, không có cách dễ dàng nào để tìm các âm thanh hệ thống mặc định và mở chúng trong ứng dụng của bạn. Một lớp SystemSounds mới trong .NET 2.0 giải quyết vấn đề này, cho phép bạn mở các âm thanh phổ biến nhất bằng một dòng mã duy nhất.

Ghi chú

Bạn cần phát ra âm thanh không phổ biến lắm của Windows? Với lớp SystemSounds mới, các file audio này luôn có sẵn cho bạn.

3.7.1. Bạn làm điều đó như thế nào?

Lớp SystemSounds trong namespace System.Windows.Forms cung cấp năm thuộc tính chia sẻ. Mỗi thuộc tính này là một đối tượng SystemSound riêng biệt tiêu biểu cho một sự kiện hệ điều hành nhất định. Sau đây là danh sách đầy đủ:

- Asterisk
- Beep
- Exclamation
- Hand
- Question

Ngay sau khi bạn quyết định âm thanh nào sẽ được sử dụng, bạn chỉ cần gọi phương thức Play ( ) của nó để mở âm thanh. Sau đây là một ví dụ:

Ghi chú

Để cấu hình các file WAV nào được sử dụng cho mỗi âm thanh, bạn chọn biểu tượng Sounds and Audio Devices trong Control Panel.

```
SystemSounds.Beep.Play ( )
```

3.7.2. Mở các file WAV tùy ý

Lớp SystemSounds hoạt động tốt nhất nếu bạn chỉ cần một cách dễ dàng để thêm một âm thanh cho sự phản hồi đơn giản. Nếu bạn cần mở

một file audio do bạn lựa chọn, bạn cần sử dụng SoundPlayer, như được trình bày trong phần thực hành kế tiếp, "Mở Audio WAV đơn giản".

3.8. Mở Audio WAV đơn giản

Cả .NET 1.0 và .NET 1.1 đều không có cách nào được quản lý để mở audio. Nhược điểm này sau cùng đã được khắc phục trong .NET 2.0 với lớp SoundPlayer mới, lớp này cho phép bạn mở audio đồng bộ hay không đồng bộ.

Ghi chú

Nhờ sử dụng lớp SoundPlayer, bạn có thể mở các file WAV mà không cần phải vào Windows API.

3.8.1. Bạn làm điều đó như thế nào?

Bạn có thể thể hiện một đối tượng SoundPlayer bằng lập trình, hoặc bạn có thể thêm một đối tượng vào khay thành phần bằng cách rê nó từ hộp công cụ vào lúc thiết kế. Ngay khi bạn đã tạo SoundPlayer, bạn cần chỉ nó đến nội dung âm thanh mà bạn muốn mở. Bạn thực hiện điều này bằng cách xác lập một trong hai thuộc tính sau:

SoundLocation

Nếu bạn có một đường dẫn file hay URL chỉ đến một file WAV, chỉ định thông tin này trong thuộc tính SoundLocation.

Stream

Nếu bạn có một đối tượng dựa vào Stream có chứa nội dung audio WAV, hãy sử dụng thuộc tính Stream.

Ngay sau khi bạn đã xác lập thuộc tính Stream hay SoundLocation, bạn cần yêu cầu SoundPlayer thật sự tải dữ liệu audio bằng cách gọi phương thức Load ( ) hay LoadAsync ( ). Phương thức Load ( ) tạm dừng mã của bạn cho đến khi mọi audio được tải vào bộ nhớ. Mặt khác, LoadAsync ( ) thực hiện công việc của nó trên một luồng khác và kích khởi event LoadCompleted ngay khi nó hoàn tất và audio có sẵn. Thông thường, bạn sẽ sử dụng Load ( ) trừ khi bạn có một file audio cực kỳ lớn hoặc mất quá nhiều thời gian để đọc toàn bộ file audio (ví dụ, khi truy xuất audio qua một nối kết Internet hay mạng có tốc độ chậm).

Sau cùng, ngay khi audio đã có sẵn, bạn có thể gọi một trong các phương thức sau:

PlaySync( )

Tạm dừng mã của bạn cho đến khi hoàn tất playback audio.

Play( )

Phát audio trên một luồng khác, cho phép mã tiếp tục với các tác vụ khác và bảo đảm giao diện của ứng dụng vẫn phản hồi.

PlayLooping( )

Tương tự Play ( ), ngoại trừ nó tạo vòng lặp audio, lặp lại liên tục.

Để tạm ngưng playback không đồng bộ vào bất kỳ lúc nào, bạn chỉ cần gọi Stop ( )

Đoạn mã sau đây minh họa một ví dụ mở một âm thanh mẫu một cách đồng bộ:

```
Dim Player As New SoundPlayer( )
Player.SoundLocation = Application.StartupPath & "\mysound.wav"
Try
    Player.Load( )
    Player.PlaySync( )
Catch Err As Exception
    ' An error will occur here if the file can't be read
    ' or if it has the wrong format.
End Try
```

3.8.2. Các loại audio khác

Thật không may, SoundPlayer chỉ có thể mở dạng audio WAV. Nếu bạn muốn mở các loại multimedia khác, như các file MP3 hay WMA, bạn cần sử dụng một giải pháp khác và không có lớp nào được quản lý để giúp bạn thử điều này.

Sau đây là hai lựa chọn:

- Sử dụng COM Interop để truy cập thư viện Quartz, đây là thành phần của DirectX. Thư viện Quartz cho phép bạn mở bất kỳ kiểu file nào được hỗ trợ bởi Windows Media Player, bao gồm MP3, WMA, và các dạng video như MPED và AVI. Đối với một ví dụ trong mã C#, hãy xem dự án mẫu của Microsoft tại http://msdn.microsoft.com/library/en-s/csref/html/vcwlkcominteroppart1cclienttutorial.aso
- Sử dụng các thư viện DirectX 9.0 được quản lý. Bạn cần cài đặt client DirectX và SDK trên máy tính để đạt được điều này,

nhưng nó cho bạn nhiều chức năng, bao gồm cả khả năng mô phỏng các đồ họa 3D. Xem http://msdn.microsoft.com/library/en-us/directx9_m/directx/dx9intro.asp để tìm hiểu thêm.

3.9. Tạo một cửa sổ phân chia giống như Windows Explorer

.NET 1.0 đã cung cấp cho các nhà phát triển những công cụ mà họ cần để tạo các cửa sổ phân chia thuộc loại như được nhìn thấy trong Internet Explorer với control Splitter. Thật không may, việc tạo các cửa sổ này không phải lúc nào cũng dễ dàng, bởi vì nó thường đòi hỏi sự kết hợp giữa Splitter và ba control Panel, tất cả đều cần phải được neo theo thứ tự đúng. Nếu bạn cần phân chia một cửa sổ bằng nhiều cách, tác vụ này trở nên phức tạp hơn. .NET 2.0 đã đơn giản tiến trình này bằng control SplitContainer.

Ghi chú

Các cửa sổ bây giờ được phân chia dễ dàng hơn nhiều vì đã có control SplitContainer thay cho Splitter.

3.9.1. Bạn làm điều đó như thế nào?

Về cơ bản, control SplitContainer tiêu biểu cho hai ô được tách nhau bởi một vạch phân chia. Người dùng có thể rê vạch này sang phía này hoặc phía kia để thay đổi chiều rộng tương đối của mỗi ô. Để giúp báo hiệu tính có sẵn của chức năng này, con trỏ chuột chuyển đổi từ một biểu tượng mũi tên một đầu sang hai đầu khi người dùng di chuyển chuột qua vạch phân chia này.

Ghi chú

Một control SplitContainer thường được sử dụng khi nội dung trong hai ô có liên quan với nhau. Khi người dùng thực hiện chọn trong ô đầu tiên, nội dung trong ô thứ hai được làm mới.

Để tạo một giao diện đơn giản với SplitContainer, trước tiên bạn phải quyết định SplitContainer sẽ chiếm bao nhiêu phần màn hình. Chẳng hạn, nếu bạn cần dành riêng một chỗ phía dưới SplitContainer, hãy bắt đầu bằng cách neo một Panel vào đáy của form. Khi bạn thêm SplitContainer, thuộc tính Dock của nó sẽ tự động sang DockStyle.Fill để nó lấp đầy bất kỳ chỗ nào được chừa lại.

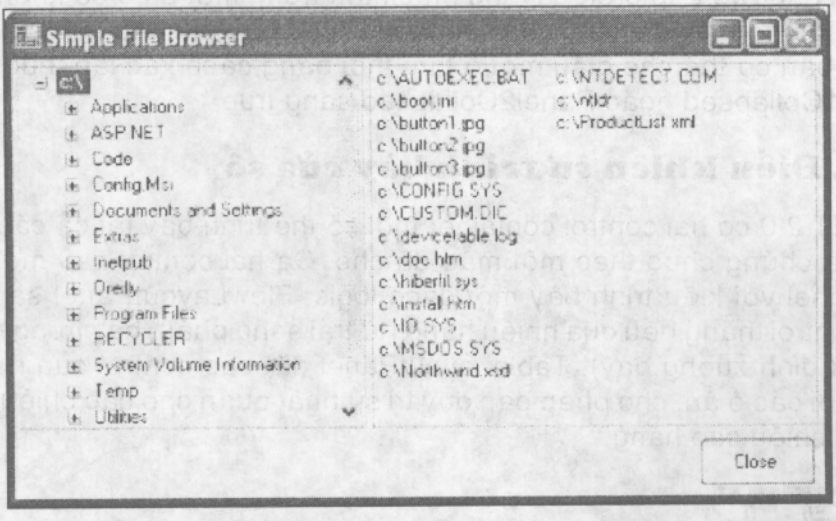
SplitContainer luôn bao gồm hai ô. Nếu bạn xác lập thuộc tính Orientation sang Orientation.Vertical (mặc định), vạch phân chia sẽ chạy từ

đỉnh xuống đáy, tạo ra các ô trái và phải. Tùy chọn khác là Orientation.Horizontal, vốn tạo các ô đỉnh và đáy với một vạch phân chia chạy từ trái sang phải ở giữa chúng.

Ngay sau khi bạn đã xác lập hướng thích hợp, bước kế tiếp là thêm các control vào mỗi phía của SplitContainer. Nếu bạn muốn có một control duy nhất ở mỗi phía, bạn chỉ cần rê control đó đến ô thích hợp trong SplitContainer và xác lập thuộc tính Dock của control sang DockStyle.Fill, để nó lấp đầy toàn bộ khoảng trống có sẵn giữa vạch phân chia và các mép của SplitContainer.

Nếu bạn cần thêm nhiều control trong cùng miền của SplitContainer, hãy bắt đầu bằng cách thêm một Panel và xác lập thuộc tính Dock sang DockStyle.Fill. Sau đó, bạn có thể neo các control khác bên trong Panel.

Ngay khi bạn đã cài đặt SplitContainer, bạn không cần viết bất kỳ mã nào để quản lý việc định kích cỡ control hay sự tương tác từ người dùng. Hình 3.9 minh họa một ví dụ. (Project SplitWindow hoàn chỉnh sẽ có sẵn với các mẫu có thể được tải xuống).



Hình 3.9. Một cửa sổ được phân chia theo chiều dọc

Ghi chú

Bạn cũng có thể xếp lồng một SplitContainer bên trong một SplitContainer khác. Điều này hữu dụng nhất nếu bạn đang sử dụng các hướng khác nhau (chẳng hạn, chia một cửa sổ thành các vùng trái và phải rồi sau đó chia vùng bên phải thành các ô đỉnh và đáy).

3.9.2. Thế còn...

... việc giới hạn cách một SplitContainer có thể được thay đổi kích cỡ thì sao? SplitContainer cung cấp nhiều thuộc tính đã được chỉnh sửa cho mục đích này. Chẳng hạn, bạn có thể xác lập các thuộc tính Panel1MinSize và Panel2MinSize với chiều rộng pixel tối thiểu của các ô thích hợp. Ngay sau khi bạn xác lập các thuộc tính này, người dùng sẽ không thể đối với vạch phân chia sang một vị trí mà nó sẽ thu hẹp ô dưới cỡ được phép tối thiểu của nó. Bạn cũng có thể ngăn chặn việc thay đổi lại kích cỡ bằng cách xác lập thuộc tính IsSplitterFixed sang False (trong trường hợp này bạn vẫn có thể điều chỉnh vị trí của vạch phân chia bằng cách chỉnh sửa thuộc tính SplitterDistance theo lập trình).

Ngoài ra, bạn có thể cấu hình cách mà SplitContainer xử lý khi toàn bộ form được thay đổi kích cỡ. Theo mặc định, các ô được định kích cỡ theo tỷ lệ phù hợp. Tuy nhiên, bạn có thể chỉ định một trong các ô là có cỡ không đổi bằng cách xác lập thuộc tính FixedPanel. Trong trường hợp này, ô đó sẽ không bị dửa đổi khi form được thay đổi kích cỡ. (Ví dụ, trong Windows Explorer, cây thư mục nằm trong một ô có cỡ không đổi và nó không thay đổi kích cỡ khi bạn mở rộng hay thu hẹp cửa sổ). Sau cùng, bạn có thể che giấu một ô tạm thời bằng cách xác lập thuộc tính Panel1Collapsed hoặc Panel2Collapsed sang true.

3.10. Điều khiển sự trình bày cửa sổ

.NET 2.0 có hai control container mới có thể trình bày tất cả các control mà chúng chứa theo một mẫu đã cho. Cả hai control này mở rộng lớp Panel với kiểu trình bày một cách logic. FlowLayoutPanel sắp xếp các control thẳng đều qua nhiều hàng (từ trái sang phải), hay trong nhiều cột (từ đỉnh xuống đáy). TableLayoutPanel đặt các control của nó vào một lưới các ô ẩn, cho phép bạn duy trì sự nhất quán cho các chiều rộng cột và chiều cao hàng.

Ghi chú

Các control trình bày .NET mới cung cấp cho bạn một cách để trình bày các control theo các dạng đã có sẵn một cách tự động, điều này giúp bạn tránh được nhiều tác vụ với các giao diện có thể cấu hình hay có tính động cao.

3.10.1. Bạn làm điều đó như thế nào?

Các control trình bày (layout) được sử dụng thường xuyên nhất trong hai tình huống sau đây:

- Bạn có một giao diện động vốn tạo một số thành phần của nó bằng lập trình. Nhờ sử dụng các control layout, bạn có thể sắp xếp một nhóm control một cách chặt chẽ mà không cần tính vị trí cho mỗi control (và sau đó xác lập thuộc tính Location cho phù hợp).
- Bạn có một giao diện cục bộ hóa phải điều chỉnh cho phù hợp với các ngôn ngữ khác vốn đòi hỏi nhiều chỗ khác nhau trên màn hình. Do vậy, khi text hiển thị thay đổi, các control cũng phải điều chỉnh kích cỡ của chúng. Trong trường hợp này, các control layout cũng giúp bạn bảo đảm rằng các control vẫn được sắp xếp chính xác ngay cả khi kích cỡ của chúng thay đổi.

Ví dụ 3.4 minh họa một phần thực thi của tình huống đầu tiên. Nó bắt đầu với một form có chứa một FlowLayoutPanel trống. FlowLayoutPanel này có BorderStyle được xác lập sang BorderStyle.Fixed3D vì vậy đường viền này là thấy được.

Không có control nào được thêm vào FlowLayoutPanel vào lúc thiết kế. Thay vào đó, một vài nút mới được thêm vào bằng lập trình khi một nút cmdGenerate được nhấp.

Ví dụ 3.4. Trình bày các nút bằng lập trình

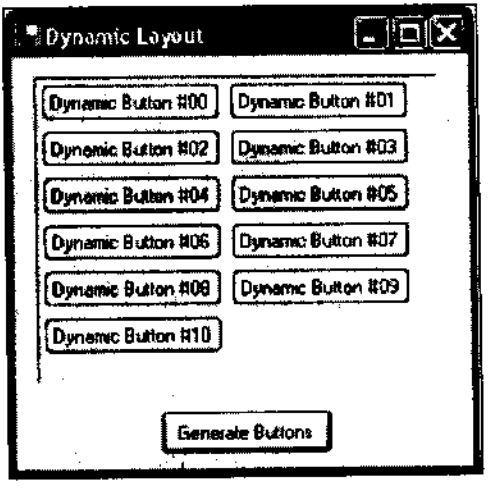
```
Private Sub Button1_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles Button1.Click
  For i As Integer = 0 To 10
    ' Create a new button.
    Dim Button As New Button
    Button.Text = "Dynamic Button #" & String.Format("{0:00}", i)

    ' Size the button the width of the text.
    Button.AutoSize = True

    ' Add the button to the layout panel.
    FlowLayoutPanel1.Controls.Add(Button)
  Next
End Sub
```

Lưu ý rằng mã không xác lập thuộc tính Location cho mỗi nút. Đó là bởi vì FlowLayoutPanel không sử dụng thông tin này. Thay vào đó, nó sẽ sắp xếp các nút theo thứ tự mà chúng được thêm vào, cách khoảng mỗi

nút từ trái sang phải và sau đó từ trên xuống dưới. (Để đảo ngược thứ tự này, hãy thay đổi thuộc tính FlowLayoutPanel.FlowDirection).

Có một mẫu thông tin mà FlowLayoutPanel sử dụng. Đó chính là thuộc tính Margin của mỗi control container. Điều này xác lập đường viền tối thiểu được yêu cầu giữa control này và control kế tiếp. Mã phía trên không thay đổi thuộc tính Button.Margin, bởi vì xác lập mặc định 3 pixel là hoàn toàn thích hợp.

Hình 3.10 minh họa diện mạo của các nút ngay khi chúng được thêm vào.



Hình 3.10. Trình bày các nút bằng lập trình

Ghi chú

Thật ra có bốn thành phần khác nhau của thuộc tính Margin: Margin.Left, Margin.Right, Margin.Top và Margin.Bottom. Bạn có thể xác lập các thành phần này một cách riêng lẻ để chỉ định các lề khác nhau cho control ở mỗi phía.

.NET cũng có một TableLayoutPanel. Panel này hoạt động giống như FlowLayoutPanel, trình bày các control một cách tự động, nhưng nó gán chúng theo các đường lưới cột và hàng ẩn. Chẳng hạn, nếu bạn có nhiều control đã được định kích cỡ khác nhau, bạn có thể sử dụng TableLayoutPanel để bảo đảm mỗi control được cách đều nhau trong một ô tưởng tượng.

3.10.2. Thế còn...

... các ví dụ trình bày cao cấp hơn thì sao? Có nhiều thứ hơn để bạn có thể thực hiện với việc sử dụng sáng tạo các control layout. Dĩ nhiên,

bạn chỉ nên sử dụng các control layout trong những tình huống đặc biệt mà các tính năng anchoring và docking của Windows Forms không đủ đáp ứng. Nếu bạn không có một giao diện động cao thì các trình quản lý layout có thể đưa ra nhiều sự phức tạp hơn bạn cần.

3.11. Điều khiển khi nào ứng dụng của bạn thoát

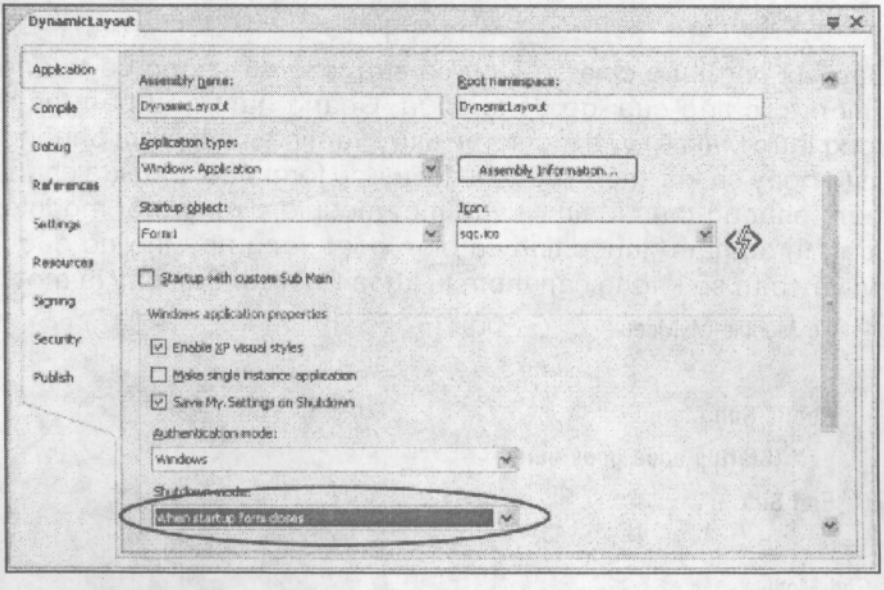
Trong Visual Studio 2005, một tính năng "Shutdown mode" mới cho phép bạn điều khiển khi nào ứng dụng của bạn sẽ kết thúc. Bạn có thể thoát ngay khi cửa sổ chính được đóng (cửa sổ được chỉ định là đối tượng khởi động), hoặc bạn có thể chờ đợi cho đến khi tất cả các cửa sổ ứng dụng đều được đóng. Và nếu không có cách nào trong số những lựa chọn này cung cấp cho bạn điều mà bạn mong muốn, bạn có thể nhận được sự điều khiển hoàn chỉnh với lớp Application.

Ghi chú

Trong .NET 2.0, việc chỉ định khi nào ứng dụng Windows của bạn sẽ thoát là dễ dàng hơn bao giờ hết.

3.11.1. Bạn làm điều đó như thế nào?

Trong Visual Studio, hãy nhấp double vào mục My Project trong Solution Explorer. Một cửa sổ với các xác lập ứng dụng sẽ xuất hiện, như minh họa trong hình 3.11. Nhấp tab Application và nhìn vào mục Windows Application Properties ở cuối tab.



Hình 3.11. Các xác lập ứng dụng trong Visual Studio 2005

Bạn có hai lựa chọn cho hộp "Shutdown mode":

When startup form closes (Khi form khởi động đóng)

Tùy chọn này phù hợp với cách hoạt động chuẩn của một ứng dụng trong .NET 1.0 và 1.1. Ngay khi form khởi động đóng, toàn bộ chương trình sẽ thoát, cùng với bất kỳ cửa sổ khác. (Form khởi động là form được chỉ định trong hộp "Startup object").

When last form closes (Khi form cuối cùng đóng)

Tùy chọn này phù hợp với cách hoạt động của Visual Basic 6. Ứng dụng của bạn vẫn tiếp tục cuộn miễn sao vẫn còn một cửa sổ đang mở. Khi cửa sổ cuối cùng được đóng, ứng dụng sẽ tự thoát.

Nếu không có tùy chọn nào phù hợp, bạn có thể thực hiện điều này bằng tay. Trước tiên, chọn hộp kiểm "Startup with custom Sub Main". Bây giờ, bạn cần thêm một thường con được gọi là Main ( ) vào ứng dụng của bạn. Bạn có thể đặt thường con này vào một form hay lớp hiện có, miễn sao bạn nhớ thêm từ khóa truy cập Shared vào. Sau đây là một ví dụ:

```
Public Class MyForm

    Public Shared Sub Main
        ' (Startup code goes here.)
    End Sub

End Class
```

Các phương thức chia sẻ luôn có sẵn, cho dù không có trường hợp thể hiện trực tiếp của lớp đang chứa. Chẳng hạn, nếu bạn thêm một phương thức Main ( ) vào một form, .NET runtime có thể gọi phương thức Main ( ) ngay cả khi không có một đối tượng form nào. Một lựa chọn khác là thêm phương thức Main ( ) vào một module. Trong một module, mọi phương thức, hàm, thuộc tính và biến hoạt động như thể nó được chia sẻ, vì vậy bạn sẽ không cần thêm từ khóa Shared. Sau đây là một ví dụ:

```
Public Module MyModule

    Public Sub Main
        ' (Startup code goes here.)
    End Sub

End Module
```

Cho dù bạn lựa chọn như thế nào, phải bảo đảm lớp hay module chứa phương thức Main ( ) được chọn trong hộp "Startup object".

Ghi chú

Sử dụng một module là một lựa chọn thích hợp nếu bạn có nhiều phần khởi tạo cần thực hiện, bởi vì nó tách mã khởi động của bạn với mã form.

Khi bạn sử dụng một phương thức Main ( ) để khởi động ứng dụng, ứng dụng chỉ chạy nếu phương thức Main ( ) hoạt động. Ngay sau khi Main ( ) kết thúc, ứng dụng của bạn sẽ đóng. Sau đây là một ví dụ về một ứng dụng được kết thúc sớm:

```
Public Sub Main
    ' Show one form modelessly (without blocking the code).
    Form1.Show( )

    ' Show another form modelessly (at the same time as the first).
    Form2.Show( )

    ' After this line, the Main method ends, the application shuts
    ' itself down, and both windows close (after only being open a
    ' for a few milliseconds of screen time).
End Sub
```

Và sau đây là mã chính xác để hiển thị hai cửa sổ nối tiếp nhau:

```
Public Sub Main
    ' Show one form modally (code stops until the window is closed).
    Form1.ShowDialog( )

    ' After the first window is closed, show the second modally.
    Form2.ShowDialog( )

    ' Now the application ends.
End Sub
```

Trong một số trường hợp, bạn có thể muốn khởi động ứng dụng của mình với một phương thức Main ( ) để thực hiện một số phần khởi tạo cơ bản và hiển thị một vài form. Sau đó, bạn có thể muốn chờ đợi cho đến

khi tất cả các form đều được đóng trước lúc ứng dụng kết thúc. Mẫu này thì dễ thực thi, nếu bạn sử dụng lớp Application. Mục đích cơ bản là gọi Application.Run ( ) để giữ cho ứng dụng của bạn tồn tại một cách không xác định, và gọi Application.Exit ( ) vào một thời điểm sau đó để kết thúc nó. Sau đây là cách bạn có thể khởi động ứng dụng với hai cửa sổ hiển thị:

```
Public Sub Main
    ' Show two forms modelessly (and at the same time).
    Form1.Show( )
    Form2.Show( )

    ' Keep the application going until you say otherwise.
    Application.Run( )
End Sub
```

Để chỉ định ứng dụng sẽ kết thúc khi một trong hai cửa sổ đóng, bạn sử dụng mã sau trong event handler Form.Unload của cả hai form.

```
Private Sub Form1_FormClosed(ByVal sender As Object, _
  ByVal e As FormClosedEventArgs) Handles Me.FormClosed
    Application.Exit( )
End Sub
```

3.11.2. Thế còn...

... việc dọn sạch khi ứng dụng thoát thì sao? Khi ứng dụng của bạn kết thúc, bạn có thể muốn giải phóng các nguồn tài nguyên chưa được quản lý, xóa các file tạm thời hoặc lưu một số xác lập sau cùng. Lớp Application cung cấp một giải pháp với event ApplicationExit của nó. Tất cả những gì bạn cần làm là gán event này cho một event handler thích hợp trong phương thức Main ( ). Sau đây là một ví dụ sử dụng phương thức Shutdown ( ):

Ghi chú

ApplicationExit Event luôn kích khởi (và mã trong một event handler dành cho nó luôn chạy), ngay cả khi ứng dụng đã bị làm lệch bởi một ngoại lệ chưa được xử lý.

```
AddHandler Application.ApplicationExit, AddressOf Shutdown
```

Và đây là phương thức Shutdown ( ) chạy tự động ngay trước khi ứng dụng kết thúc:

```
Public Sub Shutdown(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Cleaning up.")

End Sub
```

Thủ thuật

Phần thực hành này sử dụng lớp Application từ network System.Windows.Forms. Mục này tương tự đối tượng My.Application, nhưng thật ra nó khác. Về kỹ thuật, đối tượng My.Application là một lớp được tạo động (được tạo bởi Visual Studio và được che giấu khỏi khung xem), vốn kế thừa từ WindowsFormsApplicationBase. Trên hết, đối tượng My.Application thường hoạt động giống như một phiên bản đã được đơn giản của lớp System.Windows.Forms.Application. Điều này cho phép .NET cung cấp một lớp cho những người lập trình muốn sự đơn giản và một lớp khác cho những người muốn có đầy đủ các tính năng.

3.12. Ngăn chặn ứng dụng khởi động hai lần

Bạn muốn bảo đảm người dùng không chạy quá một bản sao của ứng dụng trên cùng một máy tính? Trong VN .NET 1.0, bạn sẽ cần thực hiện qua tác vụ tìm kiếm mọi tiến trình đã được tải để bảo đảm chương trình của bạn chưa có trong bộ nhớ. Trong VB 2005, công việc này được thực hiện cho bạn.

Ghi chú

Bây giờ bạn không còn cần viết mã nữa để kiểm tra ứng dụng của bạn đã chạy hay chưa. VB 2005 sẽ thực hiện việc kiểm tra này giúp bạn.

3.12.1. Bạn làm điều đó như thế nào?

Trong Visual Studio, hãy nhấp double vào mục My Project trong Solution Explorer. Một cửa sổ với các xác lập ứng dụng sẽ xuất hiện. Nhấp tab Application và nhìn vào mục Windows Application Properties ở cuối tab. Bây giờ nhấp hộp kiểm "Make single instance application" và tạo dự án.

Nếu bạn cố khởi động ứng dụng trong khi nó đã chạy, nó sẽ bỏ qua bạn hoàn toàn và không có điều gì xảy ra.

3.12.2. Thế còn...

... việc hiển thị một thông báo lỗi tùy ý? Nếu bạn cần hiển thị một thông báo lỗi, hãy kiểm tra các trường hợp khác mà không dừng ứng dụng hoặc điều chỉnh mã, sau đó bạn cần kiểm tra bằng cách sử dụng lớp System.Diagnostics.Process. Sau đây là mã để bạn bắt đầu:

```
' Get the full name of the process for the current application.
Dim ModuleName, ProcessName As String
ModuleName = Process.GetCurrentProcess.MainModule.ModuleName
ProcessName = System.IO.Path.GetFileNameWithoutExtension(ModuleName)

' Check for other processes with this name.
Dim Proc( ) As System.Diagnostics.Process
Proc = Process.GetProcessesByName(ProcessName)
If Proc.Length > 1 Then
    ' (There is another instance running.)
Else
    ' (There are no other instances running.)
End If
```

3.13. Giao tiếp giữa các Form

Trong các phiên bản trước của .NET, bạn chịu trách nhiệm theo dõi mọi form mở. Nếu không, bạn có thể gặp phải tình trạng cửa sổ vẫn được để mở nhưng bị cắt ra khỏi phần còn lại của ứng dụng. VB 2005 phục hồi phương pháp được ưa thích bởi các nhà phát triển VB 6, trong đó luôn có một trường hợp mặc định của form đã sẵn sàng, chờ đợi và có thể truy cập từ bất kỳ nơi khác trong ứng dụng của bạn.

Ghi chú

VB 2005 giúp cho các form dễ dàng tương tác nhờ vào các trường hợp thể hiện mặc định mới. Tính năng này giúp tiết kiệm khá nhiều thời gian.

3.13.1. Bạn làm điều đó như thế nào?

Để truy cập thể hiện mặc định của một form, bạn chỉ cần sử dụng tên lớp của nó. Nói cách khác, nếu bạn đã tạo một form được đặt tên là Form1, bạn có thể hiển thị phần thể hiện (instance) mặc định của nó như sau:

```
Form1.Show ( )
```

Điều này sẽ tự động tạo một thể hiện của Form1 và sau đó hiển thị nó. Phần thể hiện này của Form1 được gọi là thể hiện mặc định (default instance).

Để giao tiếp giữa các form, bạn chỉ cần thêm vào các phương thức public chuyên biệt. Chẳng hạn, nếu Form1 cần có khả năng làm mới Form2, bạn có thể thêm một phương thức RefreshData ( ) vào Form2, như sau:

```
Public Class Form2

    Private Sub RefreshData( )

        MessageBox.Show("I've been refreshed!")

    End Sub

End Class
```

Sau đó, bạn có thể gọi nó như sau:

```
Form2.RefreshData ( )
```

Mã này gọi phương thức RefreshData ( ) của instance mặc định của Form2. Việc RefreshData ( ) là một phương thức bạn đã thêm vào (không phải là phương thức thừa kế như phương thức Show ( )) không tạo ra bất kỳ sự khác biệt về cách bạn sử dụng nó.

Bạn cũng có thể đến tại các form bằng cách sử dụng tập hợp My. Chẳng hạn, mã ở trên tương đương với câu lệnh hơi dài hơn này:

```
My.Form2.Form2.RefreshData ( )
```

Bạn luôn có thể truy cập instance mặc định của một form, cho dù nó không đang hiển thị. Thật vậy, .NET tạo ra instance mặc định của form ngay khi bạn truy cập một trong các thuộc tính hay phương thức của nó. Nếu bạn chỉ muốn tìm ra những form nào đang mở, tốt hơn bạn nên sử dụng tập hợp My.Application.OpenForms. Sau đây là một ví dụ lặp qua tập hợp và hiển thị chú thích của mỗi form:

```
For Each frm As Form In My.Application.OpenForms

    MessageBox.Show(frm.Text)

Next
```

Thủ thuật này không áp dụng được trong các phiên bản trước của .NET nếu như không viết mã riêng để theo dõi các form bằng phương pháp thủ công.

3.13.2. Thế còn...

... các vấn đề tiềm ẩn thì sao? Trong trường hợp này, bạn không cần bận tâm về bộ nhớ hao phí hay bất kỳ sự giảm chậm hiệu suất, vì .NET đủ thông minh để tạo ra các form khi bạn cần chúng. Vấn đề thật sự là bạn có thể gặp phải các kết quả từ việc các instance mặc định làm xáo trộn các khái niệm của các lớp và các đối tượng, khiến cho mọi thứ trở nên quá dễ dàng đến nỗi không thể quy chiếu các instance khác nhau của cùng một form trong các phần khác nhau của ứng dụng.

Ghi chú

Bạn cũng có thể có một tham chiếu đến form khởi động của ứng dụng nhờ dùng thuộc tính My.Application.StartupForm.

Chẳng hạn, hãy tưởng tượng bạn sử dụng mã sau đây để hiển thị một form:

```
Dim FormObject As New Form1

FormObject.Show( )
```

Trong ví dụ này, form bạn đã hiển thị là một instance của Form1, nhưng nó không phải là instance mặc định. Điều đó có nghĩa là nếu một phần khác của mã sử dụng mã như sau:

```
Form1.Refresh ( )
```

nó sẽ không có hiệu ứng như bạn mong đợi. Instance hiển thị của Form1 sẽ không được làm mới. Thay vào đó, instance mặc định (có lẽ cũng không hiển thị) sẽ được làm mới. Hãy lưu ý vấn đề này vì nó có thể dẫn đến những sự cố khác thường!

3.14. Cải tiến tốc độ vẽ lại cho GDI+

Mọi control và form trong .NET kế thừa từ lớp control cơ sở. Trong .NET 2.0, lớp Control xuất một thuộc tính mới được gọi là DoubleBuffered. Nếu bạn xác lập thuộc tính này sang TRue, form hay control sẽ tự động sử dụng double-buffering (đệm kép), vốn làm giảm sự nhấp nháy khi bạn thêm mã vẽ tùy ý.

Ghi chú

Bạn cần tăng cường các ảnh động GDI+? Trong .NET 2.0, lớp Form có thể thực hiện double-buffering cho bạn.

3.14.1. Bạn làm điều đó như thế nào?

Trong một số ứng dụng, bạn cần vẽ lại một cửa sổ hay control thường xuyên. Chẳng hạn, bạn có thể làm mới một cửa sổ 10 mili gây một lần để tạo ảo giác về một ảnh động liên tục. Mỗi lần cửa sổ được làm mới, bạn cần xóa nội dung hiện tại và vẽ khung mới lại từ đầu.

Trong một ứng dụng đơn giản, logic vẽ của bạn có thể vẽ một hình dạng đơn. Trong một ảnh động phức tạp hơn, bạn có thể dễ dàng mô phỏng hàng chục thành phần đồ họa khác nhau cùng một lần. Sự mô phỏng các thành phần này chiếm một lượng thời gian nhỏ nhưng đáng kể. Vấn đề là nếu bạn vẽ mỗi thành phần đồ họa trực tiếp trên form, ảnh động sẽ nhấp nháy khi ảnh được xóa và tạo lại nhiều lần. Để tránh vấn đề phiền toái này, các nhà phát triển thường sử dụng một kỹ thuật được gọi là double-buffering (đệm kép). Với double-buffering, mỗi khung mới được lắp ráp đầy đủ trong bộ nhớ và chỉ được vẽ trên form khi nó hoàn chỉnh.

.NET 2.0 hoàn toàn giúp bạn tránh được vấn đề rắc rối của double-buffering. Tất cả những gì bạn cần làm là xác lập thuộc tính DoubleBuffered của form hoặc control sang TRue. Chẳng hạn, hãy tưởng tượng bạn tạo một form và xử lý event Paint để cung cấp logic vẽ tùy ý của bạn. Nếu form được xác lập để sử dụng double-buffering, nó sẽ không được làm mới cho đến khi Paint event handler đã hoàn tất, vào lúc này nó sẽ sao chép ảnh đã hoàn chỉnh trực tiếp lên trên form. Nếu DoubleBuffered được xác lập False, mỗi lần bạn vẽ một thành phần riêng lẻ lên trên form trong event handler Pain, form sẽ được làm mới. Do vậy, form sẽ được làm mới hàng chục lần cho bất kỳ thành phần nào ngoại trừ những hoạt động đơn giản nhất.

Ví dụ 3.5 minh họa một form sử dụng logic vẽ tùy ý. Khi người dùng nhấp nút cmdStart, một bộ định giờ (timer) được mở. Bộ định giờ này kích khởi cứ vài mili giây một lần và không hợp thức form bằng cách gọi phương thức Invalidate ( ) của nó. Để phản hồi, Windows yêu cầu ứng dụng vẽ lại cửa sổ, kích khởi phương thức OnPaint ( ) với mã vẽ tùy ý.

Ví dụ 3.5. Một form được tạo động

```
Public Class AnimationForm

    ' Indicates whether the animation is currently being shown.
    Private IsAnimating As Boolean = False

    ' Track how long the animation has been going on.
    Private StartTime As DateTime
```

```
Private Sub Form_Paint(ByVal sender As Object, _
  ByVal e As System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint

  ' Check if the animation is in progress.
  If IsAnimating Then

    ' Get reading to draw the current frame.
    Dim g As Graphics = e.Graphics
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality

    ' Paint the background.
    Dim BackBrush As New LinearGradientBrush( _
      New Point(0, 0), New Point(100, 100), _
      Color.Blue, Color.LightBlue)
    g.FillRectangle(BackBrush, New Rectangle(New Point(0, 0), _
      Me.ClientSize))
    g.FillRectangle(Brushes.LightPink, New Rectangle(New Point(10, 10), _
      New Point(Me.Width - 30, Me.Height - 50)))

    ' Calculate elapsed time.
    Dim Elapsed As Double = DateTime.Now.Subtract(StartTime).TotalSeconds

    Dim Pos As Double = (-100 + 24 * Elapsed ^ 2) / 10

    ' Draw some moving objects.
    Dim Pen As New Pen(Color.Blue, 10)
    Dim Brush As Brush = Brushes.Chartreuse
    g.DrawEllipse(Pen, CInt(Elapsed * 100), CInt(Pos), 10, 10)
    g.FillEllipse(Brush, CInt(Elapsed * 100), CInt(Pos), 10, 10)

    g.DrawEllipse(Pen, CInt(Elapsed * 50), CInt(Pos), 10, 10)
    g.FillEllipse(Brush, CInt(Elapsed * 50), CInt(Pos), 10, 10)

    g.DrawEllipse(Pen, CInt(Elapsed * 76), CInt(Pos) * 2, 10, 10)
    g.FillEllipse(Brush, CInt(Elapsed * 55), CInt(Pos) * 3, 10, 10)
```

```
            g.DrawEllipse(Pen, CInt(Elapsed * 66), CInt(Pos) * 4, 10, 10)
            g.FillEllipse(Brush, CInt(Elapsed * 72), CInt(Pos) * 3, 10, 10)

            If Elapsed > 10 Then
                ' Stop the animation.
                tmrInvalidate.Stop( )
                IsAnimating = False
            End If

        Else
            ' There is no animation underway. Paint the background.
            MyBase.OnPaintBackground(e)
        End If

    End Sub

    Private Sub tmrInvalidate_Tick(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles tmrInvalidate.Tick
        ' Invalidate the form, which will trigger a refresh.
        Me.Invalidate( )
    End Sub

    Private Sub cmdStart_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles cmdStart.Click
        ' Start the timer, which will trigger the repainting process
        ' at regular intervals.
        Me.DoubleBuffered = True
        IsAnimating = True
        StartTime = DateTime.Now
        tmrInvalidate.Start( )
    End Sub

    ' Ensure that the form background is not repainted automatically
    ' when the form is invalidated. This isn't necessary, because the
    ' Paint event will handle the painting for the form.
    ' If you don't override this method, every time the form is painted
```

```
    ' the window will be cleared and the background color will be painted
    ' on the surface, which causes extra flicker.
    Protected Overrides Sub OnPaintBackground( _
      ByVal pevent As System.Windows.Forms.PaintEventArgs)
        ' Do nothing.
    End Sub


End Class
```

Thử chạy ví dụ trên có và không có áp dụng kỹ thuật double-buffering. Bạn sẽ thấy một sự khác biệt khá lớn về mức độ nhấp nháy.

3.14.2. Thế còn...

... còn control owner-drawn thì sao? Double-buffering hoạt động giống hệt như cách hoạt động của các control owner-drawn (do người sở hữu vẽ) với các form, bởi vì cả hai lớp Form và Control đều cung cấp thuộc tính DoubleBuffered và event Paint. Dĩ nhiên, không có điểm nào trong việc double-buffering cả form lẫn các control của nó, vì điều đó sẽ chỉ làm cho ứng dụng của bạn chiếm thêm lượng bộ nhớ không cần thiết mà thôi.

3.15. Xử lý các tác vụ không đồng bộ một cách an toàn

Một trong những tính năng quan trọng nhất của .NET là sự hỗ trợ bao quát của nó đối với lập trình đa luồng (multithreaded). Tuy nhiên, lập trình đa luồng không dễ như vậy.

Ghi chú

Bạn cần điều khiển một tác vụ chiếm nhiều thời gian trong nền mà không xử lý các vấn đề tạo luồng (threading)? Lớp BackgroundWorker mới giúp dễ dàng thực hiện điều này.

Một trong những thách thức chính với các ứng dụng Windows là không an toàn để chỉnh sửa một form hay control từ một luồng trong nền, nghĩa là sau khi tác vụ trong nền đã hoàn tất, không có cách đơn giản nào để cập nhật giao diện ứng dụng của bạn. Bạn có thể sử dụng phương thức Control.Invole ( ) để viện dẫn một phương thức theo đúng luồng (thread), nhưng các vấn đề khác sau đó lại xuất hiện, chẳng hạn như chuyển thông tin bạn cần để thực hiện việc cập nhật. Thật may, bạn có thể tránh được tất cả những vấn đề đau đầu này nhờ vào thành phần BackgroundWorker mới.

3.15.1. Bạn làm điều đó như thế nào?

Thành phần BackgroundWorker cung cấp cho bạn một cách để chạy một tác vụ chiếm thời gian trên một luồng chuyên dụng riêng biệt. Điều này bảo đảm giao diện ứng dụng của bạn vẫn phản hồi và nó cho phép mã của bạn thực hiện những tác vụ khác trong tiền cảnh. Trên hết, các tính phức tạp của lập trình đa luồng được ẩn giấu. Ngay sau khi tiến trình trong nền hoàn tất, bạn chỉ cần xử lý một event, vốn kích khởi trên luồng chính. Ngoài ra, BackgroundWorker hỗ trợ việc báo cáo tiến độ và hủy.

Bạn có thể tạo một đối tượng BackgroundWorker bằng lập trình hoặc rê nó sang một form từ tab Components của hộp công cụ. Để bắt đầu hoạt động trong nền, bạn gọi phương thức RunWorkerAsync ( ). Nếu bạn cần chuyển một trị nhập liệu đến tiến trình này, bạn có thể cung cấp nó ở dạng một đối số cho phương thức này (mọi kiểu dữ liệu của đối tượng đều được chấp nhận):

```
Worker.RunWorkerAsync (inputValue)
```

Kế tiếp, bạn cần xử lý event DoWork để thực hiện tác vụ trong nền. Event DoWork kích khởi trên luồng trong nền (background thread), nghĩa là vào lúc này bạn không thể tương tác với bất kỳ phần khác của ứng dụng (trừ khi bạn sẵn sàng sử dụng các khóa hay các kỹ thuật khác để bảo vệ sự truy cập. Thông thường, event handler DoWork truy xuất trị nhập liệu từ thuộc tính DoWorkEventArgs.Argument và sau đó thực hiện hoạt động chiếm thời gian. Ngay khi hoạt động này hoàn tất, bạn chỉ cần xác lập thuộc tính DoWorkEventArgs.Result với kết quả. Bạn có thể sử dụng bất kỳ kiểu dữ liệu nào hoặc thậm chí một đối tượng tùy ý. Sau đây là dạng cơ bản mà bạn sẽ sử dụng:

```
Private Sub backgroundWorker1_DoWork(ByVal sender As Object, _
  ByVal e As DoWorkEventArgs) Handles backgroundWorker1.DoWork

    ' Get the information that was supplied.
    Dim Input As Integer = CType(e.Argument, Integer)

    ' (Perform some time consuming task.)

    ' Return the result.
    e.Result = Answer

  End Sub
```

Sau cùng, BackgroundWorker kích khởi một event RunWorker Com-

pleted để thông báo cho ứng dụng của bạn biết rằng tiến trình đã hoàn tất. Vào lúc này, bạn có thể truy xuất kết quả từ RunWorkerCompleted EventArgs và cập nhật form phù hợp:

```
Private Sub backgroundWorker1_RunWorkerCompleted( _
ByVal sender As Object, ByVal e As RunWorkerCompletedEventArgs) _
Handles backgroundWorker1.RunWorkerCompleted

  result.Text = "Result is: " & e.Result.ToString( )

End Sub
```

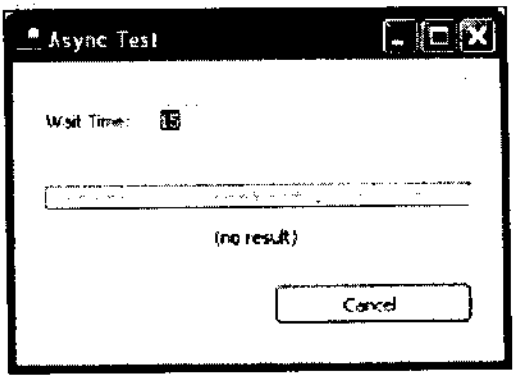
Ví dụ 3.6 minh họa một form đặt mọi thành phần này lại với nhau. Nó thực hiện một vòng lặp giới hạn thời gian trong khoảng số giây mà bạn chỉ định. Ví dụ này cũng minh họa hai kỹ thuật cao cấp khác: hủy và tiến độ. Để hủy hoạt động, bạn chỉ cần gọi phương thức BackgroundWorker.CancelAsync ( ). Mã xử lý event DoWork sau đó có thể kiểm tra để xem form chính có đang cố hủy hoạt động và thoát một cách êm ái hay không. Để duy trì thông tin tiến độ, mã xử lý event DoWork cần gọi phương thức BackgroundWorker.ReportProgress ( ) và cung cấp một phần trăm ước tính hoàn chỉnh (0% có nghĩa là "chỉ mới bắt đầu" và 100% có nghĩa là "đã hoàn tất"). Mã form có thể phản hồi event ProgressChanged để đọc phần trăm tiến độ mới và cập nhật một control khác, chẳng hạn như ProgressBar. Hình 3.12 minh họa ứng dụng này đang hoạt động.

Ví dụ 3.6. Một form không đồng bộ với BackgroundWorker

```
Public Class AsyncForm

  Private Sub startAsyncButton_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles startAsyncButton.Click

    ' Disable the Start button until
    ' the asynchronous operation is done.
    startAsyncButton.Enabled = False

    ' Enable the Cancel button while
    ' the asynchronous operation runs.
    cancelAsyncButton.Enabled = True
    ' Start the asynchronous operation.
```

```
    backgroundWorker1.RunWorkerAsync(Int32.Parse(txtWaitTime.Text))
End Sub

' This event handler is where the actual work is done.
Private Sub backgroundWorker1_DoWork(ByVal sender As Object, _
  ByVal e As DoWorkEventArgs) Handles backgroundWorker1.DoWork

    ' Get the information that was supplied.
    Dim Worker As BackgroundWorker = CType(sender, BackgroundWorker)

    Dim StartTime As DateTime = DateTime.Now
    Dim SecondsToWait As Integer = CType(e.Argument, Integer)
    Dim Answer As Single = 100
    Do
        ' Check for any cancellation requests.
        If Worker.CancellationPending Then
            e.Cancel = True
            Return
        End If

        ' Continue calculating the answer.
        Answer *= 1.01

        ' Report the current progress (percentage complete).
        Worker.ReportProgress(( _
          DateTime.Now.Subtract(StartTime).TotalSeconds / SecondsToWait) * 100)

        Thread.Sleep(50)
    Loop Until DateTime.Now > (StartTime.AddSeconds(SecondsToWait))

    e.Result = Answer
End Sub

' This event handler fires when the background work
' is complete.
Private Sub backgroundWorker1_RunWorkerCompleted( _
```

```
ByVal sender As Object, ByVal e As RunWorkerCompletedEventArgs) _
Handles backgroundWorker1.RunWorkerCompleted

    ' Check what the result was, and update the form.
    If Not (e.Error Is Nothing) Then
        ' An exception was thrown.
        MessageBox.Show(e.Error.Message)
    ElseIf e.Cancelled Then
        ' Check if the user cancelled the operation.
        result.Text = "Cancelled"
    Else
        ' The operation succeeded.
        result.Text = "Result is: " & e.Result.ToString( )
    End If

    startAsyncButton.Enabled = True
    cancelAsyncButton.Enabled = False
End Sub

' This event handler updates the progress bar.
Private Sub backgroundWorker1_ProgressChanged( _
  ByVal sender As Object, ByVal e As ProgressChangedEventArgs) _
  Handles backgroundWorker1.ProgressChanged
    Me.progressBar1.Value = e.ProgressPercentage
End Sub

Private Sub cancelAsyncButton_Click( _
  ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles cancelAsyncButton.Click

    ' Cancel the asynchronous operation.
    Me.backgroundWorker1.CancelAsync( )

    cancelAsyncButton.Enabled = False
  End Sub
End Class
```



Hình 3.12. Giám sát một tác vụ trong nền

3.15.2. Thế còn...

... các tình huống khác mà trong đó bạn có thể sử dụng BackgroundWorker thì sao? Ví dụ này đã sử dụng BackgroundWorker với một phép tính mất nhiều thời gian đang được thực hiện trong nền. Các tình huống khác chứng tỏ không thể thiếu BackgroundWorker bao gồm:

- Liên hệ một dịch vụ web
- Tải xuống một file qua Internet
- Truy xuất dữ liệu từ một cơ sở dữ liệu
- Đọc hay viết các lượng dữ liệu lớn

3.16. Sử dụng một lưới liên kết dữ liệu tốt hơn

DataGrid đi kèm với .NET 1.0 và 1.1 có một vài hạn chế. Nó khó tùy biến, hầu như không thể mở rộng được, và không hỗ trợ các tính năng quan trọng như chỉnh sửa hay lấp đầy DataGrid bằng lập trình, truy cập các ô riêng lẻ, hoặc áp dụng kiểu định dạng trên từng ô. Trong nhiều trường hợp, các nhà phát triển VB đã tránh DataGrid và sử dụng các lưới của nhóm thứ ba hoặc thậm chí các control cũ hơn dựa vào COM như MSFlexGrid.

Ghi chú

DataGrid của .NET đã là một sự thất vọng rất lớn trong một framework hiện đại. Bây giờ đội ngũ Windows Form đã lấp đầy các khoảng hở này bằng một lưới hạng nhất.

Trong thiết kế .NET 2.0, đội ngũ Windows Form đã quyết định không thể khắc phục các nhược điểm nếu như không phá vỡ tính tương thích ngược. Vì vậy, họ đã chọn đưa ra một control DataGridView hoàn toàn mới với sự hỗ trợ cho mọi tính năng còn thiếu và nhiều tính năng khác.

3.16.1. Bạn làm điều đó như thế nào?

Bạn có thể kết buộc DataGridView vào một đối tượng DataTable theo cách mà bạn đã kết buộc một DataGrid. Sau đây là mã ở dạng cơ bản nhất mà bạn có thể sử dụng để kết buộc một table có tên Customers:

```
DataGridView1.DataSource = ds

DataGridView.DataMember = "Customers"
```

Dĩ nhiên, để mã này hoạt động, bạn cần tạo đối tượng DataSet ds và điền thông tin vào nó. Đối với một ví dụ hoàn chỉnh vốn thêm mã ADO.NET cần thiết cho bước này, hãy xem nội dung có thể tải xuống cho chương này.

Khi bạn sử dụng mã này, DataGridView tạo một cột cho mỗi trường trong biên dịch dữ liệu và đặt tựa cho nó nhờ dùng tên trường. Lưới cũng có một mức chức năng đáng kể. Một số đặc điểm mà bạn sẽ chú ý bao gồm:

- Các header cột được đóng băng tại chỗ. Điều đó có nghĩa là chúng sẽ không biến mất khi bạn cuộn xuống danh sách.
- Bạn có thể hiệu chỉnh các trị. Chỉ cần nhấp double vào một ô hoặc nhấn F2 để đưa nó vào chế độ hiện hành. (Bạn có thể tắt tính năng này bằng cách xác lập thuộc tính DataColumn.ReadOnly sang true trong DataTable nền tảng).
- Bạn có thể phân loại các cột. Chỉ cần nhấp vào header cột một hoặc hai lần.
- Bạn có thể tự động định kích cỡ các cột. Chỉ cần nhấp double vào vạch phân chia cột giữa các header để mở rộng một cột (cột ở bên trái) cho vừa với nội dung hiện hành.
- Bạn có thể chọn một dãy ô. Bạn có thể bật sáng một hay nhiều ô hoặc nhiều hàng bằng cách nhấp và rê. Để chọn toàn bộ table, bạn nhấp vào ô vuông ở góc trên cùng bên trái.
- Bạn có thể thêm các hàng bằng cách cuộn đến cuối lưới và nhập vào các trị mới. Để tắt tính năng này, hãy xác lập thuộc tính AllowUserToAddRows sang False.

- Bạn có thể xóa các hàng bằng cách chọn toàn bộ hàng (nhấp nút row ở bên trái) và chọn phím Delete. Để tắt tính năng này, bạn xác lập thuộc tính AllowUserToDeleteRows sang False.

Trước khi đi xa hơn với DataGridView, có hai phương thức mà bạn sẽ muốn xem xét sử dụng ngay vào lúc này:

AutoResizeColumns ( ) và AutoResizeRows ( ). AutoResizeColumns () mở rộng tất cả các cột cho vừa với text header và dữ liệu ô. AutoResizeRows ( ) mở rộng hàng với nhiều dòng cho vừa với text header và dữ liệu ô (DataGridView hỗ trợ sự bao bọc tự động). Cả hai phương thức này đều chấp nhận một trị từ một phép liệt kê mà nó cho phép bạn chỉ định các tùy chọn bổ sung (chẳng hạn như mở rộng cột cho vừa với chỉ tất cả các cột hoặc chỉ với text header):

```
' Create wider columns to fit data.
DataGridView1.AutoResizeColumns( _
  DataGridViewAutoSizeColumnsMode.AllCells)

' Create multi-line columns to fit data.
DataGridView1.AutoResizeRows( _
  DataGridViewAutoSizeRowsMode.HeaderAndColumnsAllCells)
```

Bạn cũng có thể sử dụng các phương thức AutoResizeColumn ( ) và AutoResizeRow ( ) để thay đổi chỉ một cột hay một hàng (được chỉ định là một số index).

Ngay sau khi bạn đã tạo một DataGridView và tập hợp nó với dữ liệu, bạn có thể tương tác với nó qua hai tập hợp hữu ích: Columns và Rows. Tập hợp Columns hiển thị một tập hợp các đối tượng DataGridViewCell, mỗi đối tượng dành cho một cột trong lưới. Bạn có thể xác lập thứ tự hiển thị của các cột (bằng cách xác lập một số index trong thuộc tính DisplayIndex), che giấu một cột (xác lập Visible sang false), hoặc làm đóng băng một cột sao cho nó luôn hiển thị ngay cả khi người dùng cuộn sang cạnh bên (xác lập Frozen sang true). Bạn cũng có thể chỉnh sửa text header của cột (HeaderText), kích cỡ (Width), và làm cho nó không thể được hiệu chỉnh (ReadOnly). Để tra tìm một cột, hãy sử dụng số index hoặc tên trường tương ứng.

Chẳng hạn, sau đây là mã mà bạn cần để thay đổi một số thuộc tính cột trong cột OrderID của một DataGridView đã được kết buộc.

```
' Keep this column visible on the left at all times.
DataGridView1.Columns("CustomerID").Frozen = True
DataGridView1.Columns("CustomerID").DisplayIndex = 0
```

```
' Configure the column appearance.
DataGridView1.Columns("CustomerID").HeaderText = "ID"
DataGridView1.Columns("CustomerID").Resizable = DataGridViewTriState.True
DataGridView1.Columns("CustomerID").MinimumWidth = 50
DataGridView1.Columns("CustomerID").Width = 50

' Don't allow the values in this column to be edited.
DataGridView1.Columns("CustomerID").ReadOnly = True
```

Tập hợp Rows cho phép bạn truy cập các đối tượng dataGridViewRow riêng lẻ theo số index (chỉ mục). Ngay khi bạn đã có một DataGridViewRow, bạn có thể xem xét tập hợp Cells của nó để tra tìm các trị riêng lẻ trong hàng.

Tuy nhiên, có nhiều khả năng bạn sẽ muốn truy cập chỉ những hàng tương ứng với sự lựa chọn hiện tại của người dùng. DataGridView thật sự cung cấp ba thuộc tính liên quan có thể giúp bạn được điều này.

SelectedRows

Cung cấp một tập hợp với một DataGridViewRow cho mỗi hàng được chọn hoàn toàn. Điều này hữu ích nếu SelectionMode chỉ cho phép chọn toàn bộ một hàng.

SelectedColumns

Cung cấp một tập hợp với một DataGridViewColumncho mỗi cột được chọn hoàn toàn. Điều này hữu ích nếu SelectionMode chỉ cho phép chọn toàn bộ một cột.

SelectedCells

Cung cấp một tập hợp với một DataGridViewCell cho mỗi ô được chọn, bất kể chế độ chọn nào đang được sử dụng. Bạn có thể sử dụng thuộc tính này nếu chế độ chọn của bạn cho phép chọn từng ô hoặc nếu bạn chỉ muốn xử lý mỗi ô một cách riêng biệt.

Chẳng hạn, nếu bạn đang sử dụng DataGridViewSelectionMode. FullRowSelect, bạn có thể dùng mã sau đây để truy xuất vùng chọn hiện hành và hiển thị một trường cụ thể từ mỗi hàng được chọn khi người dùng nhấp một nút:

Ghi chú

Bạn có thể điều khiển loại vùng chọn được chấp nhận bằng cách xác lập thuộc tính DataGridView.SelectionMode. Các trị khác cho phép chọn

các ô, các hàng hay các cột riêng lẻ. DataGridView.MultiSelect xác định nhiều mục có được chọn đồng thời hay không.

```
Private Sub cmdSelection_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles cmdSelection.Click

    For Each SelectedRow As DataGridViewRow In DataGridView1.SelectedRows
        MessageBox.Show(SelectedRow.Cells("CustomerID").Value)
    Next

End Sub
```

3.16.2. Thế còn...

... việc thực hiện những tác vụ khác với DataGridView thì sao? Các tính năng được mô tả cho đến lúc này cung cấp tóm tắt những điểm cơ bản về DataGridView, nhưng chúng chỉ là sự sơ lược về các tính năng tùy biến. Để tìm hiểu thêm, hãy xem hai mục 3.17 và 3.18 trong chương này).

3.17. Định dạng DataGridView

Việc định dạng .NET 1.x DataGrid thì khó và cũng có thể không thực hiện được. Tuy nhiên, nhờ vào mô hình nhiều lớp của nó, việc định dạng DataGridView trở nên dễ dàng hơn nhiều. Mô hình này dựa trên một lớp duy nhất, DataGridViewCellStyle, gói gọn mọi thuộc tính định dạng chính. Bạn có thể gán các đối tượng DataGridViewCellStyle khác cho các hàng, các cột hay thậm chí các ô phân biệt.

Ghi chú

Bằng cách sử dụng vài thuộc tính style đơn giản, bạn có thể cấu hình diện mạo của toàn bộ lưới, các cột hay các hàng riêng lẻ với dữ liệu quan trọng.

3.17.1. Bạn làm điều đó như thế nào?

DataGridView đã có vẻ tốt hơn DataGrid trong trạng thái mặc định của nó. Chẳng hạn, bạn sẽ nhận thấy các header cột đã có một diện mạo phẳng, hiện đại và được bật sáng khi người dùng di chuyển chuột qua chúng. Tuy nhiên, có nhiều thứ hơn mà bạn có thể làm với sự trợ giúp của lớp DataGridViewCellStyle.

DataGridViewCellStyle thu thập mọi thuộc tính định dạng của DataGridView. Nó định nghĩa các xác lập liên quan đến diện mạo (ví dụ, màu sắc, font) và định dạng dữ liệu (ví dụ, dạng tiền tệ, ngày tháng). Trên hết, DataGridViewCellStyle cung cấp các thuộc tính chính sau đây:

Alignment

Xác lập cách text được canh đều bên trong ô.

BackColor và ForeColor

Xác lập màu của nền ô và màu của text ô.

Font

Xác lập font được sử dụng cho text ô.

Format

Một chuỗi định dạng cấu hình cách các trị dữ liệu số hay ngày tháng sẽ được định dạng là các chuỗi. Bạn có thể sử dụng các thuộc tính định dạng .NET chuẩn và chuỗi định dạng tùy ý của riêng bạn. Chẳng hạn, C chỉ định một trị tiền tệ.

NullText

Một chuỗi text sẽ được thay thế cho bất kỳ trị null (thiếu).

SelectionBackColor và SelectionForeColor

Xác lập các màu nền ô và các màu text cho các ô được chọn.

WrapMode

Xác định text sẽ tràn qua nhiều dòng (nếu hàng đủ cao để chứa nó) hay nó sẽ bị cắt bớt. Theo mặc định, các ô sẽ bao bọc.

Phần hấp dẫn là bạn có thể tạo và xác lập các đối tượng DataGridViewCellStyle ở các cấp độ khác nhau. Khi DataGridView hiển thị một ô, nó tìm thông tin style ở nhiều nơi. Sau đây là thứ tự quan trọng từ cao nhất đến thấp nhất:

1. DataGridViewCell.Style
2. DataGridViewRow.DefaultCellStyle
3. DataGridView.AlternatingRowsDefaultCellStyle
4. DataGridView.RowsDefaultCellStyle
5. DataGridViewColumn.DefaultCellStyle
6. DataGridView.DefaultCellStyle

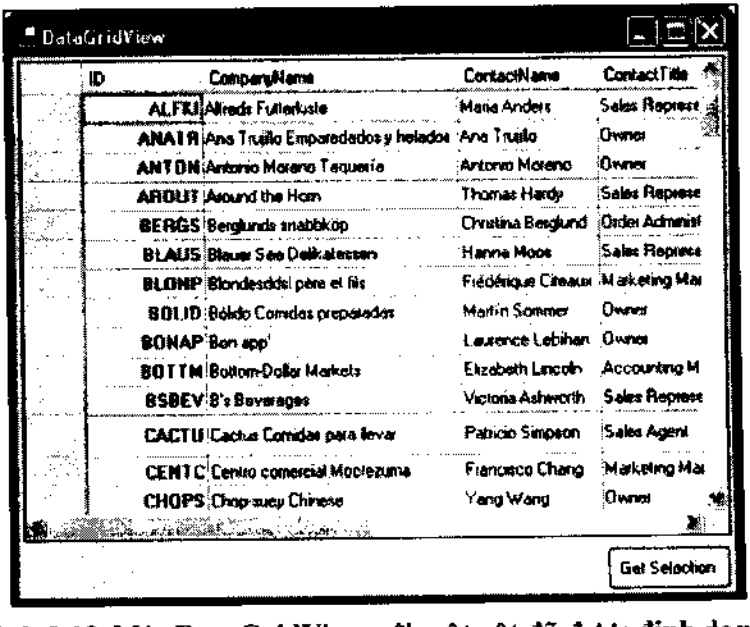
Nói cách khác, nếu DataGridView tìm thấy một đối tượng DataGridViewCellStyle được gán cho ô hiện hành (tùy chọn 1), nó luôn sử dụng nó. Nếu không, nó kiểm tra DataGridViewCellStyle dành cho hàng và cứ thế tiếp tục.

Đoạn mã sau đây thực hiện việc định dạng cho cột. Nó bảo đảm mọi trị trong cột CustomerID được cung cấp một font, kiểu gióng thẳng và tập hợp màu khác. Hình 3.13 minh họa kết quả.

Ghi chú

Nếu bạn sử dụng các tính năng kết buộc dữ liệu vào lúc thiết kế của Visual Studio, bạn có thể tránh viết mã này. Chỉ cần nhấp liên kết Edit Columns trong Properties Windows và sử dụng trình thiết kế để chọn kiểu định dạng.

```
Dim Style As DataGridViewCellStyle = _
  DataGridView1.Columns("CustomerID").DefaultCellStyle
Style.Font = New Font(DataGridView1.Font, FontStyle.Bold)
Style.Alignment = DataGridViewContentAlignment.MiddleRight
Style.BackColor = Color.LightYellow
Style.ForeColor = Color.DarkRed
```



Hình 3.13. Một DataGridView với một cột đã được định dạng

3.17.2. Thế còn...

... cách dễ nhất để áp dụng kiểu định dạng ô tùy ý thì sao? Đôi lúc, bạn muốn tập trung sự chú ý vào các ô có chứa các trị nhất định. Bạn có thể xử lý điều này bằng cách rảo lặp qua toàn bộ lưới, tìm những ô hấp dẫn đối với bạn. Tuy nhiên, bạn có thể tiết kiệm thời gian bằng cách phản hồi event DataGridView.CellFormatting. Event này xảy ra khi lưới đang được lấp đầy. Nó cho bạn cơ hội để kiểm tra ô và thay đổi style của nó trước khi nó xuất hiện.

Sau đây là một ví dụ định dạng một ô để bật sáng các giá cao:

```
Private Sub DataGridView1_CellFormatting(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.DataGridViewCellFormattingEventArgs) _
Handles DataGridView1.CellFormatting

    ' Check if this is the right column.
    If DataGridView1.Columns(e.ColumnIndex).Name = "Price" Then
        ' Check if this is the right value.
        If e.Value > 100 Then
            e.CellStyle.ForeColor = Color.Red
            e.CellStyle.BackColor = Color.Yellow
        End If
    End If

End Sub
```

Lưu ý rằng bạn nên sử dụng lại các đối tượng style nếu hoàn toàn có thể được. Nếu bạn gán một đối tượng style mới cho mỗi ô, bạn sẽ mất nhiều lượng bộ nhớ. Phương pháp tốt hơn là tạo một đối tượng style và gán nó cho nhiều ô có cùng kiểu định dạng.

3.18. Thêm hình ảnh và Control vào DataGridView

Để tạo một cột tùy ý với DataGrid, bạn cần tự mình thực thi chức năng này bằng cách tạo một lớp phái sinh DataGridColumnStyle tùy ý mà sẽ cần hàng chục dòng mã. DataGridView cung cấp một mô hình đơn giản hơn nhiều. Thật vậy, bạn có thể thêm các cột mới ngay sát bên các cột kết buộc dữ liệu của bạn!

Ghi chú

Có nhiều thứ hơn mà bạn có thể làm với DataGridView, bao gồm thêm các nút và các ảnh tĩnh.

3.18.1. Bạn làm điều đó như thế nào?

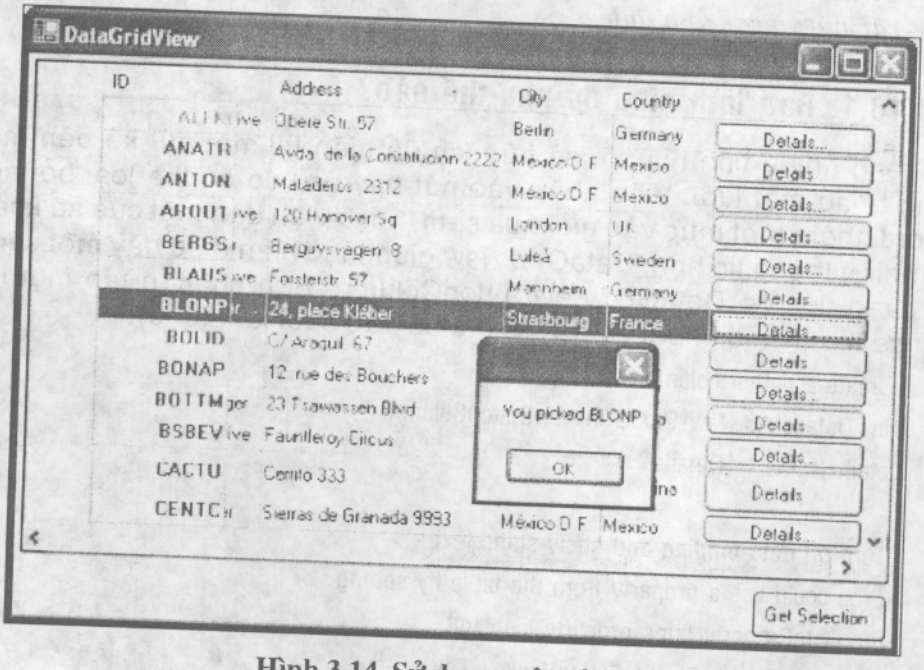
Trong nhiều tình huống, sẽ hữu ích để hiển thị một nút kế bên mỗi hàng trong một lưới. Việc nhấp vào nút này sau đó có thể loại bỏ một record, thêm một mục vào giỏ mua sắm, hoặc viện dẫn một cửa sổ khác với nhiều thông tin hơn. DataGridView giúp thực hiện điều này một cách dễ dàng với lớp DataGridViewButtonColumn. Bạn chỉ cần tạo một instance mới, chỉ định text nút và thêm nó vào cuối lưới (grid):

```
' Create a button column.
Dim Details As New DataGridViewButtonColumn( )
Details.Name = "Details"

' Turn off data-binding and show static text.
' (You could use a property from the table by setting
' the DataPropertyName property instead.)
Details.UseColumnTextForButtonValue = False
Details.Text = "Details..."

' Clear the header.
Details.HeaderText = ""

' Add the column.
DataGridView1.Columns.Insert(DataGridView1.Columns.Count, Details)
```

Ngay sau khi bạn đã thực hiện tác vụ đơn giản này, bạn có thể ngăn chặn event CellClick thực hiện một hành động khác (hình 3.14 minh họa kết quả của tiến trình thử đơn giản này):

```
Private Sub DataGridView1_CellClick(ByVal sender As System.Object, _
  ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) _
  Handles DataGridView1.CellClick

    If DataGridView1.Columns(e.ColumnIndex).Name = "Details" Then
      MessageBox.Show("You picked " & _
      DataGridView1.Rows(e.RowIndex).Cells("CustomerID").Value)
```

```
End If

End Sub
```



Hình 3.14. Sử dụng một cột nút

Việc tạo một cột ảnh cũng dễ dàng như vậy. Trong trường hợp này, bạn chỉ cần tạo và thêm một đối tượng DataGridViewImageColumn mới. Nếu bạn muốn hiển thị cùng một ảnh tĩnh trong mỗi ô, bạn chỉ cần xác lập thuộc tính Image với đối tượng Image mà bạn muốn sử dụng.

Một kỹ thuật phức tạp hơn là hiển thị một ảnh riêng biệt cho mỗi record. Bạn có thể lấy record này từ một trường nhị phân trong cơ sở dữ liệu hoặc đọc nó từ một file đã được chỉ định một trường chuỗi. Trong mỗi trường hợp, kỹ thuật này về cơ bản là như nhau. Trước tiên, bạn che giấu cột có chứa dữ liệu thật (thông tin nhị phân thô cho hình ảnh hay đường dẫn đến file) bằng cách xác lập thuộc tính Visible của nó sang False. Sau đó, bạn tạo mục DataGridViewImageColumn mới:

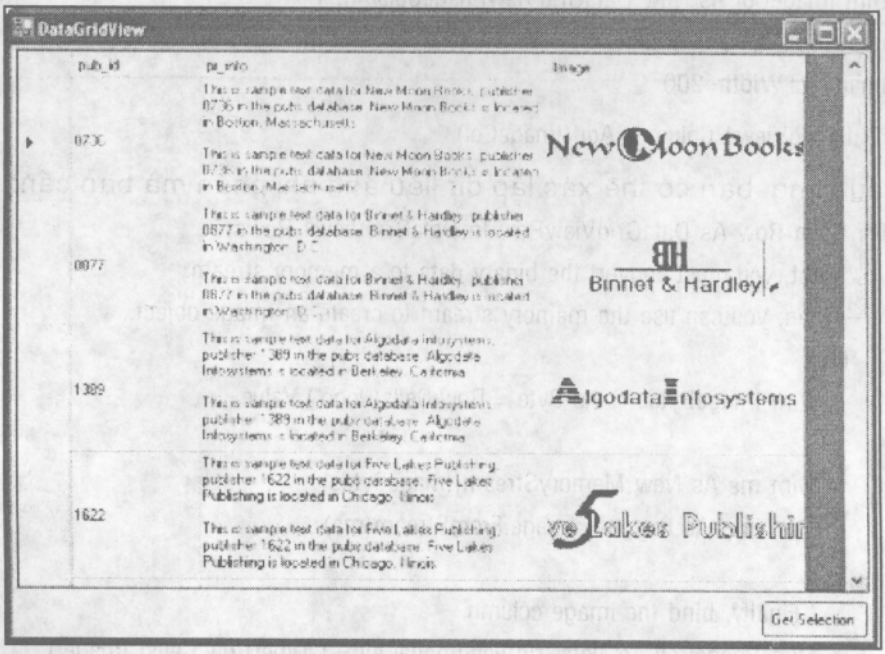
```
DataGridView1.DataSource = ds
DataGridView1.DataMember = "pub_info"

' Hide the binary data.
DataGridView1.Columns("logo").Visible = False

' Add an image column.
```

```
Dim ImageCol As New DataGridViewImageColumn( )
ImageCol.Name = "Image"
ImageCol.Width=200
DataGridView1.Columns.Add(ImageCol)
```

Sau cùng, bạn có thể xác lập dữ liệu ảnh nhị phân mà bạn cần:

```
For Each Row As DataGridViewRow In DataGridView1.Rows
    ' First, you must convert the binary data to a memory stream.
    ' Then, you can use the memory stream to create an Image object.
    Try
        Dim ImageBytes( ) As Byte = Row.Cells("logo").Value

        Dim ms As New MemoryStream(ImageBytes)
        Dim img As Image = Image.FromStream(ms)

        ' Finally, bind the image column.
        Dim ImageCell As DataGridViewImageCell = CType(Row.Cells("Image"), _
          DataGridViewImageCell)
        ImageCell.Value = img

        ' Now you can release the original information to save space.
        Row.Cells("logo").Value = New Byte( ) { }

        Row.Height = 100
    Catch
        ' Ignore errors from invalid images.
    End Try

Next
```

Hình 3.15 minh họa DataGridView với dữ liệu ảnh.



Hình 3.15. Sử dụng một cột ảnh

Ghi chú

Trong nhiều trường hợp, DataGridView đủ thông minh để nhận ra các kiểu dữ liệu ảnh và sử dụng chúng liên tục trong các cột ảnh, không cần chuyển đổi. Tuy nhiên, nếu cần thực hiện thêm (ví dụ, chuyển đổi hay loại bỏ thông tin header dư thừa), bạn cần sử dụng kỹ thuật được nêu ở đây.

Chương 4

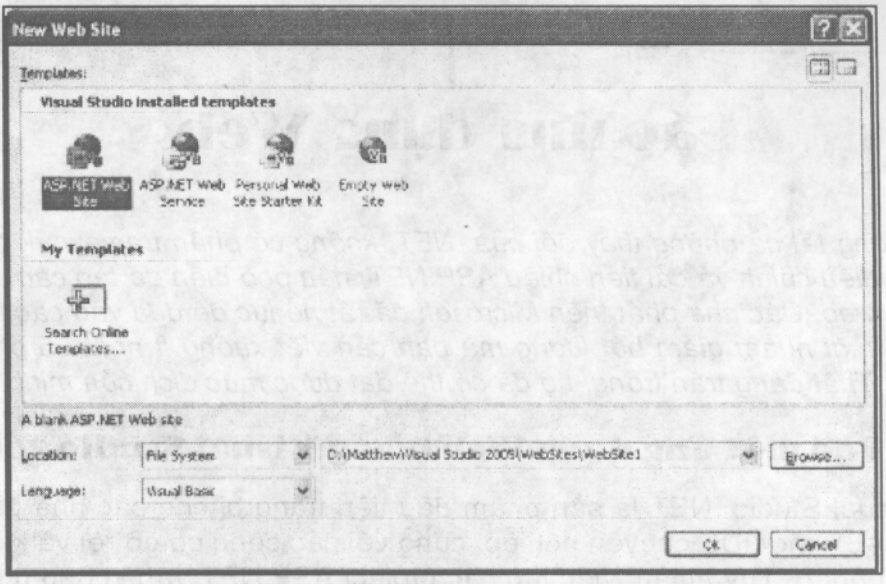
Các ứng dụng Web

Trong tất cả những thay đổi của .NET, không có phần framework nào được điều chỉnh và cải tiến nhiều ASP.NET, nền phổ biến để tạo các ứng dụng web. Các nhà phát triển Microsoft đã rất nỗ lực để đưa vào các tính năng mới nhằm giảm bớt lượng mã bạn cần viết xuống ít nhất 75 phần trăm. Thật đáng trân trọng, họ đã có thể đạt được mục đích của mình.

4.1. Tạo một ứng dụng Web trong Visual Studio 2005

Visual Studio .NET là sản phẩm đầu tiên trang bị cho các nhà phát triển ASP một IDE chuyên nghiệp, cùng với các công cụ gỡ rối và kiểm tra cú pháp. Tuy nhiên, việc tạo các project ASP.NET vẫn không hoàn toàn dễ dàng như tạo các project cho các ứng dụng Windows và console bình thường. Một phần của vấn đề này là quản lý sự tương tác giữa Visual Studio và Internet Information Services (IIS). Thật vui sướng vì Visual Studio 2005 đã cải tiến rất nhiều kinh nghiệm lúc thiết kế cho các ứng dụng web bằng cách đưa ra một phương pháp mới để làm việc với các dự án web.

Để tạo một dự án web trong Visual Studio 2005 hay Visual Studio Web Developer 2005 (Visual Studio 2005 Express không tùy thuộc vào tác vụ), chọn File > New > Web Site, không chọn File > New > Project. Bạn sẽ nhìn thấy hộp thoại New Web Site (xem hình 4.1), trong đó bạn cần chọn nơi mà dự án web sẽ được đặt và ngôn ngữ phát triển của nó. Trong tương lai, bạn cũng có thể mở một dự án (project) mới dựa vào bộ khởi động từ cửa sổ này.



Hình 4.1. Tạo một thư mục ứng dụng web mới

Ghi chú

Hãy quên các thư mục ảo và các vấn đề IIS khác. Visual Studio 2005 (và Visual Web Developer 2005) có một web server cài sẵn và hỗ trợ các web site ít dự án hơn.

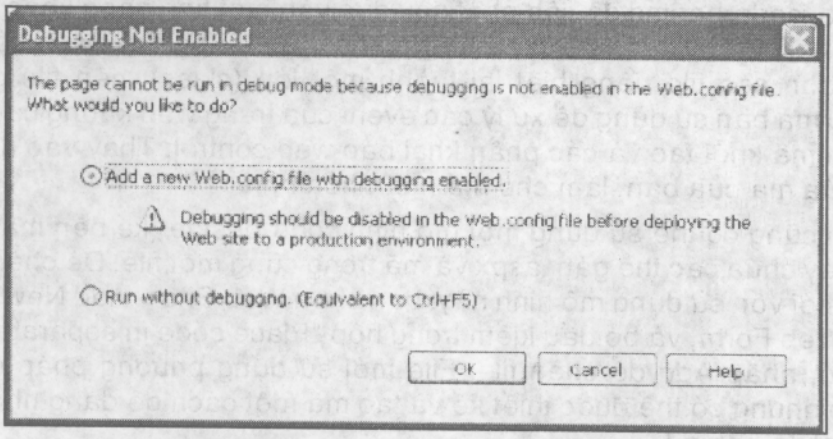
Visual Studio bắt đầu với một thư mục mới chứa hai file: default.aspx (lối vào cho ứng dụng web) và default.aspx.vb (file đằng sau mã). Không có file dự án (.vbproj) hay giải pháp (.sln) nào, các file này giữ cho cấu trúc file đơn giản hơn và giúp dễ dàng triển khai chỉ điều mà bạn cần. Thay vào đó, Visual Studio tự động hiển thị các file và các thư mục con trong thư mục web.

Ghi chú

Khi bạn tạo một dự án ASP.NET mới, Visual Studio tạo thư mục và lưu ngay trang web mặc định. Điều này khác với cách xử lý mà bạn đã thấy với các loại dự án, trong đó Visual Studio tạo một dự án tạm thời cho đến khi bạn lưu nó theo cách tường minh.

Khi bạn bắt đầu làm việc với ứng dụng web, bạn sẽ nhận thấy Visual Studio làm bạn ngạc nhiên hơn. Một là khả năng tự động tạo một file web.config khi bạn cần nó. Để thử điều này, bạn nhấp Start để khởi động ứng dụng cho lần đầu tiên. Vào lúc này, Visual Studio thông báo

rằng bạn không có file web.config. Sau đó nó hỏi bạn có muốn thêm một file tự động vào để mở tính năng gỡ rối hay không (xem hình 4.2).



Hình 4.2. Tự động thêm một file web.config mới

File web.config mà Visual Studio tạo ra thì rõ ràng hơn rất nhiều so với phiên bản được tạo tự động trong Visual Studio .NET 2003. Nó chỉ chứa thông tin bạn cần (trong trường hợp này là các xác lập gỡ rối). Khi bạn thay đổi các cấu hình khác cho ứng dụng của bạn nhờ dùng Visual Studio, các mục khác sẽ được thêm vào tự động. Một lần nữa, Visual Studio tập trung nhấn mạnh vào tính đơn giản và sự thông suốt.

Khi bạn chạy các trang web, web server hợp nhất của Visual Studio khởi động tự động (tìm một biểu tượng nhỏ trong khay hệ thống). Do vậy, bạn không cần sử dụng IIS để kiểm tra một web site. Web server đã thu nhỏ của Visual Studio cung cấp sự bảo vệ tốt hơn bởi vì nó chỉ phục vụ cho những yêu cầu đến từ máy tính cục bộ. Nó cũng thoát ngay khi bạn thoát Visual Studio. Trên hết, nó cho phép bạn tạo các trang web và các dịch vụ web ở nơi bạn muốn, không cần phải bận tâm về việc tạo thư mục ảo phù hợp trước tiên.

Bạn có thể thử tất cả các phần thực hành trong chương này, sử dụng ứng dụng web cơ bản mà bạn đã tạo ở đây.

4.1.2. Mô hình tạo mã trang web

Nếu bạn đã dành nhiều thời gian để lập trình các trang web ASP.NET, bạn có thể nhận thấy có nhiều cách khác nhau để tách mã nguồn với nội dung trực quan. Trong các phiên bản trước của Visual Studio, code-behind (đằng sau mã) là chuẩn duy nhất được hỗ trợ (từng xung đột với xác lập mặc định và loại trừ của Web Matrix, một ứng dụng web Microsoft IDE khác vốn sử dụng mã nội dòng). Tuy nhiên, Visual Studio 2005 hỗ trợ cả hai mô hình tạo mã này.

Theo mặc định, khi bạn thêm các trang web mới, Visual Studio 2005 sử dụng một dạng đơn giản hơn của code-behind. Thay vì sử dụng tính kế thừa, code-behind đã cập nhật này dựa vào một tính năng khác được gọi là partial classes (lớp theo từng phần). Bởi vì partial class có khả năng trộn các file riêng biệt lại thành một lớp (class), nên file code-behind mà bạn sử dụng để xử lý các event của trang web không cần làm nổi bật mã khởi tạo và các phần khai báo web control. Thay vào đó, nó chỉ chứa mã của bạn, làm cho mã ngắn hơn nhiều.

Bạn cũng có thể sử dụng một mô hình code-beside (kế bên mã), mô hình này chứa các thẻ gán .aspx và mã trong cùng một file. Để chèn một trang mới vốn sử dụng mô hình này, bạn chọn Web Site > Add New Item, chọn Web Form, và bỏ dấu kiểm trong hộp "Place code in separate file". Sau đó, nhấp Add để chèn file. File mới sử dụng phương pháp code-beside nhưng có thể được thiết kế và tạo mã một cách dễ dàng như một trang code-behind.

Để hỗ trợ hai mô hình này, Visual Studio cần thay đổi mô hình biên dịch của nó cho các file ASP.NET. Bây giờ, các trang web và các dịch vụ web không được biên dịch cho đến khi bạn truy cập chúng vào lần đầu tiên. (Trong các phiên bản trước của Visual Studio, toàn bộ web site được biên dịch mỗi lần bạn khởi động bằng cách nhấp nút Start). Tuy nhiên, bạn vẫn có thể biên dịch trước ứng dụng của mình sau khi bạn triển khai nó trong một môi trường sản xuất để bảo đảm đạt được hiệu suất tốt nhất cho những nhu cầu đầu tiên. Để làm như vậy, bạn chỉ cần thực thi một yêu cầu cho phần mở rộng precompile.axd trong thư mục ảo gốc ngay khi bạn triển khai ứng dụng của mình. Chẳng hạn, nếu ứng dụng web của bạn được chứa trong thư mục ảo có tên là WebApplication, bạn sẽ sử dụng URL này từ máy tính web server.

Ghi chú

Đừng bị xao lãng bởi việc không có file nào có tên là precompile.axd trong thư mục ảo của bạn. Mọi yêu cầu .axd sẽ viện dẫn các phần mở rộng ASP.NET vốn được cấu hình trong file cấu hình machine.config.

http://localhost/WebApplication/precompile.axd

4.2. Quản lý một ứng dụng Web

Nhiều xác lập điều khiển cách hoạt động của một ứng dụng ASP.NET được tìm thấy trong file web.config, một tài liệu XML đặc biệt được đặt trong thư mục ảo của một ứng dụng web. Trước đây, các nhà phát triển ASP.NET buộc phải hiệu chỉnh các xác lập web.config bằng tay. Nhưng mọi thứ bây giờ trở nên dễ dàng hơn nhờ vào một giao diện đồ họa

ASP.NET 2.0 mới được gọi là Web Site Administration Tool (WAT).

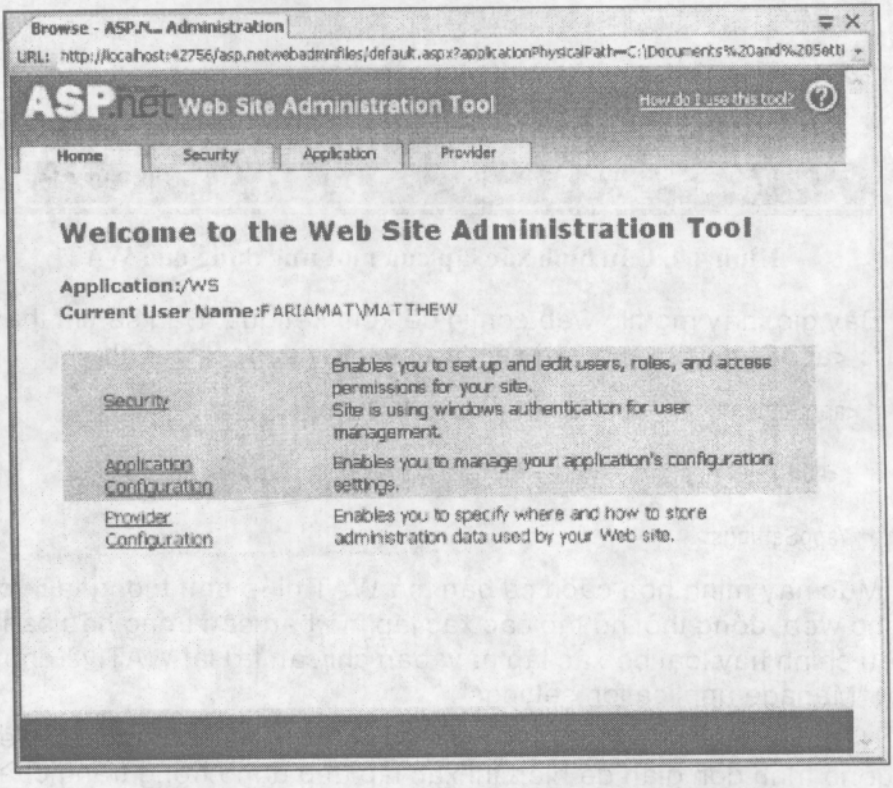
Ghi chú

Nhờ vào Web Site Administration Tools mới, bây giờ bạn không cần hiệu chỉnh file cấu hình web.config bằng tay nữa.

4.2.1 Bạn làm điều đó như thế nào?

Web Site Administration Tool (WAT) được cài đặt trên máy tính của bạn với .NET Framework 2.0. Nó cho phép bạn cấu hình các xác lập ứng dụng web ASP.NET nhờ dùng một trang web chuyên dụng.

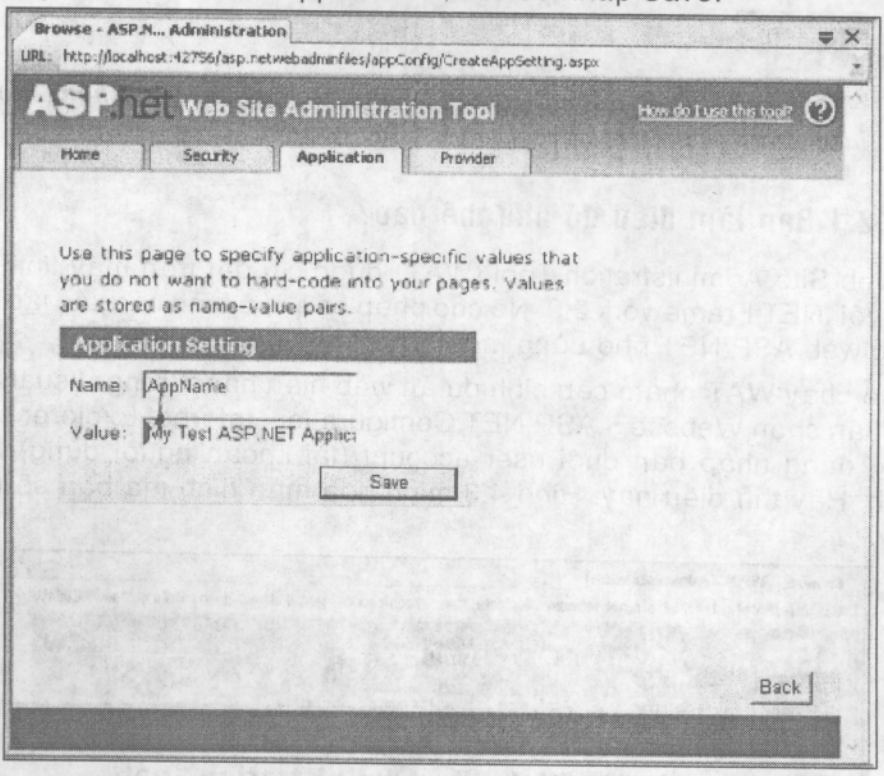
Để chạy WAT nhằm cấu hình dự án web hiện hành trong Visual Studio, bạn chọn Website > ASP.NET Configuration. Internet Explorer sẽ tự động đăng nhập bạn dưới user account (tài khoản người dùng) hiện hành. Hãy thử điều này. Hình 4.3 minh họa màn hình mà bạn sẽ nhìn thấy.



Hình 4.3. Web Site Administration Tool

Để thử WAT, bạn nhấp tab Application và sau đó nhấp liên kết "Create application setting". Hai hộp text sẽ xuất hiện nhằm cho phép bạn

xác định tên và trị của một xác lập mới (xem hình 4.4). Nhập "AppName" và "My Test ASP.NET Application", sau đó nhấp Save.



Hình 4.4. Cấu hình xác lập của một ứng dụng qua WAT

Bây giờ, hãy mở file web.config để xem kết quả. Bạn sẽ tìm thấy một mục <appSettings> mới với xác lập sau đây được xác định:

```
<appSettings>
  <add key="AppName" value="My Test ASP.NET Application" />
</appSettings>
```

Mục này minh họa cách cơ bản mà WAT giúp bạn tương tác với một trang web, đồng thời nó tạo các xác lập mà bạn cần trong hậu cảnh. Để hiệu chỉnh hay loại bỏ xác lập này, bạn chỉ cần trở lại WAT và chọn liên kết "Manage application settings".

Nếu bạn muốn, bạn có thể hoàn chỉnh ví dụ này bằng cách viết một thường trình đơn giản để hiển thị xác lập ứng dụng trong trang của bạn. Chỉ cần thêm một label control vào trang web của bạn và chèn mã sau đây trong event handler Page_Load ( )

```
Label1.Text = "You are running " & _

    ConfigurationSettings.AppSettings("AppName")
```

Dĩ nhiên, việc sử dụng WAT để tạo các xác lập ứng dụng chỉ mới là phần bắt đầu. Bạn cũng có thể sử dụng WAT để thực hiện các tác vụ sau:

Security

Sử dụng tab này để xác lập chế độ xác thực (authentication mode), định ra các quy tắc tạo và quản lý người dùng (user). Bạn sẽ tìm hiểu về tab này trong mục "Xác thực các User một cách dễ dàng".

Application

Sử dụng tab này để thiết lập các xác lập ứng dụng (như minh họa trong phần thực hành ở đây) và cấu hình các bộ đếm web site cũng như gỡ rối.

Provider

Sử dụng tab này để cấu hình nơi chứa thông tin vai trò của user và sự cá nhân hóa. Theo mặc định, ASP.NET sử dụng Access để chứa thông tin này trong AspNetDB.mdb trong thư mục con App_Data (trong Beta 1) hoặc trong một cơ sở dữ liệu SQL Server (trong Beta 2).

4.2.2. Thế còn...

... việc thực hiện các thay đổi cho các xác lập cấu hình của một ứng dụng web bằng lập trình thì sao? Một cách ấn tượng, ASP.NET có chứa một hệ thống các lớp bao quát cho mục đích này trong các namespace System.Web.Configuration và System.Web.Administration. Bạn có thể sử dụng các lớp này để truy xuất hay thay đổi các xác lập ứng dụng web trong trang web của bạn hay trong mã dịch vụ web. Thật vậy, toàn bộ Web Site Administration Tool được viết ở dạng một ứng dụng ASP.NET và bạn sẽ tìm thấy mã nguồn (trong C#) trong thư mục sau đây:

```
c:\[Windows Directory]\Microsoft.NET\Framework\[Version]\
ASP.NETWebAdminFiles
```

4.3. Kết buộc với dữ liệu không cần viết mã

Đa số các ứng dụng web quan trọng đều cần truy xuất các record (bản ghi) từ một cơ sở dữ liệu. Trong .NET, framework truy cập cơ sở dữ liệu được lựa chọn là ADO.NET. Tuy nhiên, việc viết mã để thực hiện các hoạt động cơ sở dữ liệu phổ biến với ADO.NET, như mở một nối kết và tìm về một bảng kết quả, có thể lặp lại và gây nhàm chán, điều mà các lập trình viên VB không mong đợi. Để đơn giản những tác vụ như

vậy, ASP.NET 2.0 giới thiệu một vài control nguồn dữ liệu mới nhằm giúp đơn giản tác vụ truy xuất dữ liệu và kết buộc nó với một trang web.

Ghi chú

Với các control cung cấp dữ liệu (data provider) ASP.NET mới, bạn có thể tạo và kết buộc toàn bộ mã cơ sở dữ liệu của bạn vào lúc thiết kế, không cần viết một dòng mã nào.

4.3.1 Bạn làm điều đó như thế nào?

Để sử dụng control nguồn dữ liệu (data source) ASP.NET 2.0 mới, tất cả những gì bạn cần làm là rê nó từ hộp công cụ Visual Studio đến một trang web, cấu hình một vài thuộc tính của nó và sau đó kết buộc nó với các control khác vốn hiển thị dữ liệu mà nó đưa ra. Khi bạn chạy trang web, control nguồn dữ liệu thực hiện công việc chuyển nặng nề, liên hệ cơ sở dữ liệu của bạn và truy xuất các hàng bạn cần.

ASP.NET đã kèm theo nhiều control nguồn dữ liệu và nhiều control đã được hoạch định. Tuy danh sách này thay đổi giữa các lần tạo nhưng phiên bản mới nhất bao gồm:

SqlDataSource

Tương tác với một cơ sở dữ liệu SQL Server (Phiên bản 7.0 hoặc mới hơn).

XmlDataSource

Tương tác với dữ liệu XML từ một file hay một nguồn dữ liệu khác.

ObjectDataSource

Tương tác với một đối tượng tùy ý mà bạn tạo. Mục kế tiếp "Kết buộc các Web Control với một lớp tùy ý" sẽ trình bày chi tiết hơn về kỹ thuật này.

Để thử kết buộc dữ liệu không cần tạo mã, hãy rê một SqlDataSource mới sang bề mặt thiết kế của một trang web từ một Data của hộp công cụ. Sau đó, nhấp thẻ thông minh của control và chọn Configure Data Source. Visual Studio sẽ đưa bạn qua một wizard ngắn để bạn chỉ định chuỗi nối kết cho cơ sở dữ liệu của bạn (mà sau đó sẽ được xác lập trong thuộc tính ConnectionString) và mẫu truy vấn (query) bạn muốn thực hiện (mà sau đó sẽ được xác lập trong thuộc tính SelectCommand). Tìm server cơ sở dữ liệu này và chọn cơ sở dữ liệu Northwind. Tuy bạn có thể tạo mẫu truy vấn động bằng cách chọn các cột trong một table, nhưng đối với ví dụ này bạn chỉ cần chỉ định chuỗi SQL "SELECT ContactName FROM Customers".

Ghi chú

Các nguồn dữ liệu khác được hoạch định để cho phép truy xuất dễ dàng mọi thứ từ các danh sách thư mục đến dữ liệu dịch vụ web.

Khi bạn đã hoàn tất, Visual Studio sẽ đã thêm một thẻ control SqlDataSource vào trang web của bạn, có dạng như sau:

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"
 SelectCommand="SELECT ContactName FROM Customers"
 ConnectionString=
 "Data Source=127.0.0.1;Integrated Security=SSPI;Initial Catalog=Northwind">
</asp:SqlDataSource>
```

Nguồn dữ liệu này định nghĩa một nối kết với cơ sở dữ liệu Northwind và một hoạt động Select truy xuất một danh sách tất cả các tên liên hệ trong bảng Customers.

Ghi chú

Để chỉnh sửa bất kỳ control ASP.NET, bạn có hai lựa chọn. Bạn có thể chọn nó và thực hiện sự thay đổi trong cửa sổ Properties, hoặc bạn có thể chuyển sang khung xem Source và hiệu chỉnh thẻ control.

Việc kết buộc một control với nguồn dữ liệu của bạn thì dễ thực hiện. Để thử điều này, bạn rê một control BulletedList sang trang web của bạn, bạn có thể sử dụng nó để hiển thị danh sách các tên liên hệ từ bảng Customers. Nhấp thẻ thông minh và chọn Connect to Data Source. Sau đó, bạn có thể chọn nguồn dữ liệu mà bạn muốn sử dụng (đây là nguồn dữ liệu được tạo ở bước cuối) và trường mà bạn muốn hiển thị (nghĩa là ContactName).

Sau đây là thẻ gán đã hoàn tất:

```
<asp:BulletedList ID="BulletedList1" Runat="server"
  DataSourceID="SqlDataSource1"
  DataTextField="ContactName">
</asp:BulletedList>
```

Thật là lạ, hai phần khai báo control này là tất cả những gì bạn cần để tạo trang kết buộc dữ liệu này. Khi bạn chạy trang, BulletedList sẽ yêu

cầu dữ liệu từ SqlDataSource, vốn sẽ tìm nó về từ cơ sở dữ liệu nhờ sử dụng mẫu truy vấn bạn đã xác định. Bạn không cần viết bất kỳ dòng mã nào.

Phức tạp hơn, bạn có thể sử dụng một control khác để lọc danh sách những người cần liên hệ (contacts) theo một tiêu chuẩn nào đó, như những người đó sống ở quốc gia nào. Điều này làm phát sinh một vấn đề mới, nghĩa là làm thế nào để cập nhật mẫu truy vấn trong SqlDataSource.SelectCommand theo trị đã được nhập vào control kia.

ASP.NET giải quyết vấn đề này một cách gọn gàng với các thông số (parameter). Để thử điều này, hãy bắt đầu bằng cách thêm một nguồn dữ liệu mới vốn truy nạp một danh sách các quốc gia mà khách hàng đang sinh sống từ cơ sở dữ liệu. Sau đây là một ví dụ trong trường hợp này:

```
<asp:SqlDataSource ID="Countries" Runat="server" ConnectionString="..."

  SelectCommand="SELECT DISTINCT Country FROM Customers">

</asp:SqlDataSource>
```

Kế tiếp, thêm một control DropDownList có tên là lstCountries để cho ra danh sách quốc gia. Bạn có thể sử dụng cùng phương pháp này như khi bạn làm với BulletedList, hoặc bạn có thể gõ thẻ gán bằng tay. Sau đây là thẻ gán đã hoàn chỉnh mà bạn cần:

```
<asp:DropDownList ID="lstCountries" Runat="server"

  DataValueField="Country" DataTextField="Country"

  DataSourceID="Countries" AutoPostBack="True">

</asp:DropDownList>
```

Bây giờ bạn có thể chỉnh sửa mẫu truy vấn tạo danh sách các khách hàng. Trước tiên, bạn chèn một thông số đã được đặt tên vào mẫu truy vấn của bạn. Nhớ đặt một dấu @ở đầu tên thông số để SqlDataSource có thể nhận ra nó. Trong ví dụ này, hãy dùng @Country. (@biểu thị một thông số đã được đặt tên khi sử dụng provider SQL Server).

Sau đây là diện mạo của thẻ gán nguồn dữ liệu đã sửa đổi:

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"

  SelectCommand="SELECT ContactName FROM Customers WHERE Country='@Country'

  ...

</asp:SqlDataSource>
```

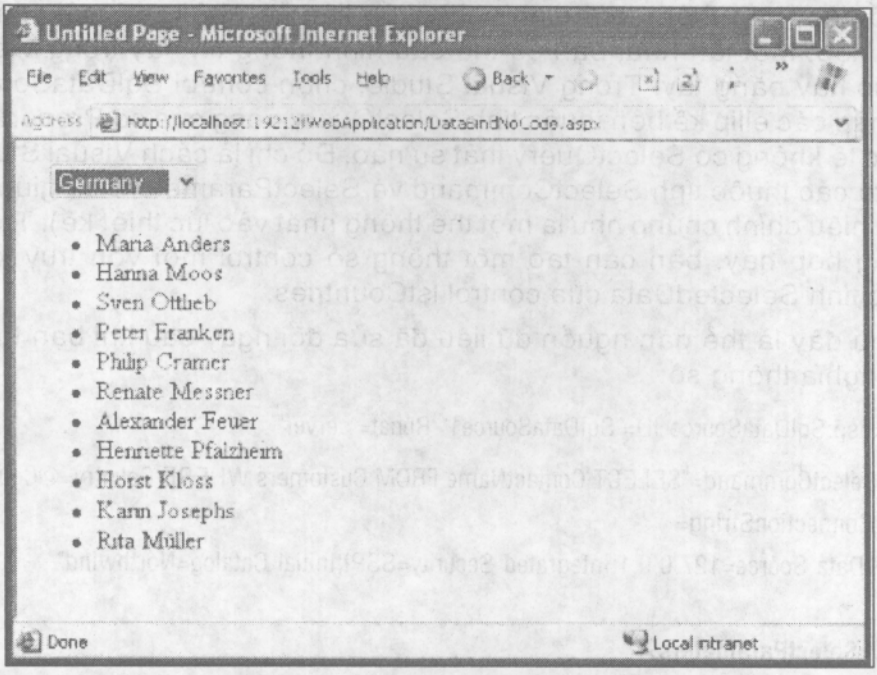
Kế tiếp, bạn thêm một định nghĩa vốn liên kết thông số với control thích hợp. Một lần nữa, bạn có thể cấu hình thông tin này trong Visual Studio hay bằng tay. Trong Visual Studio, chọn control SqlDataSource và nhấp các ellip kế bên thuộc tính SelectQuery trong cửa sổ Properties. (Thực tế không có SelectQuery thật sự nào. Đó chỉ là cách Visual Studio đưa ra các thuộc tính SelectCommand và SelectParameters để giúp dễ dàng hiệu chỉnh chúng như là một thể thống nhất vào lúc thiết kế). Trong trường hợp này, bạn cần tạo một thông số control mới vốn truy xuất thuộc tính SelectedData của control lstCountries.

Sau đây là thẻ gán nguồn dữ liệu đã sửa đổi ngay sau khi bạn thêm định nghĩa thông số:

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"
SelectCommand="SELECT ContactName FROM Customers WHERE Country=@Country"
ConnectionString=
"Data Source=127.0.0.1;Integrated Security=SSPI;Initial Catalog=Northwind"

 <SelectParameters>
  <asp:ControlParameter Name="Country"
   ControlID="lstCountries" PropertyName="SelectedValue">
  </asp:ControlParameter>
 </SelectParameters>
</asp:SqlDataSource>
```

Lưu ý rằng tên của thông số control trùng với tên của thông số trong biểu thức SQL, với một khác biệt nhỏ: dấu a@đứng đầu luôn bị loại bỏ.

Hình 4.5 minh họa trang đã hoàn chỉnh. Khi bạn chọn một quốc gia từ danh sách xổ xuống, danh sách khách hàng được đánh dấu bullet sẽ được tự động làm mới với các khách hàng phù hợp. Bây giờ bạn đã có được vài tính năng mà vẫn chưa phải viết bất kỳ mã nào.



Hình 4.5. Các control kết buộc dữ liệu liên kết không cần mã

Ví dụ này đã đưa ra các khả năng mà bạn có thể sử dụng nhiều control nguồn dữ liệu. Chẳng hạn, hãy tưởng tượng bạn muốn cung cấp một khung xem chính - chi tiết của các đơn đặt hàng từ khách hàng. Bạn có thể sử dụng một nguồn dữ liệu để điền các khách hàng vào một hộp danh sách. Khi người dùng chọn một khách hàng, bạn có thể sử dụng nguồn dữ liệu khác để thực hiện một mẫu truy vấn cho các đơn đặt hàng đã liên kết và hiển thị nó trong một control khác.

4.3.2. Thế còn...

... các lý do để không sử dụng các control kết buộc dữ liệu không cần mã mới thì sao? Nhiều nhà phát triển có suy nghĩ đúng đắn tránh sử dụng các kỹ thuật kết buộc dữ liệu này bởi vì họ nhúng các chi tiết cơ sở dữ liệu vào mã giao diện người dùng (user-interface code). Thật vậy, đây đúng là điều mà ví dụ này đưa ra, có những nhược điểm làm ảnh hưởng đến khả năng bảo trì sự tối ưu hóa và gỡ rối. Nói một cách đơn giản, với các định danh cơ sở dữ liệu được nhúng ở mọi nơi trong một site lớn, thì khó có thể giữ được sự nhất quán.

Các nhà phát triển ASP.NET đã không quên mặt này của vấn đề. Chỉ cần một chút thận trọng, bạn có thể sử dụng các data source provider mà vẫn tập trung hóa logic cơ sở dữ liệu của bạn. Một trong những cách tốt nhất để làm điều này là sử dụng control ObjectDataSource, vốn cho

phép bạn liên kết với một lớp tùy ý mã bạn đã tạo với mã truy cập dữ liệu. Mục kế tiếp "Kết buộc các Web Control với một lớp tùy ý" sẽ trình bày về kỹ thuật này.

Các nguồn dữ liệu (data source) cũng cung cấp một nơi hữu ích để thêm tính năng cao cấp hơn. Một trong những ví dụ hấp dẫn nhất là caching (lưu trữ truy cập nhanh). Nếu bạn xác lập EnableCaching sang true, control nguồn dữ liệu sẽ tự động chèn dữ liệu đã truy xuất vào các ASP.NET và sử dụng lại nó trong các yêu cầu sau này, có khả năng làm giảm tải cơ sở dữ liệu. Bạn có thể cấu hình khoảng thời gian mà một mục được truy cập nhanh bằng cách xác lập các thuộc tính CacheDuration và CacheExpirationPolicy.

4.4. Kết buộc các Web Control với một lớp tùy ý

Các ứng dụng được thiết kế tốt thường phân biệt logic truy cập dữ liệu với phần còn lại của mã. Trong ASP.NET 2.0, bạn có thể đạt được sự phân biệt này trong khi vẫn sử dụng các control nguồn dữ liệu ASP.NET mới để tiện kết buộc dữ liệu lúc thiết kế mà không cần mã. Bí quyết là sử dụng control ObjectDataSource mới, control này biết cách tìm về kết quả từ một lớp truy cập dữ liệu (data access class). Sau đó, bạn có thể kết buộc các control khác với ObjectDataSource để dễ dàng hiển thị nhanh trang web.

Ghi chú

Bạn muốn sử dụng kỹ thuật kết buộc nguồn dữ liệu mà không phân tán các chi tiết cơ sở dữ liệu qua suốt hàng chục trang web? Control ObjectDataSource cung cấp giải pháp này.

4.4.1 Bạn làm điều đó như thế nào?

Để sử dụng control ObjectDataSource, trước tiên bạn phải tạo một lớp tùy ý truy xuất dữ liệu từ cơ sở dữ liệu. Lớp cơ sở dữ liệu sẽ chứa một phương thức cho mọi hoạt động cơ sở dữ liệu mà bạn muốn thực hiện. Các phương thức truy xuất kết quả từ cơ sở dữ liệu có thể trả về các đối tượng DataTable hay DataSet, các tập hợp hay các lớp tùy ý.

Ví dụ 4.1 minh họa một lớp cơ sở dữ liệu (database class) được gọi là CustomerDB, lớp này cung cấp một phương thức GetCustomers ( ) đơn. Phương thức GetCustomers ( ) truy vấn cơ sở dữ liệu và trả về một tập hợp các đối tượng CustomerDetails. Đối tượng CustomerDetails cũng là một đối tượng tùy ý. Nó chỉ bao hàm mọi chi tiết của một record customer (khách hàng) từ cơ sở dữ liệu.

Ví dụ 4.1. Một lớp cơ sở dữ liệu tùy ý

```
Imports System.Data.SqlClient
Imports System.Collections.Generic

Public Class CustomerDB

    Private ConnectionString As String = _
    "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=SSPI"

    Public Function GetCustomers( ) As List(Of CustomerDetails)
        Dim Sql As String = "SELECT * FROM Customers"

        Dim con As New SqlConnection(ConnectionString)
        Dim cmd As New SqlCommand(Sql, con)
        Dim Reader As SqlDataReader
        Dim Customers As New List(Of CustomerDetails)
        Try
            con.Open( )
            Reader = cmd.ExecuteReader( )
            Do While Reader.Read( )
                Dim Customer As New CustomerDetails( )
                Customer.ID = Reader("CustomerID")
                Customer.Name = Reader("ContactName")
                Customers.Add(Customer)
            Loop
        Catch Err As Exception
            Throw New ApplicationException( _
             "Exception encountered when executing command.", Err)
        Finally
            con.Close( )
        End Try

        Return Customers
    End Function

End Class
```

```
Public Class CustomerDetails

    Private _ID As String
    Private _Name As String

    Public Property ID( ) As String
        Get
            Return _ID
        End Get
        Set(ByVal Value As String)
            _ID = Value
        End Set
    End Property

    Public Property Name( ) As String
        Get
            Return _Name
        End Get
        Set(ByVal Value As String)
            _Name = Value
        End Set
    End Property

End Class
```

Có vài điểm quan trọng cần lưu ý về ví dụ này. Trước tiên, lớp cơ sở dữ liệu phải không có trạng thái để hoạt động chính xác. Nếu bạn cần bất kỳ thông tin, hãy truy xuất nó từ các xác lập ứng dụng tùy ý trong file web.config. Thứ hai, chú ý cách lớp CustomerDetails sử dụng các thủ tục thuộc tính thay vì các biến thành viên chung (public member variables). Nếu bạn sử dụng các biến thành viên chung, ObjectDataSource sẽ không thể truy xuất thông tin từ lớp và kết buộc với nó.

Thủ thuật

Ví dụ 4.1 sử dụng một tập hợp chung. Để tìm hiểu thêm thông tin về tính năng CLR mới, hãy xem mục 2.5 ở chương 2.

Để sử dụng lớp truy cập dữ liệu tùy ý trong tình huống kết buộc dữ liệu, trước tiên bạn cần làm cho nó trở thành một thành phần của ứng dụng web. Bạn có hai tùy chọn:

- Đặt nó trong một dự án thư viện lớp riêng biệt rồi biên dịch nó sang một file DLL. Sau đó, trong ứng dụng web, thêm một phần tham chiếu vào hợp ngữ này. Visual Studio sang thư mục con Bin của ứng dụng web.
- Đặt mã nguồn trong một file .vb bình thường trong thư mục con App_Code của ứng dụng web. ASP.NET tự động biên dịch bất kỳ mã nguồn không nằm trong thư mục này và làm cho nó có sẵn với ứng dụng web của bạn. (Để bảo đảm nó đã được biên dịch, chọn Build > Build Website trước khi đi xa hơn).

Ngay sau khi bạn đã thực hiện một trong các bước này, rê một ObjectDataSource từ tab dữ liệu của hộp công cụ Visual Studio sang bề mặt thiết kế của một trang web. Nhấp thẻ thông minh của control và chọn Configure Data Source. Một wizard sẽ xuất hiện để bạn chọn lớp của mình từ một danh sách xổ xuống (một bước xác lập thuộc tính TypeName) và hỏi bạn muốn gọi phương thức nào khi thực hiện một mẫu truy vấn (vốn xác lập thuộc tính MethodName).

Sau đây là diện mạo của thẻ control ObjectDataSource đã hoàn chỉnh trong trang .aspx của ví dụ này:

```
<asp:ObjectDataSource ID="ObjectDataSource1" Runat="server"

  TypeName="CustomerDB" SelectMethod="GetCustomers">

</asp:ObjectDataSource>
```

Bây giờ bạn có thể kết buộc các control khác với các thuộc tính của lớp CustomerDetails. Chẳng hạn, BulletedList này đưa ra thông tin CustomerDetails.Name cho mỗi đối tượng trong tập hợp:

```
<asp:BulletedList ID="BulletedList1" Runat="server"

  DataTextField="Name" DataSourceID="ObjectDataSource1">

</asp:BulletedList>
```

Khi bạn chạy ứng dụng, BulletedList yêu cầu dữ liệu từ ObjectDataSource. ObjectDataSource tạo một instance của lớp CustomerDB, gọi GetCustomers ( ) và trả về dữ liệu.

4.4.2. Thế còn...

... việc cập nhật một cơ sở dữ liệu thông qua một ObjectDataSource thì sao? Không vấn đề gì. Cả hai control ObjectDataSource và SqlDataSource đã được trình bày trong mục trước "Kết buộc với dữ liệu không cần viết mã" đều hỗ trợ việc chèn, cập nhật và xóa các record. Với SqlDataSource, bạn chỉ cần xác lập các thuộc tính như DeleteCommand, InsertCommand và UpdateCommand với SQL thích hợp. Với ObjectDataSource, bạn xác lập các thuộc tính như DeleteMethod, InsertMethod và UpdateMethod bằng cách chỉ định các tên phương thức phù hợp trong lớp truy cập dữ liệu tùy ý của bạn. Trong nhiều trường hợp, bạn cũng sẽ cần chỉ định thêm thông tin nhờ dùng các thông số, vốn có thể ánh xạ sang các control khác, các đối số chuỗi truy vấn hay thông tin session (phiên làm việc). Chẳng hạn, bạn có thể muốn xóa record đang được chọn hay cập nhật một record dựa vào các trị trong các hộp text. Để thực hiện điều này, bạn cần thêm các thông số, như được trình bày trong mục trước "Kết buộc với dữ liệu không cần viết mã".

Ngay sau khi bạn đã cấu hình các hoạt động này (hoặc bằng tay hoặc bằng cách sử dụng các wizard tiện lợi lúc thiết kế). Bạn có thể kích khởi chúng bằng cách gọi các phương thức Delete ( ), Insert ( ), và Update ( ). Các control khác cài vào framework điều khiển nguồn dữ liệu cũng có thể sử dụng các phương thức này. Chẳng hạn, nếu bạn cấu hình một đối tượng SqlDataSource với thông tin mà nó cần để cập nhật các record, bạn có thể kích hoạt tính năng hiệu chỉnh GridView mà không cần thêm một dòng mã nào. Bạn sẽ xem một ví dụ về kỹ thuật này với control DetailsView trong mục sắp tới "Hiển thị mỗi lần một Record".

4.5. Hiển thị các Table tương tác không cần viết mã

Control DataGrid ASP.NET 1.0 và 1.1 đã rất phổ biến, nhưng việc thực thi một số tính năng mong muốn nhất của nó thường đòi hỏi phải viết nhiều mã. Chẳng hạn, nếu bạn muốn cho phép người dùng di chuyển qua các hàng dữ liệu, bạn phải truy vấn cơ sở dữ liệu sau mỗi lần trở lại, truy xuất trang được yêu cầu và xác lập dãy hàng mà bạn muốn hiển thị. Với control GridView mới, các vấn đề này chỉ là việc của quá khứ.

Ghi chú

Control GridView mới cho phép bạn tạo và hiển thị các table dữ liệu mà người dùng có thể phân loại, truy cập và hiệu chỉnh không đòi hỏi bạn phải viết bất kỳ một dòng mã nào.

Để chuẩn bị cho ASP.NET 2.0, các kỹ sư Microsoft đã chọn không tung ra một phiên bản mới của DataGrid hiện hành để làm đơn giản tính

tương thích ngược. Thay vào đó, control GridView sẽ sao lặp và mở rộng tính chức năng của DataGrid, trong khi làm cho các tính năng của nó có sẵn với các nhà phát triển thông qua một mô hình lập trình đơn giản hơn nhiều.

4.5.1 Bạn làm điều đó như thế nào?

Để sử dụng control GridView, hãy rê nó từ mục Data của hộp công cụ Visual Studio sang bề mặt thiết kế của một trang web. Đối với sự kết buộc dữ liệu không cần viết mã, bạn có thể thêm một control SqlDataSource (được trình bày trong mục "Kết buộc với dữ liệu không cần viết mã") hay sử dụng một control ObjectDataSource kết hợp với một đối tượng truy cập dữ liệu tùy ý, như được trình bày trong mục "Kết buộc các Web Control với một lớp tùy ý". Trong trường hợp, chúng ta sẽ sử dụng một control SqlDataSource và mẫu truy vấn select (chọn) được minh họa ở đây để truy xuất tất cả các trường và các record trong table Customers của cơ sở dữ liệu Northwind. Sau đây là thẻ gán nguồn dữ liệu sau cùng:

```
<asp:SqlDataSource ID="CusomtersList" Runat="server"
  SelectCommand="SELECT * FROM Customers"
  ConnectionString=
  "Data Source=127.0.0.1;Integrated Security=SSPI;Initial Catalog=Northwind">
</asp:SqlDataSource>
```

Bây giờ, hãy xác lập thuộc tính GridView.DataSourceID sang tên của SqlDataSource (trong ví dụ này là CustomerList). Điều này sẽ kết buộc GridView với SqlDataSource.

Vào lúc này, bạn có thể chạy trang của mình và xem một table HTML đơn giản với danh sách đầy đủ khách hàng. Tuy nhiên, để tinh chỉnh diện mạo của table, bạn cần thực hiện thêm vài bước nữa. Các bước này là:

Ghi chú

Bạn có thể nhìn thấy các cột của lưới vào lúc thiết kế. Nếu không, chọn Refresh Schema trên thẻ thông minh SqlDataSource (để lấy thông tin cột từ cơ sở dữ liệu) và sau đó chọn Refresh Schema trên thẻ thông minh GridView.

- Xác lập thuộc tính Font để sử dụng một font hấp dẫn hơn. Một lựa chọn phổ biến được hỗ trợ bởi hầu hết các trình duyệt web là Verdana (sử dụng một cỡ X-Small hay XX-Small).

- Áp dụng một kiểu định dạng với các style. Bạn có thể xác lập các màu sắc, font, và kích cỡ cho FooterStyle, HeaderStyle, RowStyle và nhiều hơn nữa nhờ dùng cửa sổ Properties. Hoặc để thay đổi nhanh toàn bộ diện mạo, nhấp thẻ thông minh GridView và chọn AutoFormat. Khi bạn chọn một trong những xác lập đã có sẵn, mọi style GridView sẽ được xác lập tự động.

Việc thay đổi diện mạo GridView chỉ là một phần của công việc. Bạn cũng có thể chuyển sang các tính năng GridView khác nhờ dùng các tùy chọn trong thẻ thông minh GridView. Sau đây là một số liên kết bạn có thể nhấp vào để có được kết quả nhanh chóng:

Enable Paging

Tùy chọn này xác lập thuộc tính AllowPaging sang true. GridView sau đó sẽ tách các danh sách record dài ra thành các trang riêng biệt (mỗi trang có số hàng được chỉ định trong thuộc tính PageSize). Người dùng có thể chuyển từ trang này sang trang kia bằng cách nhấp vào các liên kết được đánh số xuất hiện ở cuối GridView.

Enable Sorting

Tùy chọn này xác lập AllSorting sang TRue. GridView sau đó sẽ cung cấp các hyperlink (siêu liên kết) cho cột. Khi người dùng nhấp vào một hyperlink, toàn bộ table sẽ được sắp xếp lại theo thứ tự bảng chữ cái (hoặc theo thứ tự số tăng dần) tùy theo cột đó.

Enable Selection

Tùy chọn này thêm một liên kết Select trong một cột mới ở phía bên trái của lưới. Người dùng có thể nhấp vào liên kết này để chọn hàng (vào lúc này thuộc tính SelectedIndex sẽ được xác lập theo cho phù hợp).

Enable Deleting

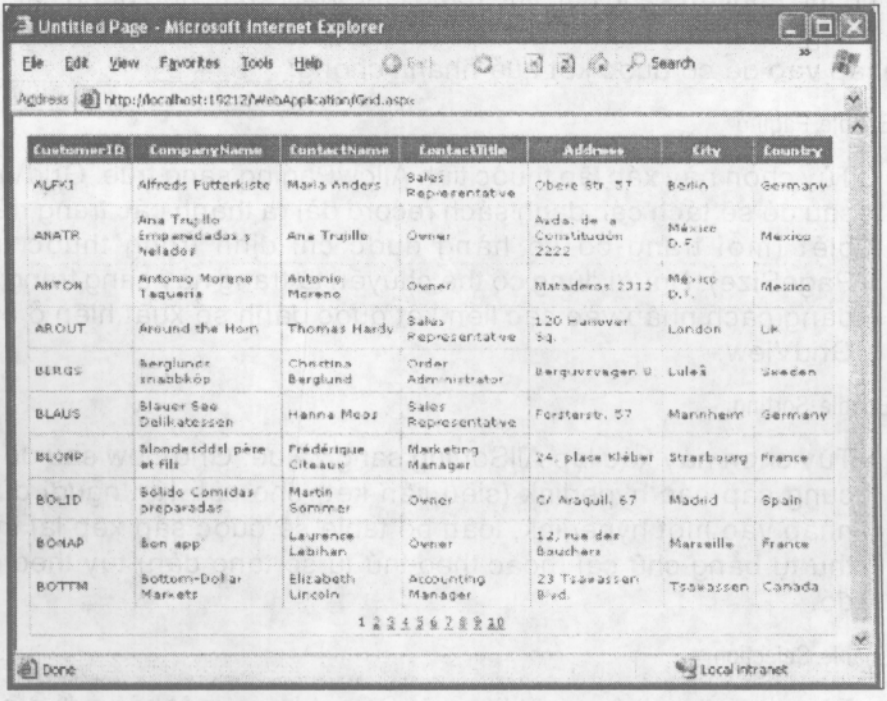
Tùy chọn này thêm một liên kết Delete trong một cột mới ở phía bên trái của lưới. Người dùng có thể nhấp liên kết này để xóa hàng khỏi cơ sở dữ liệu. Bạn sẽ chỉ nhìn thấy tùy chọn này nếu bạn đã định nghĩa một DeleteCommand cho nguồn dữ liệu đã gán.

Enable Editing

Tùy chọn này thêm một liên kết Edit trong một cột mới ở phía bên trái của lưới. Người dùng có thể nhấp liên kết này để đặt một

hàng trong chế độ hiệu chỉnh (vào lúc này một liên kết Update and Cancel sẽ xuất hiện, cho phép bạn đẩy sự thay đổi sang cơ sở dữ liệu hay cuộn lại nó). Bạn sẽ chỉ nhìn thấy tùy chọn này nếu bạn đã định nghĩa một UpdateCommand cho nguồn dữ liệu đã gán.

Hình 4.6 minh họa một table hỗ trợ sự phân trang và phân loại theo cột, vốn được tạo bởi GridView mà không cần sử dụng bất kỳ dòng mã tùy ý nào.



Hình 4.6. Một GridView với tính năng phân loại và phân trang được kích hoạt

4.5.2. Tin chỉnh sự hiển thị GridView

Chẳng hạn, bạn có thể muốn điều chỉnh thứ tự phân loại, text được sử dụng cho các liên kết chọn và hiệu chỉnh, các tựa cột hay thứ tự của các cột. Bạn cũng có thể cần xác lập các chuỗi text và định dạng mặc định. Để thực hiện bất kỳ tác vụ này, bạn chỉ cần tùy biến các đối tượng cột mà GridView tạo dựa vào kiểu định dạng của các record nguồn dữ liệu. Cách dễ nhất để làm điều này là chọn liên kết Edit Columns trên thẻ thông minh GridView và sử dụng hộp thoại Fields để tùy biến các thuộc tính của mỗi cột. Hãy thử điều này.

4.6. Hiển thị mỗi lần một Record

Trong khi control GridView là một công cụ hoàn hảo để trình bày các record của một cơ sở dữ liệu ở dạng các hàng dữ liệu trong một table, nó trở nên ít tiện lợi hơn khi bạn có các record với nhiều trường (nhất là khi một số trường khá dài), và bạn muốn cho phép người dùng xử lý hay thêm vào dữ liệu chứa trong chúng. Một giải pháp là hiển thị chỉ các trường được chọn trong một lưới, nhưng có những lúc bạn cần hiển thị toàn bộ một record trên một trang và cho phép người dùng vừa hiệu chỉnh các record riêng lẻ vừa thêm các record mới vào cơ sở dữ liệu. Trong ASP.NET, DetailsView mới cung cấp cho bạn mọi tính năng bạn cần để xử lý các record riêng lẻ một cách dễ dàng (nghĩa là không cần viết mã).

Ghi chú

Control DetailsView mới đem lại cho bạn một cách tiện lợi để cho phép người dùng xem, hiệu chỉnh, chèn và xóa các record riêng lẻ.

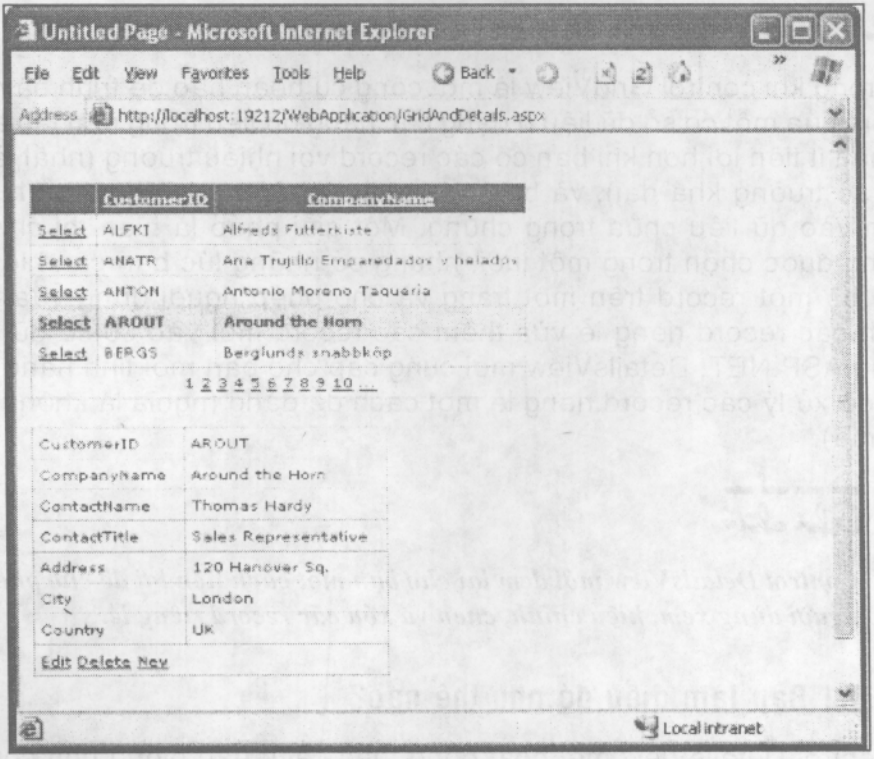
4.6.1 Bạn làm điều đó như thế nào?

Control DetailsView mới hoạt động theo cách gần giống như control GridView đã mô tả trong mục trước "Hiển thị các Table tương tác không cần viết mã". Điểm khác biệt là DetailsView hiển thị mỗi lần chỉ một record. Theo mặc định, tất cả các trường được hiển thị trong một table, mỗi trường trong một hàng của riêng nó, liệt kê từ trên xuống dưới.

Để thêm một DetailsView vào một trang web, bạn chỉ cần rê nó sang bề mặt thiết kế từ tab Visual Studio Toolbox Data. Kế tiếp, hãy nhấp vào thẻ thông minh của nó và chọn Configure Data Source để gán nó cho một control nguồn dữ liệu. Bạn cũng có thể sử dụng liên kết Auto Forms trong thẻ thông minh để áp dụng một tập hợp style cho lưới mà nó hiển thị.

Bởi vì DetailsView chỉ hiển thị một record đơn, nên bạn cần thực hiện thêm vài bước để bảo đảm nó hiển thị đúng record. Để thực hiện điều này, bạn cần sử dụng một biểu thức lọc (một biểu thức SQL giới hạn các record mà bạn nhìn thấy theo các tiêu chuẩn do bạn chỉ định). Bạn thêm biểu thức lọc (filter expression) vào nguồn dữ liệu bằng cách xác lập các thuộc tính FilterExpression và FilterParameters của DetailsView.

Chẳng hạn, hãy xem trang được minh họa ở hình 4.7. GridAndDetails.aspx chứa cả GridView hiển thị thông tin chọn về năm record đầu tiên và DetailsView hiển thị tất cả các trường của record được chọn.



Hình 4.7. Nối một GridView với DetailsView

Trang này cần hai nguồn dữ liệu, một cho GridView (được định nghĩa theo cách giống như đã mô tả trong mục "Hiển thị các Table tương tác không cần viết mã") và một cho DetailsView. Dễ dàng nguồn dữ liệu DetailsView như sau:

```
<asp:SqlDataSource ID="SingleCustomerSource" Runat="server"
  SelectCommand="SELECT CustomerID, CompanyName, ContactName, ContactTitle,
Address, City, Country FROM Customers WHERE CustomerID=@CustomerID"
  ConnectionString=
"Data Source=127.0.0.1;Integrated Security=SSPI;Initial Catalog=Northwind"
>

  <SelectParameters>
    <asp:ControlParameter Name="CustomerID" ControlID="GridView1"
      PropertyName="SelectedValue">
    </asp:ControlParameter>
  </SelectParameters>
</asp:SqlDataSource>
```

Mẫu truy vấn SELECT này chọn chỉ một hàng phù hợp với CustomerID được chọn trong control GridView.

Chúng ta có thể dễ dàng nối một DetailsView cơ bản như minh họa ở hình 4.7. Nhưng mọi thứ sẽ trở nên dễ dàng hơn nhiều nếu bạn thực hiện công việc này để thêm các khả năng hiệu chỉnh, xóa và chèn vào DetailsView. Bạn có thể thêm mọi tính năng này bằng thao tác nhấp một nút, miễn là bạn phải bảo đảm nguồn dữ liệu đã nối kết có mọi thông tin cần thiết. Chẳng hạn, nếu bạn muốn tạo một SqlDataSource hỗ trợ việc xóa, bạn cần cấu hình các thuộc tính DeleteCommand và DeleteParameters. Để tạo một nguồn dữ liệu hỗ trợ chèn các record mới, bạn cần thêm một InsertCommand và InsertParameters.

Việc thêm các chi tiết bổ sung này thật sự khá dễ dàng. Tất cả những gì bạn cần làm là hiểu vài quy tắc sau:

- Mọi việc cập nhật được thực hiện thông qua các lệnh đã thông số hóa vốn sử dụng các placeholder đã được đặt tên thay cho các trị.
- Tên thông số giống như tên trường, với một dấu @đứng trước. Chẳng hạn, trường ContactName trở thành thông số @ ContactName.
- Khi bạn viết mệnh đề Where cho mẫu truy vấn, bạn cần đặt trước tên thông số text original. Điều này biểu thị bạn muốn sử dụng trị ban đầu (bỏ qua bất kỳ sự thay đổi mà người dùng có thể đã thực hiện). Chẳng hạn, @CustomerID trở thành @ original_CustomerID.

Nếu bạn tuân theo các quy tắc này, control DetailsView sẽ nối các trị thông số lại tự động. Để thử này, bạn thực hiện theo các bước dưới đây.

Trước tiên, viết một lệnh đã thông số hóa vốn sử dụng các placeholder đã được đặt tên thay cho các trị. Chẳng hạn, sau đây là một DeleteCommand đã được thông số hóa để xóa record đang được chọn, vốn tuân theo các quy tắc ở trên:

```
DELETE Customers WHERE CustomerID=@original_CustomerID
```

Lệnh này xóa record đang được chọn. Điểm hay về lệnh này là bởi vì nó tuân theo các quy tắc đặt tên đã được liệt kê ở trên nên bạn không phải bận tâm về việc cung cấp một trị. Thay vào đó, bạn chỉ định nghĩa thông số như được minh họa dưới đây, và DetailsView sẽ sử dụng CustomerID từ record đang được hiển thị.

```
<asp:SqlDataSource ID="SingleCustomerSource" Runat="server"
 DeleteCommand="DELETE Customers WHERE CustomerID=@original_CustomerID"
 ... >
```

```
<DeleteParameters>
  <asp:Parameter Name="CustomerID">
  </asp:Parameter>
</DeleteParameters>
...
</asp:SqlDataSource>
```

Ví dụ 4.2 minh họa một SqlDataSource đã hoàn chỉ vốn định nghĩa các lệnh cho các thao tác cập nhật, chèn và xóa theo cách này.

Ví dụ 4.2. Một thẻ gán SqlDataSource

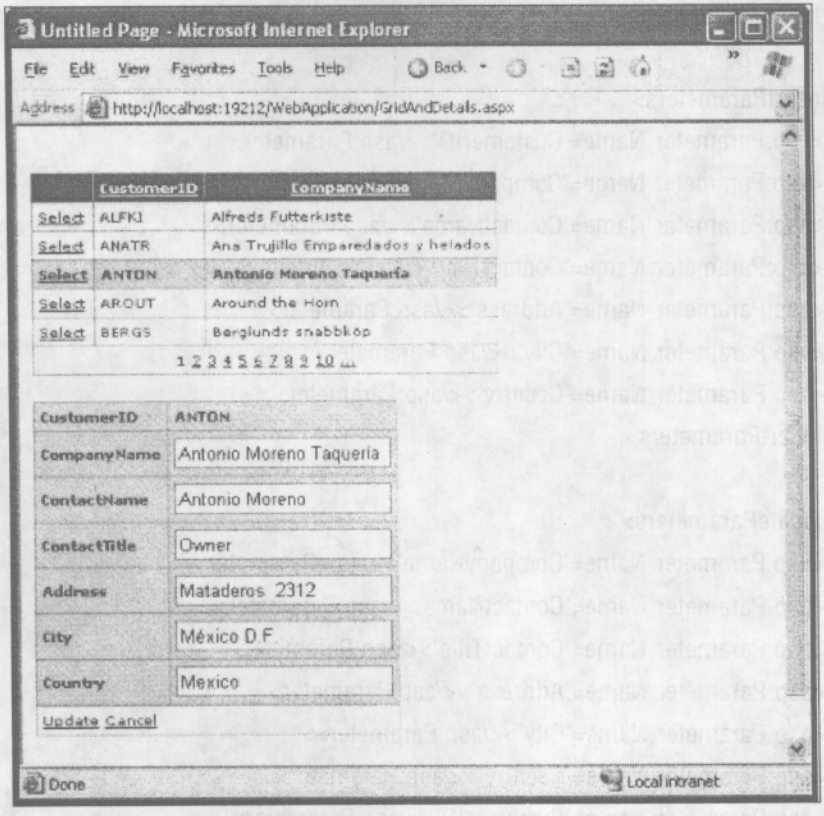
```
<asp:SqlDataSource ID="SingleCustomerSource" Runat="server"
  ConnectionString=
"Data Source=127.0.0.1;Integrated Security=SSPI;Initial Catalog=Northwind"
  SelectCommand=
"SELECT CustomerID,CompanyName,ContactName,ContactTitle,Address,
City,Country FROM Customers"
  FilterExpression="CustomerID='@CustomerID'"
  DeleteCommand="DELETE Customers WHERE CustomerID=@original_CustomerID"
  InsertCommand=
"INSERT INTO Customers (CustomerID,CompanyName,ContactName,ContactTitle,Address,
City,Country) VALUES (@CustomerID,@CompanyName,@ContactName,@ContactTile,
@Address, @City,@Country)"
  UpdateCommand=
   "UPDATE Customers SET CompanyName=@CompanyName,ContactName=
@ContactName,ContactTitle=@ContactTitle,Address=@Address,City=@City,Country=@Country
WHERE
CustomerID=@original_CustomerID">

  <FilterParameters>
    <asp:ControlParameter Name="CustomerID" Type="String" ControlID="GridView1"
     PropertyName="SelectedValue">
    </asp:ControlParameter>
  </FilterParameters>

  <DeleteParameters>
    <asp:Parameter Name="CustomerID">
    </asp:Parameter>
```

```
    </DeleteParameters>

    <InsertParameters>
      <asp:Parameter Name="CustomerID"></asp:Parameter>
      <asp:Parameter Name="CompanyName"></asp:Parameter>
      <asp:Parameter Name="ContactName"></asp:Parameter>
      <asp:Parameter Name="ContactTile"></asp:Parameter>
      <asp:Parameter Name="Address"></asp:Parameter>
      <asp:Parameter Name="City"></asp:Parameter>
      <asp:Parameter Name="Country"></asp:Parameter>
    </InsertParameters>

    <UpdateParameters>
      <asp:Parameter Name="CompanyName"></asp:Parameter>
      <asp:Parameter Name="ContactName"></asp:Parameter>
      <asp:Parameter Name="ContactTitle"></asp:Parameter>
      <asp:Parameter Name="Address"></asp:Parameter>
      <asp:Parameter Name="City"></asp:Parameter>
      <asp:Parameter Name="Country"></asp:Parameter>
      <asp:Parameter Name="CustomerID"></asp:Parameter>
    </UpdateParameters>

    </asp:SqlDataSource>
```

Đây là một thẻ gán dài nhưng các định nghĩa thông số thì khá đơn giản. Thậm chí tốt hơn, các wizard Visual Studio có thể giúp bạn tạo các lệnh chèn, cập nhật và xóa một cách nhanh chóng. Chỉ cần chọn dấu ellipsis kế bên tên thuộc tính trong cửa sổ Properties (ví dụ, thuộc tính DeleteCommand) rồi gõ lệnh đã được thông số hóa và nhấp Refresh Parameters. Việc làm mới sẽ tự động tạo các thẻ gán thông số dựa vào lệnh của bạn.

Để cấu hình DetailsView sao cho nó sử dụng các lệnh này, bạn chỉ cần nhấp thẻ thông minh và thêm một dấu kiểm kế bên các tùy chọn Enable Inserting, Enable Deleting và Enable Updating. Điều này sẽ xác lập các thuộc tính Boolean như AutoGenerateInsertButtong, AutoGenerateDeleteButton và AutoGenerateEditButton.

Hình 4.8 minh họa một DetailsView ở chế độ hiệu chỉnh.



Hình 4.8. Hiệu chỉnh một record với DetailsView

4.6.2. Thế còn...

... việc cập nhật GridView sao cho nó vẫn đồng bộ hóa với DetailsView thì sao? Nếu bạn không thực hiện thêm bất kỳ bước nào, bạn sẽ nhận thấy một vài chỗ không nhất quán; các thay đổi do bạn hiệu chỉnh, chèn hay xóa các record với DetailsView sẽ không xuất hiện trong GridView chừng nào bạn chưa làm mới trang bằng tay. Để khắc phục vấn đề này, bạn cần thêm một ít mã xử lý event. Trong trường hợp này, các event DetailsView quan trọng là ItemInserted, ItemDeleted và ItemUpdated, vốn kích khởi sau khi mỗi thao tác hiệu chỉnh này đã hoàn tất. Sau đây là mã mà bạn có thể thêm vào mỗi event handler để làm mới lưới khi một mục được chèn, xóa hay cập nhật:

```
Sub DetailsView1_ItemUpdated(ByVal sender As Object, _
  ByVal e As System.Web.UI.WebControls.DetailsViewUpdatedEventArgs)
    GridView1.DataBind( )
End Sub
```

DetailsView có nhiều tính năng hơn bạn tưởng. Chẳng hạn, bạn có thể xử lý các event ItemInserting, ItemDeleting và ItemUpdating để kiểm tra sự thay đổi theo yêu cầu, thực hiện sự hợp thức dữ liệu và ngăn chặn cập nhật. Bạn cũng có thể tạo các control hiệu chỉnh của riêng mình nhờ dùng các template (khuôn mẫu).

4.7. Tạo diện mạo nhất quán với Master Pages

Đa số các web site chuyên nghiệp chuẩn hóa kiểu trình bày (layout) của mình. Trên web site O'Reilly (http://www.oreilly.com), chẳng hạn, một thanh định hướng luôn nằm ở phía bên trái của một trang nội dung, và một lôgô công ty được hiển thị ở đỉnh. Các chi tiết này vẫn nhất quán khi người dùng di chuyển từ trang này sang trang kia.

Ghi chú

Bạn cần tuân theo một kiểu thiết kế nhất định qua tất cả các trang trong một web site? ASP.NET 2.0 có một tính năng master pages mới cho phép bạn tạo các template cho trang.

Trong ASP.NET 1.0 và 1.1, bạn có thể tạo các web site với các kiểu trình bày chuẩn hóa nhưng không có bất kỳ công cụ nào để giúp bạn thực hiện điều này một cách dễ dàng. Chẳng hạn, với các control user, bạn có thể sử dụng lại các khối dưới dạng người dùng nhưng không có bất kỳ cách nào để bảo đảm chúng luôn có ở cùng một vị trí trên các trang khác nhau. Nhờ dùng các frame HTML, bạn có thể phân chia một cửa sổ trình duyệt web sao cho nó hiển thị nhiều trang web, nhưng rất khó giữ cho tất cả các trang ở một tọa độ thích hợp. Trong ASP.NET 2.0, các giải pháp không hoàn hảo này đã được thay thế bằng một tính năng mới được gọi là master pages, một hệ thống tạo khuôn mẫu cho trang.

4.7.1 Bạn làm điều đó như thế nào?

Để tạo một trang master (chính) cơ bản trong Visual Studio, bạn chọn Website > Add New Item từ menu, chọn Master Page và nhấp OK để thêm mục.

Các trang master tương tự như các trang ASP.NET bình thường ở chỗ chúng có thể chứa HTML, các web control và mã. Tuy nhiên, chúng có một phần mở rộng khác (.master thay vì .aspx), và chúng không được yêu cầu trực tiếp bởi một trình duyệt. Thay vào đó, các trang khác (được gọi là các trang nội dung) có thể sử dụng trang master.

Bạn thiết kế trang master như bạn thường làm với một trang web ASP.NET bình thường, thêm text và các control bạn cần để đạt được một diện mạo nhất quán qua tất cả các trang của site. Các thành phần

bạn thêm vào trang master không thể được chỉnh sửa bởi các trang nội dung đang sử dụng nó. Bạn sử dụng control ContentPlaceHolder mới để đánh dấu các vùng dành riêng cho nội dung vốn sẽ thay đổi từ trang này sang trang kia. Trong các vùng này của trang master, trang nội dung có thể thêm các control và HTML riêng của chúng.

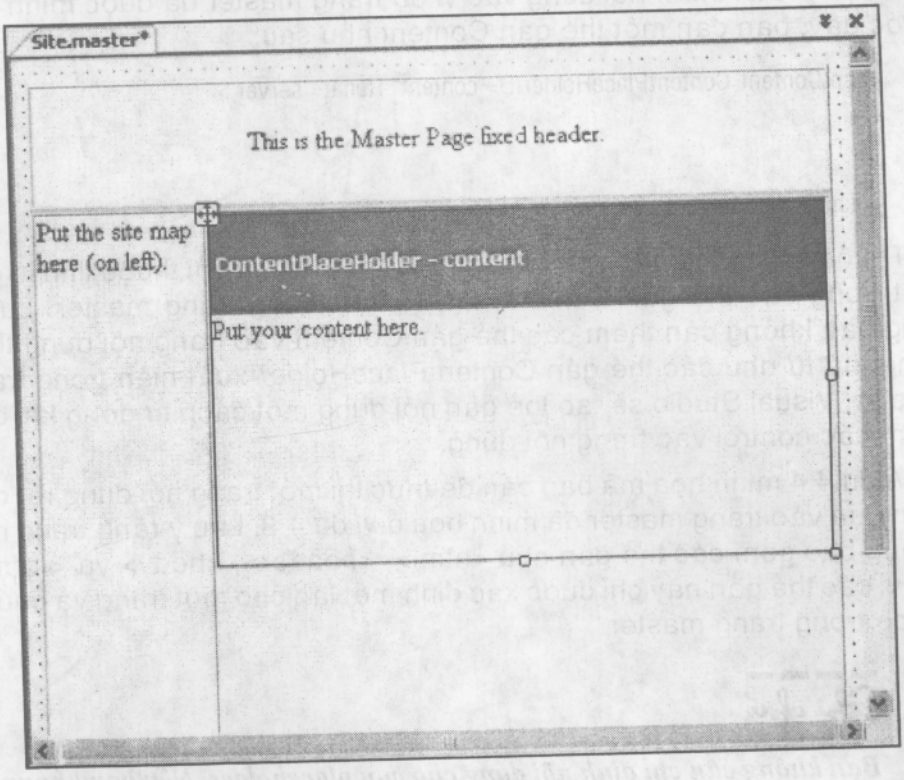
Hãy xem xét trang master mẫu mà nguồn của nó được minh họa ở ví dụ 4.3. Nó tạo hai table. Table trên cùng chứa vùng header và table thứ hai chứa phần còn lại của trang. Table thứ hai được chia ra thành hai ô, một ô ở bên trái cho thanh định hướng và một ô ở bên phải để chứa một thẻ gán ContentPlaceHolder. Bất kỳ trang nội dung sử dụng (nghĩa là thừa kế từ) trang master này có thể hoàn toàn điều khiển nội dung của ô đó, nhưng không thể điều khiển bất kỳ nội dung của ô khác trong table đó hay các table khác trên trang master.

Ví dụ 4.3. Một trang master sử dụng một table

```
<%@ Master language="VB" %>

<html>
  <head id="Head1" runat="server">
    <title>Master Page</title>
  </head>

  <body>
    <form id="Form1" runat="server">
      <table id="header" width="100%" height="80px"
       cellspacing="1" cellpadding="1" border="1">
        <tr>
          <td width="100%" style="TEXT-ALIGN: center">
            This is the Master Page fixed header.
          </td>
        </tr>
      </table>

      <table id="main" width="100%" height="100%"
       cellspacing="1" cellpadding="1" border="1">
        <tr>
          <td valign=top width="100px">
            Put the site map here (on left). </td>
```

```
        <td valign=top >
          <asp:ContentPlaceHolder id="content" runat="Server">
            Put your content here.
          </asp:ContentPlaceHolder>
        </td>
      </tr>
    </table>

  </form>
</body>

</html>
```

Ví dụ 4.9 minh họa trang master vào lúc thiết kế. Đối với kiểu trình bày cao cấp hơn, bạn có thể sử dụng các table xếp lồng hay đặt thẻ gán ContentPlaceHolder bên trong một ô đơn của một table phức tạp hơn, vốn bao gồm nhiều hàng và nhiều cột.



Hình 4.9. Một trang master đơn giản

Để tạo một trang nội dung mới, hãy nhấp phải Solution Explorer và chọn Add New Item. Chọn tùy chọn Web Form, đặt tên cho file và sau đó chọn hộp kiểm "Select master page". Khi bạn nhấp Add, một hộp thoại sẽ xuất hiện, nhắc bạn chọn một trong các trang master trong ứng dụng web hiện hành. Chọn trang master ở ví dụ 4.3 và nhấp OK.

Khi bạn tạo một trang nội dung, nó tự động có cùng diện mạo với trang master mà nó phát sinh từ đó. Bạn có thể thêm nội dung chỉ bên trong vùng nội dung được chỉ định bởi một control ContentPlaceHolder. Các vùng header và sitemap đã ấn định sẵn của trang master sẽ ở dạng tô xám trong Visual Studio.

Markup (sự đánh dấu) thật sự cho các trang nội dung có vẻ hơi khác với các trang bình thường. Trước hết, chỉ thị Page liên kết với trang master mà bạn đang sử dụng, như minh họa dưới đây:

```
<%@ Page MasterPageFile="Site.master" %>
```

Để thêm nội dung vào trang, bạn cần nhập nó vào bên trong một thẻ gán Content đặc biệt. Thẻ gán Content liên kết với một trong các thẻ gán ContentPlaceHolder mà bạn đã tạo trong trang master. Chẳng hạn, nếu bạn muốn thêm nội dung vào ví dụ trang master đã được minh họa trước đây, bạn cạn một thẻ gán Content như sau:

```
<asp:Content ContentPlaceHolderID="content" Runat="server">

...

</asp:Content>
```

Thuộc tính ContentPlaceHolderID này phải trùng với thuộc tính id của một trong các thẻ gán ContentPlaceHolder trong trang master. Lưu ý rằng bạn không cần thêm các thẻ gán Content vào trang nội dung theo cùng thứ tự như các thẻ gán ContentPlaceHolder xuất hiện trong trang master. Visual Studio sẽ tạo thẻ gán nội dung một cách tự động khi bạn thêm các control vào trang nội dung.

Ví dụ 4.4 minh họa mã bạn cần để thực thi một trang nội dung rất đơn giản dựa vào trang master đã minh họa ở ví dụ 4.3. Lưu ý rằng trang này không bao gồm các thẻ gán như <html>, <header>, <body> và <form>, bởi vì các thẻ gán này chỉ được xác định một lần cho một trang và chúng đã có trong trang master.

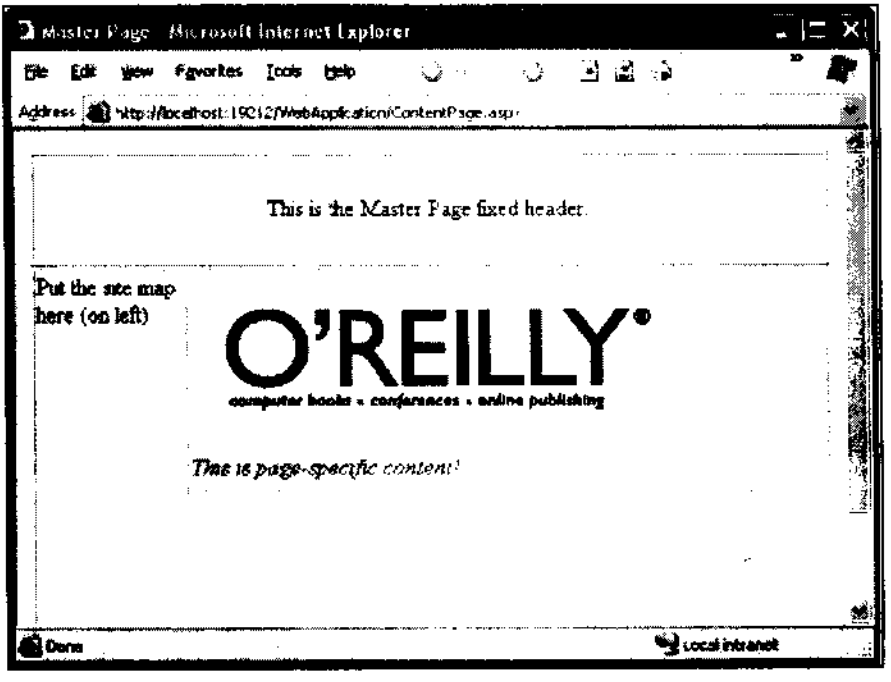
Ghi chú

Bạn không cần chỉ định nội dung cho mỗi placeholder. Nếu bạn không chỉ định, ASP.NET sẽ hiển thị bất kỳ nội dung nào trong thẻ gán ContentPlaceHolder trên trang master (nếu có).

Ví dụ 4.4. Một trang nội dung với một hình ảnh và text

```
<%@ page language="VB" MasterPageFile="Site.master" %>

<asp:Content ContentPlaceHolderID=content Runat=server>
    <asp:Image ID="image1" ImageUrl="oreilly_header.gif" Runat="server" />
    <br />
    <br />
    <i>This is page-specific content!</i>
    <hr />
</asp:Content>
```

Hình 4.10 minh họa trang nội dung được tạo.



Hình 4.10. Một trang nội dung đơn giản

Bạn có thể tạo các trang master sử dụng các trang master khác đã được xác định trước đây, xếp lồng trang master này bên trong trang master khác một cách hiệu quả. Kiểu thiết kế xếp lồng như vậy có thể hữu ích nếu bạn cần xác định một số nội dung xuất hiện trên mọi trang trong một web site (chẳng hạn như phần header của công ty) và một số nội dung xuất hiện trên nhiều trang nhưng không phải tất cả các trang (như thanh định hướng).

Một lý do thích hợp để sử dụng các trang master là dành riêng một phần trang web cho một loại control định hướng. Mục kế tiếp "Thêm sự định hướng vào Site của bạn" sẽ trình bày chủ điểm này một cách chi tiết hơn.

4.7.2. Các cách khác để bảo đảm sự nhất quán

ASP.NET 2.0 giới thiệu một tính năng khác để chuẩn hóa các web site được gọi là tạo control theme. Trong khi các trang master bảo đảm một kiểu trình bày thông thường và cho phép bạn lặp lại các thành phần nhất định qua toàn bộ một site thì việc tạo theme sẽ giúp bảo đảm các control trang web có "diện mạo" nhất quán. Về cơ bản, một control theme là một tập hợp các thuộc tính style (chẳng hạn như các font và màu) có thể được áp dụng cho các control khác.

4.8. Thêm sự định hướng vào Site của bạn

Đa số các web site có một loại thanh định hướng cho phép người dùng di chuyển từ trang này sang trang kia. Trong ASP.NET 1.0 và 1.1, bạn có thể dễ dàng tạo các control định hướng này nhưng bạn cần làm điều đó bằng tay. Trong ASP.NET 2.0, một tính năng sitemap mới cung cấp một giải pháp cài sẵn tiện lợi hơn rất nhiều. Nguyên tắc cơ bản là bạn định rõ cấu trúc của website trong một file XML đặc biệt. Ngay khi bạn thực hiện bước đó, bạn có thể cấu hình một control danh sách hay hình cây để sử dụng dữ liệu sitemap, vốn cho bạn một control định hướng mà bạn có thể nhấp vào mà không cần viết mã.

Ghi chú

ASP.NET 2.0 cung cấp các tính năng định hướng mới nhằm cho phép bạn tạo một sitemap và kết buộc nó với các control khác.

4.8.1 Bạn làm điều đó như thế nào?

Bước đầu tiên trong việc sử dụng tính năng sitemap mới của ASP.NET là định rõ cấu trúc của web site trong một file XML được gọi là web.sitemap. Để thêm file này vào site của bạn trong Visual Studio, hay nhấp phải vào Solution Explorer và chọn Add New Item. Chọn kiểu file Site Map và nhấp Add.

Thành phần đầu tiên bạn cần trong file web.sitemap là thẻ gán sitemap gốc:

```
<siteMap>

</siteMap>
```

Trong thẻ gán siteMap, bạn thêm một thành phần con <siteMapNode> cho mỗi mục bạn muốn hiển thị trong sitemap. Sau đó, bạn có thể đặt tựa, cung cấp phần mô tả chi tiết và liên kết URL cho mỗi mục nhập nhờ dùng các thuộc tính. Sau đây là một ví dụ:

```
<SiteMapNode title="Home" description="Home Page" url="default.aspx" />
```

Lưu ý rằng thẻ gán này kết thúc với các ký tự /> thay vì chỉ >. Điều này biểu thị nó là một phần tử rỗng, nói cách khác nó không chứa bất kỳ phần tử khác. Tuy nhiên, nếu bạn muốn tạo một sitemap nhiều cách, bạn phải xếp lồng phần tử bên trong một thành phần khác. Sau đây là một ví dụ:

```
<siteMapNode title="Home" description="Home Page" url="default.aspx" >

   <siteMapNode title="Products"

    description="Order Products" url="produ.aspx" />

</siteMapNode>
```

Ví dụ 4.5 minh họa một sitemap với sáu liên kết ở ba cấp độ.

Ví dụ 4.5. Một sitemap đa cấp

```
<?xml version="1.0" ?>
<siteMap>
  <siteMapNode title="Home" description="Home" url="default.aspx">
    <siteMapNode title="Personal" description="Personal Services"
     url="personal.aspx">
      <siteMapNode title="Resume" description="Download Resume"
       url="resume.aspx" />
    </siteMapNode>

    <siteMapNode title="Business" description="Business Services"
     url="business.aspx">
      <siteMapNode title="Products" description="Order Products"
       url="products.aspx" />
      <siteMapNode title="Contact Us" description="Contact Information"
       url="contact.aspx" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

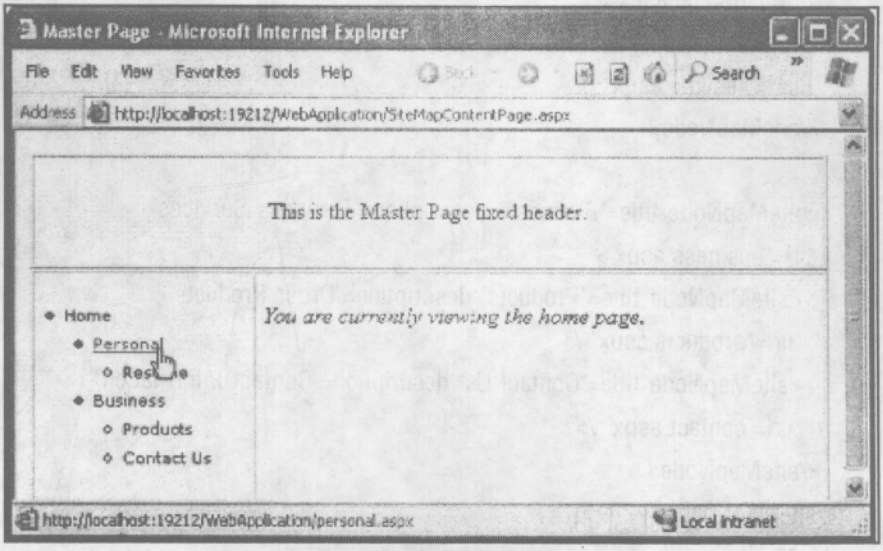
Ngay sau khi bạn tạo một sitemap, bạn có thể dễ dàng sử dụng nó trên một trang web, nhờ vào control SiteMapDataSource mới. Control này hoạt động gần giống như các control nguồn dữ liệu khác đã được trình bày trong mục "Kết buộc với dữ liệu không cần viết mã". Tuy nhiên, nó không cần bất kỳ thuộc tính nào. Ngay khi bạn thêm SiteMapDataSource, ASP.NET tự động đọc file web.sitemap và làm cho dữ liệu của nó có sẵn cho các control khác của bạn.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" Runat="server" />
```

Bây giờ bạn có thể kết buộc hầu như bất kỳ control khác với SiteMapDataSource. Bởi vì theo mặc định, các sitemap được xếp theo hệ thống phân cấp, chúng hoạt động rất tốt với control treeView mới. Sau đây là một control treeView kết buộc với dữ liệu sitemap:

```
<asp:TreeView ID="TreeView1" Runat="server" DataSourceID="SiteMapDataSource1"
  Font-Names="Verdana" Font-Size="8pt" ForeColor="Black" ImageSet="BulletedList"
  Width="149px" Height="132px">
</asp:TreeView>
```

treeView được tạo không chỉ hiển thị sitemap mà còn mô phỏng mỗi nút ở dạng một hyperlink mà nếu được nhấp vào, nó sẽ đưa người dùng đến trang thích hợp. Hình 4.11 minh họa một trang nội dung dựa vào một trang master vốn sử dụng một treeView với một sitemap. (Xem mục "Tạo diện mạo nhất quán với Master Pages" để tìm hiểu thêm thông tin về các trang master).



Hình 4.11. Sử dụng một sitemap trong một trang master

4.8.2. Tạo tùy biến sitemap

Có nhiều thứ hơn bạn có thể làm để điều khiển diện mạo của một site cũng như cách hoạt động của nó. Sau đây là một số điểm khởi đầu:

Hiển thị một sitemap trong một control không phân cấp

Các control như ListBox và GridView không hỗ trợ khung xem dựa vào hình cây của sitemap. Để giải quyết vấn đề này, hãy xác lập thuộc tính SiteMapDataSource.SiteMapViewType sang Flat thay vì tree để sitemap nhiều lớp này được ép phẳng thành một danh sách một cấp. Bạn cũng có thể sử dụng tùy chọn Flat với một TReeView để tiết kiệm chỗ trên màn hình (bởi vì các cấp kế tiếp sẽ không được thụt vào).

Thay đổi sitemap được hiển thị trong các trang khác nhau

Để đạt được điều này, hãy đặt mọi thông tin sitemap bạn cần vào cùng một file web.sitemap, nhưng ở các nhánh khác nhau. Sau đó, xác lập SiteMapDataSource.StartingNodeUrl sang URL của trang mà bạn muốn sử dụng làm gốc của sitemap. SiteMapDataSource sẽ chỉ lấy thông tin từ nút đó, và tất cả các nút mà nó chứa.

Làm cho sitemap có thể bẻ gập lại

Nếu bạn có một sitemap lớn, chỉ cần xác lập TReeView.ShowExpand Collapse sang TRue, và các hộp cộng quen thuộc sẽ xuất hiện kế bên Home, Personal, và Business, cho phép bạn hiển thị chỉ một phần của sitemap vào mỗi lần.

Tinh chỉnh diện mạo của TreeView

Điều này thật sự dễ thực hiện. Trong ví dụ trước, treeView đã sử dụng kiểu bullet, vốn hiển thị các biểu tượng bullet khác nhau kế bên mỗi mục. Bằng cách xác lập treeView.ImageSet sang các giá trị khác nhau có sẵn bên trong phép liệt kê treeViewImageSet, bạn có thể hiển thị các dấu bullet hình vuông, các mũi tên, các biểu tượng folder và file, và nhiều thứ khác. Để tìm hiểu thêm thông tin về việc điều chỉnh treeView (hay sử dụng nó trong các tình huống khác không liên quan đến sitemap), hãy tham khảo lớp System.Web.UI WebControls.TreeView.

Truy xuất thông tin sitemap từ một vị trí khác

Có lẽ bạn muốn chứa sitemap của mình trong một file khác, một cơ sở dữ liệu hay một nguồn dữ liệu khác. Thật không may, SiteMapProvider không có khả năng truy xuất thông tin từ những vị trí này. Tuy nhiên, bạn sẽ cần tạo sitemap provider tùy ý của riêng mình.

4.9. Xác thực các User một cách dễ dàng

Trong ASP.NET 1.0 và 1.1, các nhà phát triển có một công cụ tiện lợi được gọi là forms authentication (xác thực các form). Về cơ bản, form authentication theo dõi những user (người dùng) nào đã đăng nhập vào bằng cách sử dụng một cookie đặc biệt. Nếu ASP.NET nhận thấy một user chưa đăng nhập mà lại truy cập vào một trang được bảo vệ, nó tự động hướng user đó sang trang login tùy ý của bạn.

Ghi chú

Bạn đã chán viết mã xác thực của riêng bạn? ASP.NET 2.0 sẽ chú ý điều này, bảo lưu thông tin user trong một cơ sở dữ liệu và hợp thức một user login khi bạn yêu cầu nó.

Tự bản thân forms authentication hoạt động rất tốt, nhưng nó vẫn cần sự nỗ lực từ phía bạn, chẳng hạn, bạn cần tạo trang login và viết mã để xem xét tên login của user và password rồi so sánh nó với các trị trong một cơ sở dữ liệu tùy ý. Tùy thuộc vào cách hoạt động của ứng dụng, bạn cũng có thể cần viết mã để thêm các user mới và loại bỏ các user cũ. Mã này không phức tạp lắm nhưng nó có thể gây nhàm chán.

ASP.NET 2.0 giúp giảm bớt những việc mà bạn phải thực hiện với các tính năng membership mới của nó. Về cơ bản, tất cả những gì bạn cần làm là sử dụng các phương thức của các lớp membership để tạo, xóa và hợp thức thông tin người dùng. ASP.NET tự động bảo lưu một cơ sở dữ liệu chứa thông tin user thay cho bạn trong hậu cảnh.

4.9.1 Bạn làm điều đó như thế nào?

Bước đầu tiên khi thêm sự xác thực (autehntication) của bạn là chọn một membership provider, vốn xác định nơi mà thông tin user sẽ được chứa. ASP.NET bao gồm các membership provider cho phép bạn nối kết với một cơ sở dữ liệu SQL Server và một vài provider bổ sung đã được hoạch định.

Membership provider được chỉ định trong file cấu hình web.config. Tuy nhiên, thay vì gõ thông tin này bằng tay, bạn hầu như chắc chắn sử dụng WAT, đã được trình bày trong mục "Quản lý một ứng dụng Web". Chỉ cần nhấp tab Security và nhấp liên kết "Use the security Setup Wizard" để di chuyển qua một wizard vốn cung cấp cho bạn mọi tùy chọn bạn cần. Câu hỏi đầu tiên là phương thức truy cập, nói cách khác khách tham quan của bạn sẽ tự xác thực mình như thế nào? Chọn "From the internet" để sử dụng forms authentication thay vì Windows authentication.

Bước sau đây cho phép bạn chọn membership provider. Bạn có thể chọn một provider để sử dụng cho mọi tính năng ASP.NET, hay các provider riêng biệt cho các tính năng khác nhau (chẳng hạn như membership, sự bảo mật theo vai trò,...). Nếu bạn chọn sử dụng một cơ sở dữ liệu SQL Server, bạn cũng phải chạy tiện ích aspnet_regsql.exe, vốn đưa bạn qua các bước cần thiết để cài đặt cơ sở dữ liệu membership. Bạn sẽ tìm thấy tiện ích aspnet_regsql.exe trong thư mục c:\[WinDir]\Microsoft.NET > ramework\[Version].

Kế tiếp, bạn sẽ có cơ hội để tạo các user và role (vai trò). Các role sẽ được trình bày trong một mục sau ("Sử dụng sự xác thực theo vai trò"), vì vậy bạn chưa cần tạo chúng vào lúc này. Bạn không cần tạo thử một user bởi vì bạn sẽ làm điều đó thông qua web site của mình trong bước kế tiếp.

Bạn có thể chọn tên của trang login bằng cách chỉnh sửa một <authentication> trong file web.config, như sau:

```
<?xml version="1.0"?>

<configuration>
  <system.web>
    <authentication mode="Forms">
      <forms loginUrl="Login.aspx" />
    </authentication>

    <!— Other settings ommitted. —>
  </system.web>
</configuration>
```

Trong trường hợp này, trang login cho ứng dụng được gọi là Login.aspx (mặc định). Trong trang này, bạn có thể sử dụng các phương thức và thuộc tính chia sẻ của lớp System.Web.Security.Membership để xác thực các user của bạn. Tuy nhiên, bạn không cần làm như vậy, bởi vì ASP.NET có một tập hợp control liên quan đến sự bảo mật mà bạn có thể rê và thả vào các trang web của mình. Các control bảo mật bao gồm:

Login

Hiển thị các control cần thiết cho một user đăng nhập, bao gồm các hộp text username và password và một nút Login. Một cách tùy ý, bạn có thể hiển thị một hộp text địa chỉ email, và bạn có thể cấu hình toàn bộ nhãn text trong control bằng cách chỉnh sửa các thuộc tính như UserNameLabelText và PasswordLabelText.

LoginView

Hiển thị một template từ một nhóm các template, phụ thuộc vào người nào đang được đăng nhập. Điều này cho bạn một cách để tùy biến nội dung cho các user và các role khác nhau mà không cần sử dụng bất kỳ mã tùy ý nào.

PasswordRecovery

Cung cấp một cơ chế mà thông qua đó các user có thể có một password đã bị quên lãng được gởi đến cho họ. Tính năng này được tắt theo mặc định và cần phải điều chỉnh một vài xác lập web.config.

LoginStatus

Hiển thị một liên kết Login nếu user không đang được đăng nhập hay một liên kết Logout nếu user đã đăng nhập.

LoginName

Hiển thị tên của user đang được đăng nhập trong một nhãn.

CreateUserWizard

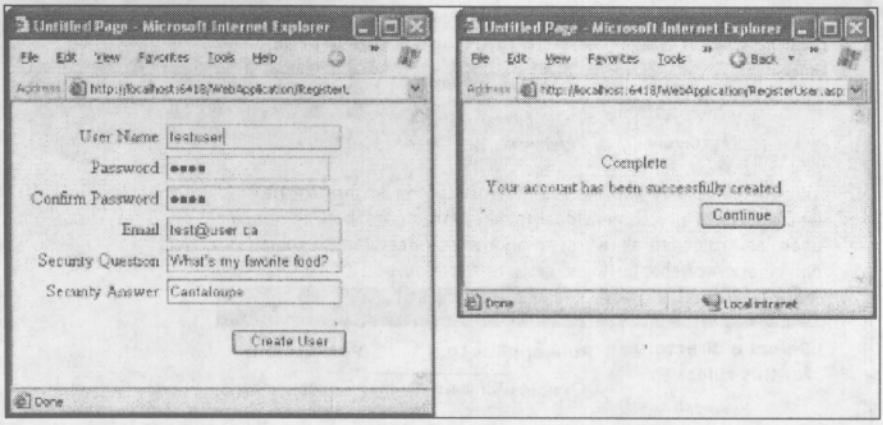
Cho phép user bước qua tiến trình tạo một user account mới.

ChangePassword

Cho phép user thay đổi password hiện tại của mình bằng cách chỉ định các password hiện tại và mới.

Bạn sẽ tìm thấy các control bảo mật trong tab Login của hộp công cụ Visual Studio. Để thử chúng, hãy tạo một trang mới được gọi là RegisterUser.aspx. Thả một control CreateUserWizard lên trang web. Bây giờ hãy chạy trang và sử dụng wizard để tạo một user mới với username testuser và password test.

Theo mặc định, control CreateUserWizard sử dụng hai bước (được minh họa ở hình 4.12). Bước đầu tiên cho phép bạn chỉ định toàn bộ thông tin user và bước hai chỉ hiển thị một thông báo xác nhận.



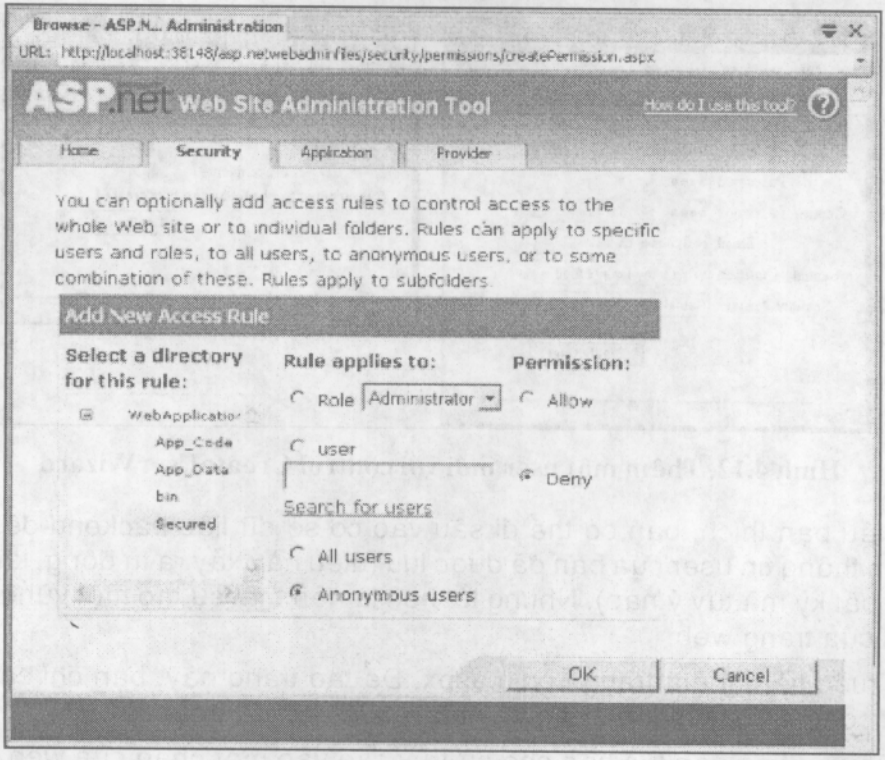
Hình 4.12. Thêm một user mới với control CreateUserWizard

Nếu bạn thích, bạn có thể đi sâu vào cơ sở dữ liệu backend để xác nhận thông tin user của bạn đã được lưu (điều này xảy ra tự động, không cần bất kỳ mã tùy ý nào). Nhưng tốt hơn là nên thật sự tạo một vùng giới hạn của trang web.

Trước tiên, thêm trang Login.aspx. Để tạo trang này, bạn chỉ cần rê một control Login sang trang và như vậy là bạn đã hoàn tất.

Bây giờ, đã đến lúc hạn chế sự truy cập vào một phần của web site. Chọn Website > New Folder để tạo một thư mục con trong thư mục ứng dụng web và đặt tên cho thư mục mới là Secured. Kế tiếp, tạo một trang mới trong thư mục này được gọi là Private.aspx và thêm text "This is a secured page".

Chạy WAT bằng cách chọn Website > ASP.NET Configuration. Chọn tab Security. Bằng cách sử dụng tab này, bạn có thể xem xét danh sách các user (bao gồm text user mà bạn đã thêm vào ở bước trước) và chỉnh sửa thông tin của họ. Tuy nhiên, điều bạn thật sự cần làm là nhấp liên kết "Create access rules" để hạn chế sự truy cập vào thư mục Secured. Chọn thư mục trong danh sách, chọn tùy chọn Deny Permission và chọn Anonymous users, như minh họa ở hình 4.13. Sau đó, nhấp OK để thêm quy tắc này.



Hình 4.13. Tạo một quy tắc để ngăn chặn sự truy cập vô danh vào thư mục Secured

Bây giờ bạn đã sẵn sàng để kiểm tra ví dụ bảo mật đơn giản này. Nhấp phải file Private.aspx và chọn "Set As Start Page". Sau đó, chạy ứng dụng của bạn. ASP.NET sẽ lập tức phát hiện yêu cầu của bạn là cho một trang đã được bảo vệ và bạn chưa xác thực mình. Bởi vì bạn đã cải tiến forms authentication nên nó hướng bạn sang trang Login.aspx.

Nhập Username testuser và password test trong control login. ASP.NET sẽ hợp thức bạn nhờ dùng membership provider và hướng bạn sang trang Private.aspx đã được yêu cầu vào lúc đầu.

Nói cách khác, bằng cách sử dụng các control CreateUserWizard và Login kết hợp với WAT, bạn đã tạo một hệ thống xác thực giúp hạn chế sự truy cập vào một phần nhất định của web site, hoàn toàn không cần bất kỳ dòng mã nào.

4.9.2. Tùy biến tiến trình xác thực

Nếu bạn cần điều khiển cách diễn ra của các tác vụ xác thực, tạo user và các tác vụ bảo mật khác, bạn sẽ hài lòng khi tìm thấy các control bảo mật (security) có thể được mở rộng một cách dễ dàng. Bạn có thể

thêm các bước mới vào CreateUserWizard để thu thập thêm dữ liệu, phản hồi các event vốn kích khởi khi người dùng được đăng nhập (hay từ chối truy cập), thậm chí bạn có thể chuyển đổi các bước sang các thảo luận có thể hiệu chỉnh được để bạn có thể tinh chỉnh giao diện người dùng, thêm các control mới hay loại bỏ các control hiện có.

Nếu bạn muốn đi xa hơn, bạn có thể hủy các control bảo mật nhưng vẫn tạo các giải pháp hầu như không cần mã nhờ dùng các phương pháp tĩnh của lớp System.Web.Security.Membership. Sau đây là một số phương thức bạn có thể gọi:

CreateUser( )

Tạo một record user mới trong cơ sở dữ liệu với một username, một password và một địa chỉ email (tùy ý).

DeleteUser( )

Loại bỏ record user ra khỏi cơ sở dữ liệu vốn có username đã chỉ định.

GeneratePassword( )

Tạo một password ngẫu nhiên với chiều dài đã định. Bạn có thể đưa ra đề nghị này cho người dùng ở dạng một password mặc định khi tạo một record user mới.

GetUser( )

GetUser ( ) truy xuất một record MembershipUser cho một user với username đã cho. Bạn có thể xem xét các thuộc tính MemberShipUser để tìm ra những chi tiết như địa chỉ email của user, account đã được tạo vào lúc nào, user đã đăng nhập lần cuối cùng vào lúc nào,... Nếu bạn không chỉ định một username, phương thức GetUser ( ) sẽ truy xuất user hiện hành cho trang.

GetUserNameByEmail( )

Nếu bạn biết địa chỉ email của một user nhưng không biết user name, bạn có thể sử dụng phương thức này để có được nó.

UpdateUser( )

Sau khi bạn đã truy xuất một đối tượng MembershipUser, bạn có thể chỉnh sửa các thuộc tính của nó rồi gởi đối tượng đến phương thức UpdateUser ( ), vốn chuyển mọi thay đổi của bạn sang cơ sở dữ liệu user.

Validate User( )

Phương thức này chấp nhận một username và password, xác nhận nó trùng với thông tin trong cơ sở dữ liệu (trong trường hợp này nó cho ra true). ASP.NET không thật sự chứa password chưa được mã hóa trong cơ sở dữ liệu, thay vào đó nó sử dụng thuộc tính hashing để bảo vệ thông tin này.

Bằng cách sử dụng các phương thức trên, bạn có thể nhanh chóng tạo các trang login cơ bản và các trang đăng ký user mà không cần viết bất kỳ mã cơ sở dữ liệu nào. Tất cả những gì bạn cần làm là tạo giao diện người dùng cho trang (nói cách khác, thêm các nhãn, các hộp text, và các control khác)

Chẳng hạn, để thiết kế một trang login tùy biến, bạn chỉ cần tạo một trang có hai hộp text (txtUser và txtPassword) và một nút (cmdLogin). Khi nút này được nhấp, hãy chạy mã sau:

```
Sub cmdLogin_Click(ByVal sender As Object, ByVal e As System.EventArgs)


    If Membership.ValidateUser(txtUser.Text, txtPassword.Text) Then
        ' ASP.NET validated the username and password.
        ' Send the user to page that was originally requested.
        FormsAuthentication.RedirectFromLoginPage(txtUser.Text, False)
    Else
        ' The user's information is incorrect.
        ' Do nothing (or just display an error message).
    End If


End Sub
```

Lưu ý mã cho trang này đơn giản như thế nào. Thay vì hợp thức user bằng tay bằng cách nối kết với một cơ sở dữ liệu, đọc một record và kiểm tra các trường, mã này chỉ gọi phương thức Membership.ValidateUser ( ), và ASP.NET thực hiện phần còn lại.

Cũng dễ dàng như vậy, bạn có thể tạo một trang để tạo một record mới với lớp Membership:

```
Sub cmdRegister_Click(ByVal sender As Object, ByVal e As System.EventArgs)


    Dim Status As MembershipCreateStatus
    Dim NewUser As MembershipUser = Membership.CreateUser(_
      txtUser.Text, txtPassword.Text, txtEmail.Text, Status)
```

```
    ' If the user was created successfully, redirect to the login page.
    If Status = MembershipCreateStatus.Success Then
        Response.Redirect("Login.aspx")
    Else
        ' Display an error message in a label.
        lblStatus.Text = "Attempt to create user failed with error " & _
          Status.ToString( )
    End If


End Sub
```

Để tìm hiểu thêm, bạn có thể tham khảo vào mục kế tiếp, các mục này trình bày cách tạo membership framework cơ bản với các tính năng mới:

"Xác định bao nhiêu người đang sử dụng Web Site của bạn"

Giải thích cách các tính năng membership bổ sung có thể theo dõi những người nào đang online (trực tuyến).

"Sử dụng sự xác thực theo vai trò"

Trình bày cách bạn có thể cải tiến logic tạo của mình bằng cách gán các user cho các vai trò (role) nhất định, về cơ bản là cho họ những đặc quyền khác. Tính năng này không được xử lý bởi dịch vụ membership mà bởi một dịch vụ role manager (quản lý vai trò).

"Chứa thông tin cá nhân hóa"

Trình bày cách bạn có thể chứa các loại thông tin khác về user trong một cơ sở dữ liệu, thay vì chỉ username, password và các địa chỉ email. Tính năng này không được xử lý bởi dịch vụ membership mà bởi một dịch vụ personalization (cá nhân hóa) bổ sung.

4.10. Xác định bao nhiêu người đang sử dụng Web Site của bạn

Web sử dụng HTTP, một giao thức không trạng thái ít khi bảo trì một nối kết lâu hơn vài giây. Do vậy, ngay cả khi các user đang đọc qua các trang web của bạn, họ cũng không được kết nối trực tiếp với server của bạn. Tuy nhiên, ASP.NET cho bạn một cách để ước tính có bao nhiêu người đang sử dụng web site của bạn vào bất kỳ thời điểm cho trước nhờ dùng timestamp (dấu thời gian). Thông tin này làm tăng thêm sự tiện lợi cho một site cộng đồng (ví dụ, một diễn đàn thảo luận web), và nó cũng hữu ích cho các mục đích chẩn đoán.

Ghi chú

Bạn đã từng thắc mắc có bao nhiêu người đang sử dụng site của bạn vào lúc này? Nếu bạn đang sử dụng các tính năng hóa (personalization) của ASP.NET. Bạn sẽ dễ dàng ước lượng được điều này.

4.10.1 Bạn làm điều đó như thế nào?

Mỗi lần một user đăng nhập nhờ dùng một membership provider (đã được trình bày trong mục "Xác thực các User một cách dễ dàng"), ASP.NET ghi lại thời gian hiện hành trong data store. Khi cùng người dùng này yêu cầu một trang mới, ASP.NET cập nhật dấu thời gian theo cho phù hợp. Để dự đoán có bao nhiêu người đang sử dụng web site của bạn, bạn có thể tính số lượng user có một dấu thời gian (timestamp) trong một cửa sổ ngắn chỉ thời gian. Chẳng hạn, bạn có thể xem xét số lượng user đã yêu cầu một trang trong 15 phút qua.

Bạn có thể truy xuất thông tin này từ ASP.NET nhờ dùng phương thức GetNumberOfUsersOnline ( ) mới của lớp Membership. Bạn cũng có thể cấu hình cửa sổ thời gian sẽ được sử dụng bằng cách xác lập thuộc tính UserIsOnlineTimeWindow (phản ánh số phút). Nó được xác lập là 15 theo mặc định.

Sau đây một đoạn mã tính số lượng user online và hiển thị số này trong một nhãn:

```
Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
  lblStatus.Text &= "<br>There are " &_
  Membership.GetNumberOfUsersOnline( ) & _
    " users online right now. That is an estimate based" &_
    " on looking at timestamps that fall in the last " &_
    Membership.UserIsOnlineTimeWindow & _
    " minutes."
End Sub
```

Lưu ý rằng con số này không bao gồm các user vô danh.

4.10.2. Thế còn...

... việc lấy thông tin chính xác về những user nào đang online thì sao? Thật không may, ASP.NET hiện không có bất kỳ cách nào để xác định những user nào đang online. Cách duy nhất là thêm mã theo dõi riêng của bạn. Chẳng hạn, bạn có thể chứa thông tin này trong một cơ sở dữ liệu hay thêm nó vào một đối tượng trong bộ nhớ như tập hợp Applica-

tion bất kỳ khi nào một user đăng nhập. Bạn cũng có thể cần chứa thời gian login và loại bỏ các mục nhập cũ theo định kỳ.

4.11. Sử dụng sự xác thực theo vai trò

Trong nhiều ứng dụng web, mọi user không như nhau. Một số có thể được phép thực hiện chỉ một số hoạt động nhất định trong khi một số khác có thể thực hiện các tác vụ quản lý cao cấp hơn. ASP.NET 2.0 giúp thực hiện điều này dễ dàng hơn ASP.NET 1.x, nó gán các permission cho các nhóm user khác nhau nhờ dùng dịch vụ quản lý vai trò mới.

Ghi chú

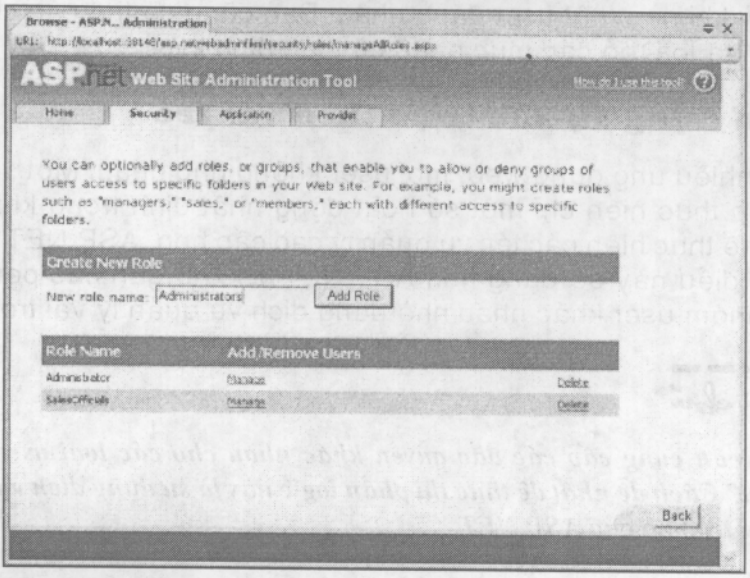
Bạn cần cung cấp các đặc quyền khác nhau cho các loại user khác nhau? Cách dễ nhất để thực thi phần logic này là sử dụng dịch vụ quản lý vai trò mới của ASP.NET.

4.11.1 Bạn làm điều đó như thế nào?

ASP.NET sử dụng một dịch vụ quản lý vai trò (role-manegement service) để quản lý sự lưu trữ và truy xuất thông tin theo vai trò (role). ASP.NET cho bạn sự linh hoạt để sử dụng các role-manager provider khác nhau nhằm chứa thông tin vai trò trong các nguồn dữ liệu khác nhau. Thông thường, bạn sẽ sử dụng cùng data store mà bạn sử dụng cho membership (đã được trình bày trong mục "Xác thực các User một cách dễ dàng"). Bởi vì membership provider và role-manager provider sử dụng các table khác nhau nên bạn không cần phải bận tâm về sự xung đột.

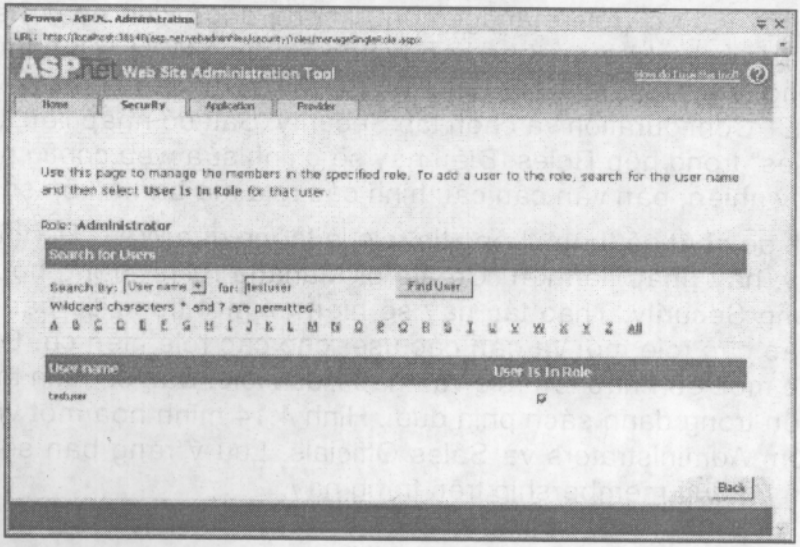
Quản lý vai trò (role management) không được kích hoạt theo mặc định. Bạn có thể kích hoạt nó bằng cách sử dụng WAT, như được trình bày trong mục "Quản lý một ứng dụng Web". Chỉ cần chọn Website > ASP.NET Configuration và chọn tab Security. Sau đó nhấp liên kết "Enable roles" trong hộp Roles. Điều này sẽ chỉnh sửa web.config như bạn cần. Tuy nhiên, bạn vẫn cần cấu hình các role mà bạn muốn sử dụng.

Cách dễ nhất để thêm thông tin role là thông qua WAT. Để thực hiện như vậy, hãy nhấp liên kết "Create or Manage roles" trong hộp Roles trên trang Security. Thao tác này sẽ hiển thị một trang để bạn có thể thêm vào các role mới và gán các user cho các role hiện có. Để thêm một role mới, gõ nhập tên role và nhấp Add Role. Bạn sẽ nhìn thấy role xuất hiện trong danh sách phía dưới. Hình 4.14 minh họa một ví dụ với hai nhóm Administrators và Sales Officials. Lưu ý rằng bạn sẽ không nhìn thấy group membership trên trang này.



Hình 4.14. Thêm một role mới

Để thay đổi group membership, hãy nhấp liên kết Manage kế bên role thích hợp. Bởi vì một hệ thống tiêu biểu có thể dễ dàng có hàng trăm user nên WAT không cố hiển thị tất cả user cùng một lần. Thay vào đó, nó cho phép bạn chỉ định user mà bạn muốn thêm vào role bằng cách gõ nhập tên, trình duyệt một thư mục theo bảng chữ cái, hay sử dụng một phép tìm với các ký tự thay thế (như trong John* để tìm các usrename bắt đầu với John). Ngay khi bạn tìm thấy user thích hợp, hãy đặt một dấu kiểm trong cột "User Is In Role" để thêm user vào role, hoặc xóa trọn hộp kiểm để loại bỏ user, như minh họa ở hình 4.15.



Hình 4.15. Gán một role (vai trò) cho một user

Nhờ sử dụng tab này, bạn có thể xem xét danh sách các user (bao gồm cả test user bạn đã thêm vào ở bước trước) và chỉnh sửa thông tin của chúng. Tuy nhiên, điều bạn thật sự cần làm là nhấp liên kết "Create access rules" để hạn chế sự truy cập vào thư mục Secured. Chọn thư mục trong danh sách, chọn tùy chọn Deny Permission, và chọn Anonymous users, như minh họa ở hình 4.13. Sau đó, nhấp OK để thêm quy tắc này.

Khi một user đăng nhập nhờ dùng forms authentication (như đã trình bày trong mục "Xác thực các User một cách dễ dàng"), dịch vụ role-management tự động truy xuất danh sách các role mà user thuộc về và chứa chúng trong một cookie đã được mã hóa. Sau lúc này, bạn có thể dễ dàng kiểm tra role membership của user đang được đăng nhập.

Chẳng hạn, mã sau đây kiểm tra user có phải là một Administrator hay không. Để đạt được điều này, user phải đã đăng nhập:

```
If Roles.IsUserInRole("Administrator") Then
    ' (Allow the code to continue, or show some content
    ' that would otherwise be hidden or disabled.)
End If
```

Và sau cùng, mã này hiển thị một danh sách tất cả các role mà user thuộc về:

```
For Each Role As String In Roles.GetRolesForUser( )
    lblRoles.Text &= Role & " "
Next
```

Rõ ràng, không cần phải thực hiện gì thêm đối với những tác vụ này!

Bạn cũng có thể xác lập và truy xuất thông tin role nhờ dùng lớp System.Web.Security.Roles. Sau đây là các phương thức chính mà bạn sẽ muốn sử dụng:

`CreateRole( )`

Phương thức này tạo một role mới với tên đã chỉ định trong cơ sở dữ liệu. Ghi nhớ rằng các role chỉ là các nhãn (như Administrator hay Guest). Nó tùy tuộc vào mã của bạn để quyết định cách phản hồi thông tin đó.

`DeleteRole( )`

Loại bỏ role khỏi nguồn dữ liệu.

AddUserToRole( )

Thêm một record trong data store vốn biểu thị user được chỉ định là thành viên của role đã xác định. Bạn cũng có thể sử dụng các phương thức khác vốn hoạt động với các mảng và cho phép bạn thêm một user vào nhiều role khác nhau cùng một lần, hoặc thêm nhiều user khác nhau vào cùng một role. Các phương thức này bao gồm AddUserToRoles ( ), AddUsersToRole ( ), và AddUsersToRoles ( ).

RemoveUserFromRole( )

Loại bỏ một user khỏi một role.

GetRolesForUser( )

Truy xuất một mảng các chuỗi vốn biểu thị tất cả các role mà một user thuộc về. Nếu bạn đang truy xuất các role cho user đang được đăng nhập, bạn không cần chỉ định một username.

GetUsersInRole( )

Truy xuất một mảng các chuỗi với mọi username trong một role đã cho.

IsUserInRole( )

Kiểm tra xem một user có ở một role nhất định nào đó hay không. Đây là nền tảng của việc tạo theo vai trò (role-based authorization). Tùy thuộc vào phương thức này cho ra true hay False, mã của bạn sẽ quyết định cho phép hay hạn chế những hành động nhất định. Nếu bạn đang kiểm tra group membership của user đang được đăng nhập, bạn không cần chỉ định username.

Đoạn mã sau đây tạo một role và thêm một user vào role đó:

```
Roles.CreateRole("Administrator")

Roles.AddUserToRole("testUser", "Administrator")
```

4.11.2. Thế còn...

... sự thực thi thì sao? Thoạt đầu, role Management có vẻ không có tính mở rộng. Việc đọc thông tin role (vai trò) cho mỗi yêu cầu web chắc chắn sẽ làm giảm chậm tốc độ của ứng dụng và thậm chí nó còn gây ra một trở ngại mới khi các luồng ASP.NET chờ đợi để truy cập vào cơ sở dữ liệu. Thật may mắn, dịch vụ role-management hoàn toàn thông minh. Nó không đi vòng đến cơ sở dữ liệu với mỗi yêu cầu web; thay vào đó, nó truy xuất thông tin role một lần, mã hóa nó và chứa nó trong một

cookie. Đối với tất cả các yêu cầu kế tiếp, ASP.NET đọc các role từ cookie đã được mã hóa. Bạn có thể loại bỏ cookie này vào bất kỳ lúc nào bằng cách gọi Roles.DeleteCookie ( ), hoặc bạn có thể cấu hình các xác lập trong file web.config để xác định khi nào nó sẽ hết hạn.

Nếu bạn có một số lượng role cực kỳ lớn, cookie có thể không chứa hết chúng. Trong trường hợp này, ASP.NET tạo cờ cho cookie để biểu thị điều đó. Khi mã của bạn thực hiện việc kiểm tra role, ASP.NET sẽ cố tương kết một trong các role trong cookie trước tiên, và nếu nó không tìm thấy phần tương kết, nó sẽ kiểm tra lại nguồn dữ liệu vào lần kế tiếp.

4.12. Chứa thông tin đã được cá nhân hóa

Các ứng dụng ASP.NET thường cần chứa thông tin về user nhiều hơn là chỉ username và password. Một cách để giải quyết vấn đề này là sử dụng tập hợp Session. Trạng thái sesstion (phiên làm việc) có hai hạn chế: nó không lâu dài (thường một session sẽ ngưng sau 20 phút không hoạt động), và nó không được tạo kiểu mạnh, nói cách khác, bạn cần biết những phần nào trong tập hợp session và ép kiểu bằng tay các phần tham chiếu đến các kiểu dữ liệu thích hợp). ASP.NET 2.0 khắc phục các hạn chế này bằng một framework mới để chứa dữ liệu về người dùng được gọi là các xác lập profile.

Ghi chú

Bạn cần chứa một số thông tin về user tùy ý trong các thời hạn lâu dài? Tại sao không sử dụng membership data provier để lưu và truy xuất thông tin không cần dùng đến mã cơ sở dữ liệu.

4.12.1 Bạn làm điều đó như thế nào?

Các profile được tạo dựa vào cùng mô hình provider được sử dụng để quản lý membership và role. Về cơ bản, profile provider đảm nhiệm việc chứa mọi thông tin liên quan đến người dùng (user) trong một data store backend. Hiện tại, ASP.NET có một profile provider đã được sửa đổi cho SQL Server.

Trước khi bắt đầu sử dụng các profile, bạn nên có một hệ thống tại chỗ để xác thực user. Đó là bởi vì thông tin đã được cá nhân hóa cần được liên kết với một user cụ thể để bạn có thể truy xuất nó vào những lần kế tiếp. Thông thường, bạn sẽ sử dụng forms authentication với sự trợ giúp của các dịch vụ membership ASP.NET đã được trình bày trong mục "Xác thực các User một cách dễ dàng".

Với các profile, bạn cần định rõ loại thông tin về user mà bạn muốn chứa. Trong các phần tạo trước, WAT có một công cụ để tạo các xác lập

profile. tuy nhiên, công cụ này đã biến mất trong các ấn bản sau này và trừ khi (hay cho đến khi) nó trở lại, bạn cần định rõ các xác lập profile trong file web.config bằng tay. Sau đây là một ví dụ về một mục profile vốn định rõ một chuỗi đơn với tên Fullname:

Ghi chú

ASP.NET có đưa vào các tính năng cơ bản nhằm cho phép bạn sử dụng sự cá nhân hóa với các user vô danh (xem mục "Thế còn..." dành cho phần này để tìm hiểu thêm).

```
<?xml version="1.0"?>
<configuration>
  <system.web>

    <profile>
      <properties>
        <add name="FullName" type="System.String" />
      </properties>
    </profile>

    <!— Other settings ommitted. —>
  </system.web>
</configuration>
```

Ban đầu, mã này không có vẻ hữu dụng hơn một xác lập ứng dụng. Tuy nhiên, Visual Studio tự động tạo một lớp mới dựa vào các xác lập profile của bạn. Bạn có thể truy cập lớp này thông qua thuộc tính Page.Profile. Ưu điểm khác là ASP.NET chứa thông tin trong một cơ sở dữ liệu backend, tự động truy xuất nó từ cơ sở dữ liệu vào lúc bắt đầu yêu cầu và viết nó trở lại vào cuối yêu cầu (nếu đây là những hoạt động cần thiết). Nói cách khác, các profile cho bạn một mô hình ở cấp độ cao hơn để bảo lưu thông tin về user vốn được chứa trong một cơ sở dữ liệu.

Nói cách khác, giả sử bạn đã định rõ thuộc tính FullName trong mục <profile>, bạn có thể xác lập và truy xuất thông tin tên của một user nhờ dùng mã sau đây:

```
Profile.FullName = "Joe Smythe"

...

lblName.Text = "Hello " & Profile.FullName
```

Lưu ý rằng lớp Profile được tạo kiểu mạnh. Không cần chuyển đổi phần tham chiếu, IntelliSense của Visual Studio thực hiện hành động khi bạn gõ Profile theo sau là một dấu chấm.

Mọi việc càng trở nên dễ dàng hơn nếu bạn muốn chứa một đối tượng đầy đủ tính năng. Chẳng hạn, hãy tưởng tượng bạn tạo các lớp chuyên biệt để theo dõi các sản phẩm trong giỏ mua sắm của một user. Ví dụ 4.6 minh họa một lớp Basket chứa một tập hợp các đối tượng BasketItem, mỗi đối tượng tiêu biểu cho một sản phẩm riêng biệt.

Ví dụ 4.6. Các lớp tùy ý cho một giỏ mua sắm (shopping cart)

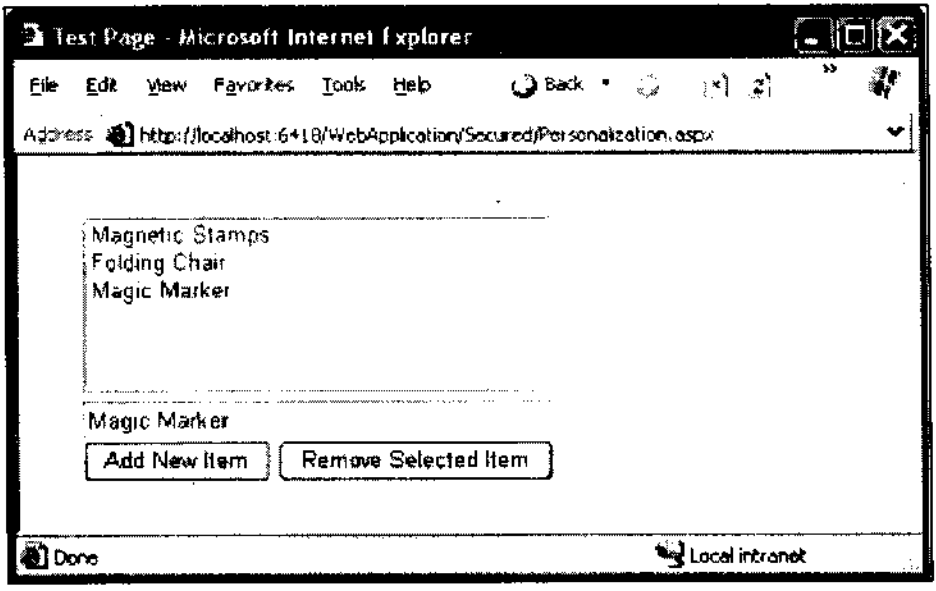
```
Imports System.Collections.Generic

Public Class Basket
    Private _Items As New List(Of BasketItem)
    Public Property Items( ) As List(Of BasketItem)
        Get
            Return _Items
        End Get
        Set(ByVal value As List(Of BasketItem))
            _Items = value
        End Set
    End Property
End Class

Public Class BasketItem
    Private _Name As String
    Public Property Name( ) As String
        Get
            Return _Name
        End Get
        Set(ByVal value As String)
            _Name = value
        End Set
    End Property

    Private _ID As String = Guid.NewGuid( ).ToString( )
    Public Property ID( ) As String
```

```
        Get
            Return _ID
        End Get
        Set(ByVal value As String)
            _ID = value
        End Set
    End Property

    Public Sub New(ByVal name As String)
        _Name = name
    End Sub

    Public Sub New( )
        ' Used for serialization.
    End Sub
End Class
```

Để sử dụng lớp này, bạn cần thêm nó vào thư mục con Code để nó được biên dịch tự động. Sau đó, để làm cho nó trở thành thành phần của profile user, bạn cần định rõ nó trong file web.config, như sau:

```
<profile>
    <properties>
        <add name="Basket" type="Basket" />
    </properties>
</profile>
```

Với thông tin này đã có, bạn có thể dễ dàng tạo một trang thử shopping card đơn giản. Hình 4.16 minh họa một ví dụ cho phép bạn thêm và loại bỏ các mặt hàng. Khi trang được tải lần đầu tiên, Nó kiểm tra xem có giỏ mua sắm cho user hiện tại hay không, và nếu không, nó tạo một giỏ mua sắm. Sau đó user có thể thêm các mặt hàng vào giỏ mua sắm của mình, hoặc loại bỏ các mặt hàng hiện có, sử dụng các nut Add và Remove. Sau cùng, tập hợp các mặt hàng trong giỏ mua sắm sẽ được kết hợp với một hộp danh sách mỗi lần trang được mô phỏng, bảo đảm trang hiển thị danh sách các mặt hàng hiện hành trong giỏ. Ví dụ 4.7 minh họa mã hoàn chỉnh.



Hình 4.16. Thêm các mặt hàng vào một giỏ mua sắm

Ví dụ 4.7. Kiểm tra một giỏ mua sắm đã được cá nhân hóa

```
<%@ Page language="VB" %>

<script runat="server">
    Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If Profile.Basket Is Nothing Then Profile.Basket = New Basket( )
    End Sub

    ' Put a new item in the basket.
    Sub cmdAdd_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        Profile.Basket.Items.Add(New BasketItem(txtItemName.Text))
    End Sub

    ' Remove the selected item.
    Sub cmdRemove_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        For Each item As BasketItem In Profile.Basket.Items
            If Item.ID = lstItems.SelectedItem.Value Then
                Profile.Basket.Items.Remove(Item)
                Return
```

```
        End If
      Next
    End Sub

    ' The page is being rendered. Create the list using data binding.
    Sub Page_PreRender(ByVal sender As Object, ByVal e As System.EventArgs)
      lstItems.DataSource = Profile.Basket.Items
      lstItems.DataTextField = "Name"
      lstItems.DataValueField = "ID"
      lstItems.DataBind( )
    End Sub

</script>

<html>
<head runat="server">
  <title>Test Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <br />
    <br />
    <asp:ListBox ID="lstItems" Runat="server" Width="266px"
      Height="106px"></asp:ListBox><br />
    <asp:TextBox ID="txtItemName" Runat="server"
      Width="266px"></asp:TextBox><br />
    <asp:Button ID="cmdAdd" Runat="server" Width="106px"
      Text="Add New Item" OnClick="cmdAdd_Click" />
    <asp:Button ID="cmdRemove" Runat="server" Width="157px"
      Text="Remove Selected Item" OnClick="cmdRemove_Click" />
  </form>
</body>
</html>
```

Ghi nhớ rằng thông tin profile không hết hạn. Điều đó có nghĩa là cho dù bạn tạo lại và khởi động lại ứng dụng web, các mặt hàng trong giỏ mua sắm vẫn còn ở đó trừ khi mã của bạn trực tiếp xóa chúng. Điều này làm cho các profile trở nên hoàn hảo cho việc lưu trữ lâu dài thông tin về user mà không cần phải bận tâm về sự rắc rối của mã ADO.NET.

4.12.2. Thế còn...

... các user vô danh thì sao? Theo mặc định, bạn có thể chỉ được truy cập thông tin profile ngay khi một user đã đăng nhập. Tuy nhiên, nhiều web site giữ lại thông tin về user ngay cả khi các user không được đăng nhập. Chẳng hạn, đa số các cửa hiệu thương mại điện thử online cho phép user bAmua sắm ngay lập tức và chỉ buộc họ đăng nhập vào lúc kiểm tra. Để thực thi kiểu thiết kế này (không cần trạng thái session), bạn cần sử dụng một tính năng mới khác trong ASP.NET, đó là anonymous identification.

Với anonymous identification (sự nhận dạng vô danh), ASP.NET gán một ID duy nhất cho mọi user mới. ID này được chứa trong một cookie lâu dài, nghĩa là cho dù một user chờ đợi nhiều ngày trước khi vào lại, ASP.NET vẫn có thể nhận ra user này và tìm thông tin đã cá nhân hóa từ lần truy cập cuối cùng của user. (Các xác lập hết hạn mặc định loại bỏ cookie sau khoảng một tuần nếu user không trở lại).

Để sử dụng anonymous identification, bạn cần thêm thẻ gán <anonymousIdentification> vào file web.config, và bạn cần cho biết rõ thông tin profile nào có thể được theo dõi vô danh bằng cách tạo cờ các thuộc tính này với thuộc tính allowAnonymous.

Sau đây là một ví dụ với web.config đã được sửa đổi vốn chứa thông tin giỏ mua sắm cho các user vô danh:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <anonymousIdentification enabled="true">

    <profile>
      <properties>
        <add name="Basket" type="Basket" allowAnonymous="true"/>
      </properties>
    </profile>

    <!-- Other settings ommitted. -->
```

```
</system.web>
</configuration>
```

Anonymous identification làm phát sinh vài vấn đề mới. Vấn đề quan trọng nhất xảy ra trong các hệ thống mà một user vô danh cần đăng nhập vào một lúc nào đó để hoàn chỉnh thao tác. Để bảo đảm thông tin không bị mất, bạn cần xử lý event PersonalizationModule.Migrate Anonymous trong file global.asax. Sau đó, bạn có thể chuyển thông tin này từ profile vô danh sang profile mới được xác thực.

Chương 5

Các File, Cơ sở dữ liệu và XML

.NET 1.0 đã cách mạng hóa sự truy cập dữ liệu Visual Basic với toàn bộ mô hình dữ liệu mới cho việc tương tác với các file, nối kết các cơ sở dữ liệu, và thu thập XML. Trong .NET.2.0, sự cách mạng hóa tiếp tục với một lô các cải tiến phụ, một số tính năng mới và một công cụ để tạo mã truy cập dữ liệu tự động, tất cả đều được thiết kế để tạo ra cuộc sống tốt đẹp hơn cho lập trình viên VB.

Thủ thuật

Điều bạn sẽ không học trong chương này là các tính năng mới .NET Framework được thiết kế cho SQL Server 2005. Những tính năng này bao gồm việc sử dụng mã .NET để lập trình các loại dữ liệu do user (người dùng) ấn định và các thủ tục được lưu trữ, các khai báo SQL Server 2005 và các nhóm bản tin đang hoạt động (MARS). Để biết thêm thông tin về những tính năng SQL Server 2005 này, hãy truy cập đến MSDN SQL Server 2005 Developer Center tại http://msdn.microsoft.com/SQL/2005

5.1 Lấy thông tin ổ đĩa

.NET bao gồm các lớp Directory Info và FileInfo thuận tiện cho việc thu thập thông tin về các file và các thư mục. Tuy nhiên, trong .NEt 1.x không có bất kỳ cách nào để lấy một danh sách về tất cả các ổ đĩa trên máy tính của bạn mà không sử dụng các chỉ lệnh không được quản lý đối với Windows API. Thật may, cuối cùng lớp DriveInfo đã bắt đầu xuất hiện trong .NET 2.0.

Ghi chú

Lớp DriveInfo giúp bạn dễ dàng truy xuất thông tin từ các ổ đĩa trên máy tính của bạn.

Thủ thuật

Trước khi đi xa hơn nữa, hãy bắt đầu bằng việc nhập namespace System.IO. Các lớp truy cập file trong .NET 2.0 (FileInfo, DirInfo và DriveInfo) tất cả đều tồn tại ở đó. Dù bạn có truy cập chúng trực tiếp hay thông qua đối tượng My, bạn cũng sẽ cần phải nhập namespace này.

5.1.1 Bạn làm điều đó như thế nào?

My.Computer.FileSystem cung cấp một phương pháp nhanh để lấy một danh sách về tất cả các ổ đĩa trên máy tính hiện hành. Tất cả những gì bạn cần phải làm là lặp vòng qua tập hợp Drives, tập hợp này trình bày một tập các đối tượng System.IO.DriveInfo. Ví dụ, để xem tất cả các ổ đĩa trên máy tính hiện hành, hãy gõ mã sau:

```
' Display a list of drives.

For Each Drive As DriveInfo In My.Computer.FileSystem.Drives

   Console.WriteLine(Drive.Name)

Next
```

Mã này ghi một danh sách bao gồm các tên mẫu tự ổ đĩa ("A:\", "C:\", "D:\",...). Bạn cũng có thể tạo một đối tượng DriveInfo cho một thư mục cụ thể bằng việc sử dụng phương thức My.Computer.FileSystem.GetDriveInfo( ), và định rõ ký tự của ổ đĩa bạn muốn kiểm tra ở dạng một chuỗi. Ví dụ, hãy thử mã sau đây để nhìn gần hơn vào ổ đĩa C:

```
Console.WriteLine("The label for drive C:\ is " & _

   My.Computer.FileSystem.GetDriveInfo("C").VolumeLabel
```

Ví dụ này hiển thị volume label được xác lập cho ổ đĩa. Bạn cũng có thể kiểm tra các thuộc tính khác của đối tượng DriveInfo để lấy nhiều thông tin hơn (chẳng hạn như DriveType, TotalFreeSpace và TotalSize). Nhưng hãy nhớ rằng bạn không thể truy xuất thông tin này đối với một ổ đĩa rời nếu không có phương tiện (ví dụ, nếu CD hay đĩa mềm không có trong ổ đĩa). Để bảo vệ chống lại khả năng này, hãy kiểm tra nhằm bảo đảm DriveType không bao gồm DriveType.Fixed trước khi cố lấy nhiều chi tiết hơn.

Ví dụ 5.1 đặt những khái niệm này cùng với một ứng dụng console đơn giản, hoàn tất vốn hiển thị thông tin về tất cả các ổ đĩa trên máy tính của bạn.

Ví dụ 5.1 Hiển thị thông tin về tất cả các ổ đĩa trên một máy tính

```
Imports System.IO

Module DriveInfoTest

   Public Sub Main( )
      Console.WriteLine("Drives on this computer:")
```

```
        For Each Drive As DriveInfo In My.Computer.FileSystem.Drives
            ' Display drive information.
            Console.WriteLine(Drive.Name)

            Console.WriteLine(vbTab & "Type: " & Drive.DriveType.ToString( ))

            If (Drive.DriveType And DriveType.Fixed) = DriveType.Fixed Then
                Console.WriteLine(vbTab & "Format: " & _
                  Drive.DriveFormat.ToString( ))
                Console.WriteLine(vbTab & "Label: " & Drive.VolumeLabel)
                Console.WriteLine(vbTab & "Total Size: " & Drive.TotalSize)
                Console.WriteLine(vbTab & "Free Space: " & Drive.TotalFreeSpace)
            End If
            Console.WriteLine( )
        Next
    End Sub

End Module
```

Khi bạn chạy mã này, bạn sẽ thấy kết xuất như sau:

```
Drives on this computer:
A:\
        Type: Removable

C:\
        Type: Fixed
        Format: NTFS
        Label: Applications
        Total Size: 15726702592
        Free Space: 2788483072

D:\
 ...
```

5.1.2 Thế còn ...

... lấy thông tin về phần còn lại của hệ thống file thì như thế nào? .NET Framework luôn tạo thuận lợi cho việc lấy thông tin thư mục và file

bằng cách sử dụng các đối tượng DirectoryInfo và FileInfo. Một khi bạn đã thể hiện cho một đối tượng DriveInfo, bạn có thể sử dụng các thuộc tính RootDirectory của nó để lấy một đối tượng DirectoryInfo nằm trong thư mục gốc (ví dụ, C:\). Sau đó, bạn có thể sử dụng các phương thức giống như DirectoryInfo.GetFiles( ) và DirectoryInfo.GetDirectories( ) để truy xuất các file và các thư mục con chứa trong thư mục gốc.

5.2 Lấy thông tin File và thư mục

Trong VB 2005, bạn có thể truy cập tất cả các thông tin file và thư mục bạn cần từ một điểm bắt đầu đơn giản: đối tượng My.Computer.FileSystem mới.

Ghi chú

Đối tượng My.Computer.FileSystem mới cho phép bạn lấy thông tin file và thư mục với một mã tối thiểu.

5.2.1 Bạn làm điều đó như thế nào?

Đây là bốn phương thức chính của My.Computer.FileSystem mà bạn có thể sử dụng để lấy thông tin file và thư mục. Mỗi phương thức có cùng ký danh, lấy một thông số chuỗi đơn giản mà trị của nó là đường dẫn hoàn chỉnh của file hay thư mục vốn là đối tượng trong truy vấn của bạn. Các phương thức là:

FileExists()

Trả về TRue nếu file tồn tại.

DirectoryExists( )

Trả về True nếu thư mục tồn tại.

GetFileInfo( )

Trả về một đối tượng FileInfo. Bạn có thể kiểm tra các thuộc tính khác nhau của nó để lấy thông tin chẳng hạn như kích thước file, các thuộc tính và,...

GetDirectoryInfo( )

Trả về một đối tượng DirectoryInfo. Bạn có thể kiểm tra các thuộc tính khác nhau của nó để lấy thông tin chẳng hạn như kích thước thư mục, các thuộc tính,....

Đoạn mã được trình bày ở ví dụ 5.2 trước tiên xác định một file có tồn tại hay không và sau đó hiển thị một số thông tin khi nó tồn tại.

Ví dụ 5.2 Truy xuất thông tin về một file cụ thể

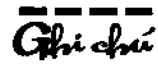
```
Imports System.IO

Module FileInfoTest

    Public Sub Main( )
        ' Get a file in a "special directory."
        Dim Info As FileInfo
        Info = My.Computer.FileSystem.GetFileInfo("c:\Windows\explorer.exe")

        ' Show the access/update times.
        Console.WriteLine("Created: " & Info.CreationTime)
        Console.WriteLine("Last Modified: " & Info.LastWriteTime)
        Console.WriteLine("Last Accessed: " & Info.LastAccessTime)

        ' Check if the file is read-only. When testing file attributes,
        ' you need to use bitwise arithmetic, because the FileAttributes
        ' collection usually contains more than one attribute at a time.
        Dim ReadOnlyFile As Boolean
        ReadOnlyFile = Info.Attributes And FileAttributes.ReadOnly
        Console.WriteLine("Read-Only: " & ReadOnlyFile)

        ' Show the size.
        Console.WriteLine("Size (bytes): " & Info.Length)
    End Sub

End Module
```

Đây là loại kết xuất bạn sẽ thấy:

```
Created: 3/30/2004 7:35:17 PM
Last Modified: 8/29/2002 4:41:24 AM
Last Accessed: 4/28/2004 10:59:38 AM
Read-Only: False
Size (bytes): 104032
Version: 6.0.1106
```

5.2.2 Tìm kiếm các thư mục và cần file

Đối tượng My.Computer.FileSystem cung cấp một phương thức GetDirectories( ) để truy xuất tên của tất cả các thư mục con trong một thư mục và một phương thức GetFile( ) để truy xuất tên của tất cả các file trong một thư mục được cho.



Trong những phiên bản beta đầu tiên, Visual Basic đã bao gồm các lớp mới FolderProperties và FileProperties sao lặp các lớp DirectoryInfo và FileInfo. Thật may thay, Microsoft quyết định không phát minh lại bánh xe và trở lại các chuẩn .NET 1.x.

Cả hai phương thức đều có thêm tính linh động bổ sung thông qua một phiên bản quá tải chấp nhận các thông số bổ sung. Bạn có thể định rõ một mảng với một hay nhiều chuỗi lọc (ví dụ, sử dụng *.doc để tìm tất cả các file với đuôi mở rộng .doc). Bạn cũng có thể cung cấp một thông số Boolean includeSubFolders, nếu TRue thì tìm kiếm các file hay thư mục tương xứng trong mỗi thư mục con được chứa.

Đây là một ví dụ minh họa về một phép tìm kiếm cao cấp tìm tất cả các file .exe trong thư mục c:\windows:

```
' Get all the EXE files in the Windows directory.

For Each File As String In My.Computer.FileSystem.GetFiles( _

 "c:\windows\", True, "*.exe")

   Info = My.Computer.FileSystem.GetFileInfo(File)

   Console.WriteLine(Info.Name & " in " & Info.Directory.Name)

Next
```

Nên chú ý rằng các phương thức GetFiles( ) và Getdirectories( ) chỉ trả về các chuỗi. Nếu bạn muốn biết thêm thông tin, bạn cần tạo một đối tượng FileInfo hay DirectoryInfo cho file hay thư mục như đã được trình bày ở trên.

Có một nguyên tắc: khi bạn thực hiện tìm kiếm với phương thức GetFiles(), danh sách file tương ứng được tạo đầu tiên rồi trả về mã của bạn. Nói cách khác, nếu bạn đang thực hiện một phép tìm chiếm nhiều thời gian, bạn sẽ không nhận được một kết quả đơn giản cho đến khi tiến trình tìm kiếm hoàn toàn kết thúc.

5.3 Sao chép, di chuyển và xóa các File

Ngoài việc giúp bạn thu thập thông tin về các thư mục và các file trên hệ thống của bạn, đối tượng My.Computer.FileSystem cho phép bạn truy cập nhanh đối với một số phương thức để thực hiện các tác vụ quản lý file chung, chẳng hạn như sao chép, di chuyển và xóa các file.

Trong VB 2005, bạn có thể thực hiện các tác vụ quản lý file chung với một dòng mã đơn.

5.3.1 Bạn làm điều đó như thế nào?

Đối tượng My.Computer.FileSystem cung cấp một số phương thức hoàn chỉnh cho việc thực hiện các tác vụ quản lý file chung. Đó là:

- CopyFile( ) và CopyDirectory( )
- MoveFile( ) và MoveDirectory( )
- RenameFile( ) và RenameDirectory( )
- DeleteFile( ) và DeleteDirectory( )

Cách bạn sử dụng mỗi phương thức này khá đơn giản. Bạn cung cấp hai thông số: một đường dẫn và (nếu được yêu cầu), một tên file hay đường dẫn đích. Ví dụ, bạn có thể đổi tên một file với dòng mã này:

```
My.Computer.FileSystem.RenameFile("c:\myfile.txt", "newname.txt")
```

Những phương thức này cũng có sẵn trong những phiên bản quá tải với cung cấp cho bạn những tính năng bổ sung. Chúng ta sẽ xem ở phần kế tiếp.

Các phương thức di chuyển và sao chép của FileSystem có sẵn trong nhiều phiên bản quá tải. Nếu bạn cần ghi đè một file hay thư mục đang hiện hữu, hãy nhớ sử dụng một thông số Boolean overwrite và gán nó với true. Nếu không, bạn sẽ nhận được một ngoại lệ và các mẫu sẽ không hoàn chỉnh. Đây là một minh họa về một tùy chọn như thế:

Ghi chú

Trong một số phiên bản beta, giao diện người dùng để di chuyển hay xóa một file không xuất hiện, thậm chí khi bạn chọn xem nó. Tuy nhiên, tác vụ bên dưới (di chuyển hay xóa file) luôn được thực hiện chính xác.

```
My.Computer.FileSystem.CopyDirectory("c:\MyFiles", _
  "c:\CopyOfMyFiles", True)
```

Điều đáng chú ý là các phương thức sao chép và xóa là các phiên

bản chấp nhận thông số Boolean showUI. Nếu thông số đó được xác lập với TRue, tác vụ diễn ra chính xác như thể một user đã khởi tạo tác vụ xóa hay sao chép trong Windows Explorer: các hộp thoại xuất hiện yêu cầu user xác nhận để nghị ghi đè hay xóa các file và một bộ chỉ báo tiến trình xuất hiện với một nút Cancel khi một tác vụ xóa hay sao chép file ở trong tiến trình (trừ khi tác vụ kết thúc rất nhanh). Thậm chí bạn có thể định rõ những gì nên xảy ra khi user nhấp vào Cancel (hoặc một ngoại lệ được loại bỏ hay không có điều gì xảy ra cả) sử dụng thông số onUserCancel.

Ví dụ 5.3 cung cấp một ứng dụng console hoàn chỉnh cho phép bạn kiểm tra cách xử lý này.

Ví dụ 5.3 Di chuyển và xóa các file với Window UI

```
Imports System.IO

Module FileManagement

    Public Sub Main( )
        ' Create a large test file (100 MB).
        Dim TestFile As String = "c:\test.bin"
        Console.WriteLine("Creating file...")
        Dim fs As FileStream = File.OpenWrite(TestFile)
        For i As Integer = 1 To 100000000
            fs.WriteByte(0)
        Next
        fs.Close( )

        ' Create the target directory.
        Console.WriteLine("Creating directory...")
        Dim TargetDir As String = "c:\TestDir"
        My.Computer.FileSystem.CreateDirectory(TargetDir)
        Dim TargetFile As String = Path.Combine(TargetDir, "test.bin")

        Console.WriteLine("Moving file...")
        ' Try moving the file. Set the following parameters:
        '    showUI = UIOption.AllDialogs
        ' (Show all the Windows UI, not just error messages.)
        '    onUserCancel = UICancelOption.ThrowException
```

```
        ' (Generate an error if the user clicks Cancel.)
        Try
            My.Computer.FileSystem.MoveFile(TestFile, TargetFile, _
              UIOption.AllDialogs, UICancelOption.ThrowException)
            Console.WriteLine("File moved.")
        Catch Err As Exception
            Console.WriteLine("You canceled the operation.")

            ' Remove the original file.
            My.Computer.FileSystem.DeleteFile(TestFile)
        End Try

        Console.WriteLine("Press Enter to continue.")
        Console.ReadLine( )

        ' Delete the target directory. Set the following parameters:
        '    showUI = UIOption.AllDialogs
        ' (Show the confirmation and Windows UI dialog box.)
        '    sendToRecycleBin = RecycleOption.SendToRecycleBin
        ' (Delete the file permanently.
        '    onUserCancel = UICancelOption.DoNothing
        ' (Allow the user to cancel this operation.)
        My.Computer.FileSystem.DeleteDirectory(TargetDir, _
          UIOption.AllDialogs, RecycleOption.SendToRecycleBin, _
          UICancelOption.DoNothing)

        Console.WriteLine("Cleanup finished.")
    End Sub

End Module
```

Như đã được trình bày trong ví dụ này, các phương thức DeleteFile( ) và DeleteDirectory() có một điểm bổ sung có sẵn. Theo mặc định, khi bạn xóa một file, nó bỏ qua Windows recycle bin (thùng rác Windows). Tuy nhiên, bạn có thể sử dụng một phiên bản quá tải của DeleteFile( ) hay DeleteDirectory( ) chấp nhận thông số sendToRecycleBin. Gán thông số này với True để giữ file xung quanh như một mã bảo vệ.

5.3.2 Các tác vụ file sử dụng các folder đặc biệt

Đối tượng My.Computer.FileSystem mới cho phép bạn truy xuất các quy chiếu đến nhiều folder do hệ thống định rõ thông qua lớp SpecialDirectories. Ví dụ, bạn có thể truy xuất nhanh đường dẫn cho các file tạm thời, các tài liệu user và nền màn hình. Đây là một minh họa:

```
Dim Desktop As String = My.Computer.FileSystem.SpecialDirectories.Desktop
Console.WriteLine("Your desktop is at: " & Desktop)

Console.Write("It's size is: ")
Console.Write(My.Computer.FileSystem.GetDirectoryInfo(Desktop).Size)
Console.WriteLine(" bytes")
Console.Write("It contains: ")
Console.Write(My.Computer.FileSystem.GetFiles(Desktop).Count)
Console.WriteLine(" files")
```

Lớp SpecialDirectories bao gồm tất cả các thuộc tính sau đây, mỗi thuộc tính trả về một chuỗi với toàn bộ đường dẫn tiêu chuẩn phù hợp:

```
AllUsersApplicationData
CurrentUserApplicationData
Desktop
MyDocuments
MyMusic
MyPictures
Programs
Temp
```

5.4 Đọc và ghi các File

Nếu bạn cần làm việc với các file text hay dữ liệu nhị phân thô, VB 2005 cung cấp một giải pháp mới bỏ qua các lớp mức thấp hơn của System.IO đối với các file nhỏ. Bây giờ, bạn có thể đọc và ghi các file text trong một tác vụ đơn bằng việc sử dụng đối tượng My.Computer.FileSystem. Tốt nhất là về sau bạn không cần tạo các stream (dùng), theo dõi vị trí hay xóa sạch.

Ghi chú

Sau hết, một cách để đọc và viết các file không cần phối hợp với các stream và các đầu đọc stream.

5.4.1 Bạn làm điều đó như thế nào?

Đối tượng My.Computer.FileIO cung cấp cách nhanh nhất để đọc và viết những nội dung của một file. Bí mật của nó nằm trong một số phương thức hoàn chỉnh. Bao gồm:

ReadAllText( )

Đọc nội dung của một file text và trả về nó như một chuỗi đơn.

ReadAllBytes( )

Đọc nội dung của bất kỳ file nào và trả về nó như một mảng của các byte.

WriteAllText( )

Ghi một mảng byte vào một file trong một tác vụ đơn. Bạn có thể hoặc thêm vào một file đang hiện hữu hoặc tạo một file mới, tùy thuộc vào bạn có cung cấp true hay False cho thông số Boolean append.

Ví dụ 5.4 tạo một file text đơn giản rồi đọc nó lại vào trong bộ nhớ.

Ví dụ 5.4 Viết một file trong một bước và đọc một file trong một bước

```
Imports System.IO

Module FileReadAndWrite

    Public Sub Main( )
        Dim Text As String = "This is line 1" & _
          vbNewLine & "This is line 2" & _
          vbNewLine & "This is line 3" & _
          vbNewLine & "This is line 4"

        ' Write the file.
        My.Computer.FileSystem.WriteAllText("c:\test.txt", Text, False)

        ' Read the file.
```

```
        Console.WriteLine(My.Computer.FileSystem.ReadAllText("c:\test.txt"))
    End Sub

End Module
```

5.4.2 Tìm hiểu thêm

Những phương thức bạn sẽ tìm thấy trong đối tượng My.Computer.FileSystem không phù hợp với sự tiện lợi hoàn toàn nhưng chúng không luôn thích hợp. Đây là một số lý do bạn có thể loại bỏ tốt hơn bằng việc sử dụng các lớp mức thấp hơn của namespace System.IO:

- Bạn có một file thật lớn và bạn muốn đọc rồi xử lý những nội dung của nó mỗi lần một phần nhỏ, thay vì tải toàn bộ file vào trong bộ nhớ cùng một lần. Đây là một phương cách hợp lý nếu bạn đang giải quyết một tài liệu dài.
- Bạn muốn sử dụng các kiểu dữ liệu khác, chẳng hạn như các con số hay ngày tháng. Để sử dụng các phương thức My.Computer.FileIO nhằm xử lý dữ liệu số, đầu tiên bạn sẽ cần chuyển đổi các con số thành các chuỗi hay các mảng byte bằng thao tác tay nhờ sử dụng các lớp .NET khác. Mặt khác, nếu bạn sử dụng một FileStream, bạn chỉ cần gói nó với một BinaryReader hay BinaryWriter.
- Bạn muốn sử dụng các tính năng .NET trên nền tảng stream khác chẳng hạn như nén, tạo chuỗi đối tượng hay mã hóa.

Các lớp .NET chính để đọc và viết các file được tìm thấy trong namespace System.IO và không được thay đổi trong .NET 2.0. Lớp hữu dụng nhất trong những lớp trên là FileStream (cho phép bạn mở một file trực tiếp để đọc hay viết), StreamReader và StreamWriter (được sử dụng cho việc đọc và viết text, chỉ một dòng tại một thời điểm) và BinaryReader và BinaryWriter (được sử dụng cho việc chuyển đổi các kiểu dữ liệu .NET cơ bản sang dữ liệu nhị phân và ngược lại). Hãy tra cứu những lớp này trong MSN Help cho các kỹ thuật truy cập file truyền thống. Ngoài ra, ở những phần tiếp theo, bạn sẽ thấy một ví dụ minh họa cao cấp hơn sử dụng FileStream để tạo mật mã dữ liệu trong một file.

5.5 Nén và giải nén dữ liệu

Ngay vả với dung lượng luôn gia tăng của các ổ đĩa cứng và việc tụt giá của bộ nhớ máy tính, cuối cùng nó vẫn tiết kiệm khoảng trống. Trong .NET 2.0, một namespace System.IO.Compression mới tạo thuận lợi cho

một lập trình viên VB 2005 dễ dàng nén dữ liệu khi cô ghi nó đến một stream và giải nén dữ liệu khi cô đọc nó từ một stream.

Ghi chú

Cần phải tiết kiệm khoảng trống trước khi bạn lưu trữ dữ liệu trong một file hay cơ sở dữ liệu? .NET 2.0 tạo thuận lợi cho việc nén và giải nén dễ dàng.

5.5.1 Bạn làm điều đó như thế nào?

Khoảng trống tên System.IO.Compression mới giới thiệu hai lớp stream mới: GZipStream và DeflateStream được sử dụng để nén và giải nén các stream của dữ liệu.

Các thuật toán được sử dụng bởi những lớp này không bị mất, có nghĩa là khi bạn nén và giải nén dữ liệu của bạn, bạn sẽ không làm mất bất kỳ thông tin nào.

Để nén, bạn cần phải hiểu rằng một stream nén gói một stream khác. Ví dụ, nếu bạn muốn ghi một số dữ liệu được nén vào một file, đầu tiên bạn tạo một FileStream cho file. Sau đó, bạn gói FileStream với GZipStream hay DeflateStream. Phương pháp như sau:

```
Dim fsWrite As New FileStream(FileName, FileMode.Create)

Dim CompressStream As New GZipStream(fsWrite, CompressionMode.Compress)
```

Bây giờ, nếu bạn muốn ghi dữ liệu vào một file. Bạn hãy sử dụng GZipStream. GZipStream nén dữ liệu đó rồi viết dữ liệu được nén vào FileStream được gói, sau đó ghi nó đến file bên dưới. Nếu bạn bỏ bớt tiến trình này và đi trực tiếp vào FileStream, cuối cùng bạn sẽ ghi dữ liệu không được nén.

Giống như tất cả các stream, GZipStream chỉ cho phép bạn ghi các byte thô. Nếu bạn muốn ghi các chuỗi hay các kiểu dữ liệu khác, bạn cần tạo một StreamWriter. StreamWriter chấp nhận các kiểu dữ liệu .NET cơ bản (như các chuỗi và các số nguyên) và chuyển đổi chúng sang các byte. Đây là một minh họa:

```
Dim Writer As New StreamWriter(CompressStream)

    ' Put a compressed line of text into the file.

Writer.Write("This is some text")
```

Cuối cùng, ngay khi bạn kết thúc, hãy bảo đảm bạn xóa sạch GZipStream để tất cả dữ liệu cuối cùng đều ở trong file:

```
Writer.Flush( )

CompressStream.Flush( )

fsWrite.Close( )
```

Tiến trình giải nén tương tự. Trong trường hợp này, bạn tạo một FileStream cho file bạn muốn đọc rồi tạo một GZipStream giải nén dữ liệu. Sau đó, bạn đọc dữ liệu bằng việc sử dụng GZipStream như sau:

```
fsRead = New FileStream(FileName, FileMode.Open)

Dim DecompressStream As New GZipStream(fsRead, CompressionMode.Decompress)
```

Ví dụ 5.5 trình bày một minh họa nối đuôi nhau ghi một số dữ liệu được nén đến một file, hiển thị lượng khoảng trống tiết kiệm được rồi giải nén dữ liệu.

Ví dụ 5.5 Nén và giải nén một file mẫu

```
Imports System.IO


Module FileCompression


    Public Sub Main( )
        ' Read original file.
        Dim SourceFile As String
        SourceFile = My.Computer.FileSystem.CurrentDirectory & "\test.txt"
        Dim fsRead As New FileStream(SourceFile, FileMode.Open)
        Dim FileBytes(fsRead.Length - 1) As Byte
        fsRead.Read(FileBytes, 0, FileBytes.Length)
        fsRead.Close( )


        ' Write to a new compressed file.
        Dim TargetFile As String
        TargetFile = My.Computer.FileSystem.CurrentDirectory & "\test.bin"
        Dim fsWrite As New FileStream(TargetFile, FileMode.Create)
        Dim CompressStream As New GZipStream(fsWrite, CompressionMode.Compress)
        CompressStream.Write(FileBytes, 0, FileBytes.Length)
        CompressStream.Flush( )
        CompressStream.Close( )
        fsWrite.Close( )
```

```
        Console.WriteLine("File compressed from " & _
          New FileInfo(SourceFile).Length & " bytes to " & _
          New FileInfo(TargetFile).Length & " bytes.")

        Console.WriteLine("Press Enter to decompress.")
        Console.ReadLine( )

        fsRead = New FileStream(TargetFile, FileMode.Open)
                Dim DecompressStream As New GZipStream(fsRead,
CompressionMode.Decompress)
        Dim Reader As New StreamReader(CType(DecompressStream, Stream))
        Console.WriteLine(Reader.ReadToEnd( ))
        Reader.Close( )
        fsRead.Close( )
    End Sub

End Module
```

5.5.2 Giải nén các file .zip như thế nào?

Thật không may, các stream nén .NET 2.0 không thể giải quyết với các file ZIP, file thường được sử dụng để rút lại các lô file (thường trước khi lưu trữ chúng dài hạn hay đính kèm chúng với một thông điệp email). Nếu bạn cần khả năng cụ thể này, có lẽ bạn sẽ quan tâm #ziplib có thể download miễn phí, có sẵn tại http://www.icsharpcode.net/OpenSource/SharpZipLib).

5.6 Thu thập số liệu thống kê trên những kết nối dữ liệu

Đa số lập trình viên thích xem các số liệu thống kê. Để xem xét cẩn thận, họ có thể gọi ra nguyên nhân tiềm ẩn trong một vấn đề có từ lâu, giải thích các vấn đề hiển thị của một ứng dụng hay gợi lên các kỹ thuật có thể tối ưu hóa. Nếu bạn đang sử dụng nhà cung cấp SQL Server, bạn có thể sử dụng SqlConnection.RetrieveStatistics( ) mới để lấy một hashtable (bảng băm) với một loạt các chi tiết chẩn đoán về kết nối cơ sở dữ liệu của bạn.

Ghi chú

Bạn muốn tìm ra những gì thật sự đang tiếp tục trong khi bạn kết nối với

một cơ sở dữ liệu? Trong .NET 2.0, bạn có thể thấy được nhiều thông tin hơn nhưng chỉ khi bạn đang sử dụng SQL Server.

5.6.1 Bạn làm điều đó như thế nào?

Trước khi bạn có thể gọi RetrieveStatistics( ), bạn cần chỉ dẫn nó để thu thập các số liệu thống kê bằng việc xác lập thuộc tính SqlConnection.StatisticsEnabled với TRue. Một khi bạn thực hiện bước này, lớp SqlConnection sẽ thu thập các số liệu thống kê cho mỗi lệnh cơ sở dữ liệu bạn thực thi qua kết nối. Nếu bạn thực hiện nhiều tác vụ với cùng kết nối, các số liệu thống kê sẽ tăng dần lên, cho dù bạn đóng kết nối giữa mỗi tác vụ.

Để xem các số liệu thống kê bất cứ lúc nào, bạn có thể gọi phương thức RetrieveStatistics( ) để truy xuất một hashtable chứa dữ liệu được thu thập. Hashtable này lập chỉ mục các thành phần của nó với một tên mô tả. Ví dụ, để truy xuất số giao dịch bạn đã thực hiện, bạn sẽ ghi mã này:

```
Dim Stats as Hashtable = con.RetrieveStatistics( )

Console.Writeline(Stats("Transactions"))
```

Để có được một ý kiến hay về các số liệu thống kê khác nhau có sẵn, hãy thử chạy ví dụ 5.6, một ứng dụng console lặp lại qua bộ sưu tập các số liệu thống kê và hiển thị tên khóa và giá trị của mỗi số liệu nó chứa.

Ví dụ 5.6 Truy xuất tất cả các số liệu thống kê nối kết

```
Imports System.Data.SqlClient

Module StatisticsTest

    Private ConnectString As String = _
      "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=SSPI"
    Private con As New SqlConnection(ConnectString)

    Public Sub Main( )
        ' Turn on statistics collection.
        con.StatisticsEnabled = True

        ' Perform two sample commands.
        SampleCommand( )
```

```
        SampleCommand( )
        ' Retrive the hashtable with statistics.
        Dim Stats As Hashtable = con.RetrieveStatistics( )

        ' Display all the statistics.
        For Each Key As String In Stats.Keys
            Console.WriteLine(Key & " = " & Stats(Key))
        Next
    End Sub

    Private Sub SampleCommand( )
        con.Open( )
        Dim cmd As New SqlCommand("SELECT * FROM Customers", con)
        Dim reader As SqlDataReader = cmd.ExecuteReader( )
        reader.Close( )
        con.Close( )
    End Sub

End Module
```

Đây là một danh sách hoàn chỉnh về các số liệu thống kê được tạo ra bởi mã này:

```
NetworkServerTime = 18
BytesReceived = 46248
Transactions = 0
SumResultSets = 2
SelectCount = 2
PreparedExecs = 0
ConnectionTime = 13
CursorFetchCount = 0
CursorUsed = 0
Prepares = 0
CursorFetchTime = 0
UnpreparedExecs = 2
SelectRows = 182
ServerRoundtrips = 2
CursorOpens = 0
```

```
BuffersSent = 2
ExecutionTime = 725
BytesSent = 108
BuffersReceived = 6
IduRows = 0
IduCount = 0
```

Để xác lập lại các giá trị của bộ sưu tập các số liệu thống kê sang zero bất cứ lúc nào, chỉ đơn giản gọi phương thức ResetStatistics( ):

```
con. ResetStatistics( )
```

5.6.2 Ý nghĩa của các số liệu thống kê được thu thập và đưa chúng vào sử dụng

Thật không may, MSDN Help chưa cung cấp đầy đủ về các số liệu thống kê SQL Server. Tuy nhiên, một số liệu thống kê đặc biệt hữu ích và không quá khó để thông dịch:

BytesReceived

Cho một ảnh chụp nhanh tổng số các byte được truy xuất từ server cơ sở dữ liệu.

ServerRoundtrips

Cho biết số lệnh riêng biệt bạn đã thực thi.

ConnectionTime

Cho biết lượng thời gian tích lũy mà kết nối được mở.

SumResultSets

Cho biết số truy vấn bạn đã thực hiện.

SelectRows

Ghi tổng số hàng được truy xuất trong mỗi truy vấn bạn đã thi hành. (Ở minh họa trước tổng số là 182, bởi vì mỗi truy vấn đã truy xuất 91 dòng).

5.7 Sắp xếp thành nhóm các lệnh Adapter dữ liệu để thực thi tốt hơn

Nhiều cơ sở dữ liệu có thể thực thi các lệnh trong các lô, làm giảm tổng số các chỉ lệnh bạn cần tạo. Ví dụ, nếu bạn đệ trình 10 lệnh cập nhật trong một lô đơn, mã của bạn chỉ cần tạo một trip đến server (thay vì 10). Việc cắt lớp số trip vòng có thể làm gia tăng hiệu suất, nhất là

trên các mạng có độ trễ cao. Trong .NET 2.0, SqlDataAdapter được cải tiến để sử dụng kỹ thuật tạo là cho việc cập nhật, chèn và xóa các mẩu tin.

Ghi chú

Nếu bạn cần một cách dễ dàng để tối ưu hóa các phần cập nhật DataSet, kỹ thuật sắp xếp thành lô mới của ADO.NET có thể giúp bạn.

5.7.1 Bạn làm điều đó như thế nào?

Ở những phiên bản trước của .NET, bạn có thể sắp xếp thành lô các lệnh trực tiếp bằng việc ghép nối chúng trong một chuỗi đơn và tách mỗi lệnh bằng một dấu chấm phẩy. Cú pháp này đòi hỏi phần hỗ trợ từ nhà cung cấp cơ sở dữ liệu, nhưng nó làm việc hoàn hảo với SQL Server. Đây là một ví dụ minh họa chèn hai dòng trong một bảng:

```
Dim TwoInserts As String ="INSERT INTO Shippers" &_
"(CompanyName, Phone) VALUES "ACME", "212-111-1111;" & _
"INSERT INTO Shippers (CompanyName, Phone)" &_
VALUES "Grey Matter", "416-123-4567"

Dim cmd As New SqlCommand(TwoInsert)
cmd.ExecuteNonQuery( )
```

Tính năng này thật hữu ích, những phiên bản trước của .NET không cung cấp bất kỳ cách nào để sắp xếp thành lô các lệnh đến một trong những nhà cung cấp ADO.NET quan trọng nhất, các đối tượng adapter dữ liệu. Đối tượng adapter dữ liệu quét một DataSet và thực thi các lệnh chèn, xóa và cập nhật bất cứ lúc nào nó tìm thấy một dòng mới đã bị xóa hay di chuyển. Mỗi lệnh này được thực thi riêng biệt, có nghĩa là nếu DataSet chứa ba dòng mới thì adapter dữ liệu sẽ tạo ba trip vòng đến server.

Ghi chú

Cần tạo ra ý nghĩa tốt để có phần hỗ trợ sắp xếp thành các lô trong adapter dữ liệu, bởi vì adapter dữ liệu thường được sử dụng để sửa đổi nhiều hơn tại một thời điểm.

.NET 2.0 cải thiện hình ảnh với một thuộc tính SqlDataAdapter. UpdateBatchSize mới. Theo mặc định, trị của thuộc tính này được gán với 1, tạo ra mỗi lệnh chèn, cập nhật hay xóa được thực thi riêng biệt.

Nếu bạn xác lập UpdateBatchSize với một con số lớn hơn, adapter dữ liệu sẽ gom nhóm các lệnh của nó thành các lô.

Ví dụ 5.7 là một ứng dụng console, BatchedDataAdapterTest, đưa kỹ thuật này vào kiểm tra. BatchedDataAdapterTest truy xuất dữ liệu từ bảng Orders trong cơ sở dữ liệu Northwind rồi thay đổi mỗi dòng. Để làm cho cuộc sống thú vị, module ứng dụng cập nhật này không chỉ một lần mà hai lần, một lần không sắp xếp thành các lô và một lần với các kích thước được xác lập với 15. BatchedDataAdapterTest hiển thị các số liệu thống kê kết nối cho mỗi phương thức, cho phép bạn so sánh hiệu suất của chúng.

Ví dụ 5.7 Cập nhật có và không có sắp xếp thành các lô

```
Imports System.Data.SqlClient

Module BatchedDataAdapterTest

   Private ConnectString As String = _
     "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=SSPI"
   Private con As New SqlConnection(ConnectString)

   Public Sub Main( )
      ' Turn on statistics collection.
      con.StatisticsEnabled = True

      Dim Query As String = "SELECT * FROM Orders"
      Dim cmd As New SqlCommand(Query, con)
      Dim Adapter As New SqlDataAdapter(cmd)
      Dim CommandBuilder As New SqlCommandBuilder(Adapter)

      Dim ds As New DataSet
      con.Open( )
      Adapter.Fill(ds, "Orders")
      con.Close( )

      ' Perform an update without batching.
      ChangeRows(ds)
      con.ResetStatistics( )
      Adapter.Update(ds, "Orders")
```

```
        Console.WriteLine("Statistics without batching....")
        DisplayStatistics( )

        ' Perform an update with batching (15 row batches).
        ChangeRows(ds)
        con.ResetStatistics( )
        Adapter.UpdateBatchSize = 15
        ' When performing a batch update you must explicitly
        con.Open( )
        Adapter.Update(ds, "Orders")
        con.Close( )

        Console.WriteLine("Statistics with batching....")
        DisplayStatistics( )
    End Sub

    Public Sub ChangeRows(ByVal ds As DataSet)
        For Each Row As DataRow In ds.Tables("Orders").Rows
            Row("ShippedDate") = DateTime.Now
        Next
    End Sub

    Public Sub DisplayStatistics( )
        ' Retrive the hasthable with statistics.
        Dim Stats As Hashtable = con.RetrieveStatistics( )

        ' Display all the statistics.
        For Each Key As String In Stats.Keys
            Console.WriteLine(Key & " = " & Stats(Key))
        Next
        Console.WriteLine( )
    End Sub

End Module
```

Khi bạn chạy ứng dụng này, các dòng sẽ được cập nhật và một danh sách các số liệu thống kê sẽ xuất hiện. Hãy ngưng xem các số liệu

thống kê này, đặc biệt chú ý đến số các trip vòng được tạo cho cơ sở dữ liệu, tổng thời gian kết nối và lượng dữ liệu được yêu cầu để hoàn tất các phần cập nhật. Đây là một phần của kết xuất được tạo ra bằng việc chạy ứng dụng nêu bật một số con số quan trọng hơn:

```
Statistics without batching....
ConnectionTime = 5682
UnpreparedExecs = 831
ServerRoundtrips = 831
BytesSent = 2637094

Statistics with batching....
ConnectionTime = 6319
UnpreparedExecs = 56
ServerRoundtrips = 56
BytesSent = 1668160
```

Phần trên liệt kê các báo cáo trong cập nhật sắp xếp thành các lô là 831 dòng được cập nhật trong 56 lô, mỗi lô gồm 15 lệnh. Như bạn có thể thấy đấy, việc sắp xếp thành lô giảm bớt dữ liệu cần được gởi (bằng cách đóng gói nó thành các lô hiệu quả hơn), là một trong những metric quan trọng nhất của khả năng chỉnh cỡ cơ sở dữ liệu. Mặt khác, toàn bộ hiệu suất của ứng dụng khó thay đổi và thời gian kết nối thậm chí tăng ít. Rõ ràng là để làm một quyết định có ý nghĩa về việc có sử dụng việc sắp xếp theo lô hay không, bạn cần tạo profile cho ứng dụng của bạn trong một tình huống thực tế.

5.7.2 Những đặc tính và những hạn chế của các cập nhật được sắp xếp theo lô như thế nào?

Hiện tại, chỉ SqlDataAdapter là hỗ trợ sắp xếp theo lô mặc dù các nhà cung cấp khác có thể hoàn thiện tính năng này trong tương lai. Những chi tiết bổ sung thật sự sẽ khác nhau đối với mỗi nhà cung cấp trong trường hợp của SqlDataAdapter, nhà cung cấp sử dụng thủ tục lưu trữ hệ thống sp_executesq1 để thực thi việc sắp xếp theo lô. Đối với những đặc tính, bạn sẽ nhận thấy một sự thay đổi về cách các event (sự kiện) RowUpdated và RowUpdating của SqlDataAdapter làm việc như thế nào. Khi việc sắp xếp theo lô được cho phép, những sự kiện này phát sinh một lần đối với mỗi lô, không phải một lần đối với mỗi dòng. Điều đó có nghĩa là khi sự kiện RowUpdated phát sinh, bạn có thể xác định số dòng chịu ảnh hưởng, chứ không phải từng dòng chi tiết của những thay đổi được tạo ra. Sự tổn thất thông tin này có thể gây khó khăn hơn cho việc xử lý các lỗi xảy ra ở nơi nào đó bên trong một lô.

Kích thước lô chuẩn mực tùy thuộc vào nhiều nhân tố cấp thấp, bao gồm kiến trúc mạng và kích thước các dòng. Lời khuyên tốt nhất là kiểm tra ứng dụng của bạn với các xác lập lô khác nhau. Nếu bạn muốn tất cả các cập nhật được thực hiện trong một lô đơn với thước không hạn chế, hãy gán thuộc tính UpdateBatchSize với 0.

5.8 Sao chép khối dòng từ một bảng đến một bảng khác

Đa số những người dùng SQL Server gurus đều quen thuộc với tiện ích dòng lệnh BCP, tiện ích này cho phép bạn di chuyển các lượng thông tin lớn từ một cơ sở dữ liệu SQL Server đến một cơ sở dữ liệu khác. BCP tiện lợi bất cứ lúc nào bạn cần tải số mẩu tin lớn ngay lập tức, nhưng nó đặc biệt hữu ích khi bạn cần chuyển dữ liệu giữa các server. Trong .NET 2.0, khoảng trống tên SqlClient bao gồm một lớp SqlBulkCopy cho phép bạn thực hiện một tác vụ sao chép lượng lớn.

Ghi chú

Lớp SqlBulkCopy cung cấp cho bạn cách sao chép hiệu quả nhất các lượng dữ liệu lớn giữa các bảng hay các cơ sở dữ liệu.

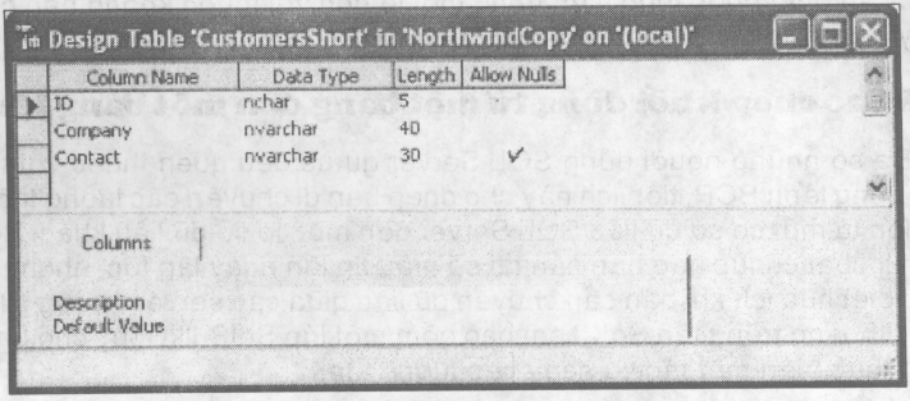
5.8.1 Bạn làm điều đó như thế nào?

Thành phần chính trong một tác vụ sao chép phần lớn là lớp SqlBulkCopy mới. Nó thực hiện tất cả những công việc của nó khi bạn gọi phương thức WriteToServer(), có thể được sử dụng bằng hai cách:

- Bạn có thể đệ trình dữ liệu như một DataTable hay một mảng của các đối tượng DataRow. Điều này hợp lý nếu bạn muốn chèn một lô các mẩu tin từ một file bạn đã tạo trước đó. Nó cũng làm việc tốt nếu bạn đang tạo một thành phần server phụ (giống như một dịch vụ web) nhận một DataSet kết nối với những mẩu tin cần được tải vào trong một bảng.
- Bạn có thể đệ trình dữ liệu của bạn như một DataReader mở kéo các mẩu tin từ một SqlConnection khác. Phương pháp này lý tưởng nếu bạn muốn chuyển các mẩu tin từ một server cơ sở dữ liệu sang một server khác.

Trước khi bạn gọi WriteToServer( ), bạn cần tạo các kết nối và các lệnh bạn cần đến và xác lập ánh xạ giữa bảng đích và bảng nguồn. Nếu các bảng nguồn và bảng đích hoàn toàn phù hợp, không có ánh xạ nào được yêu cầu. Tuy nhiên, nếu các tên bảng khác nhau, bạn cần xác lập thuộc tính SqlBulkCopy.DestinationTableName với tên của bảng đích. Ngoài ra, nếu các tên cột không phù hợp hay nếu có các cột trong bảng đích ít hơn trong dữ liệu nguồn, bạn cũng cần cấu hình việc ánh xạ cột.

Để xác lập ánh xạ cột, bạn thêm một đối tượng ánh xạ cho mỗi cột sang SqlBulkCopy.ColumnMappings. Mỗi đối tượng ánh xạ định rõ tên của cột nguồn và tên của cột đích tương ứng.



Hình 5.1 Tạo một bảng CustomersShort

Để tiến hành kiểm tra điều này, hãy tạo một cơ sở dữ liệu SQL Server mới có tên là NorthwindCopy và một bảng có tên là CustomerShort. Bảng CustomerShort được thiết kế để đưa ra một nhóm thông tin con trong bảng Customers. Bạn có thể tạo nó bằng cách sử dụng một công cụ như SQL Server Enterprise Manager (xem các xác lập cột ở hình 5.1), hay bạn có thể sử dụng script kèm với nội dung có thể tải xuống đối để tạo nó một cách tự động (tìm kiếm file Generate NorthwindCopy.sql).

Một khi bạn đã tạo CustomerShort, bạn có một bảng hoàn chỉnh cho việc kiểm tra một tác vụ sao chép lượng lớn.SQL Server. Tất cả những gì bạn cần làm là tạo hai kết nối, định rõ việc ánh xạ và bắt đầu tiến trình. Ví dụ 5.8 có mã bạn cần đến.

Ví dụ 5.8 Sử dụng SqlBulkCopy

```
Imports System.Data.SqlClient

Module Module1

    Private ConnectSource As String = _
     "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=SSPI"
    Private ConnectTarget As String = _
     "Data Source=localhost;Initial Catalog=NorthwindCopy;" &_

    Public Sub Main( )
        ' Create the source and target connections.
```

```
Dim conSource As New SqlConnection(ConnectSource)
Dim conTarget As New SqlConnection(ConnectTarget)

' Create a command for counting the number of rows in a table.
Dim cmdCount As New SqlCommand("SELECT COUNT(*) FROM CustomersShort", _
  conTarget)

' Initialize the SqlBulkCopy class with mapping information.
Dim BCP As New SqlClient.SqlBulkCopy(conTarget)
BCP.DestinationTableName = "CustomersShort"
BCP.ColumnMappings.Add("CustomerID", "ID")
BCP.ColumnMappings.Add("CompanyName", "Company")
BCP.ColumnMappings.Add("ContactName", "Contact")

' Count the rows in CustomersShort.
conTarget.Open( )
Dim Rows As Integer = CInt(cmdCount.ExecuteScalar( ))
Console.WriteLine("CustomersShort has " & Rows & " rows.")
Console.WriteLine("Starting bulk copy...")

' Retrieve the rows you want to transfer.
conSource.Open( )
Dim cmd As New SqlCommand( _
"SELECT CustomerID,CompanyName,ContactName FROM Customers", conSource)
Dim reader As SqlDataReader = cmd.ExecuteReader( )

' Write the data to the destination table.
BCP.WriteToServer(reader)

' Clean up.
BCP.Close( )
reader.Close( )
conSource.Close( )

' Count the rows in CustomersShort again.
conSource.Open( )
```

```
        Rows = CInt(cmdCount.ExecuteScalar( ))
        Console.WriteLine("Finished bulk copy.")
        Console.WriteLine("CustomersShort has " & Rows & " rows.")

        conTarget.Close( )
        Console.ReadLine( )
    End Sub

End Module
```

Khi bạn chạy mã, bạn sẽ thấy kết xuất như sau, kết xuất này cho biết tác vụ sao chép lượng lớn đã hoàn tất một cách thành công:

```
CustomersShort has 0 rows.
Starting bulk copy...
Finished bulk copy.
CustomersShort has 91 rows.
```

5.8.2 Những thuộc tính SqlBulkCopy khác là gì?

SqlBulkCopy cung cấp hai thuộc tính hữu ích: SqlBulkCopyTimeout (cho phép bạn xác lập bạn sẽ chờ đợi một server không phản hồi bao lâu) và BatchSize (cho phép bạn xác lập bao nhiêu tác vụ được xếp theo lô với nhau). Các lỗi được xử lý bằng cách tương tự như khi bạn thực thi trực tiếp một SqlCommand. Cách khác, nếu một lỗi xảy ra ở phía server (như một xung đột trị duy nhất), tiến trình sẽ bị ngắt ngay tức khắc, và bạn sẽ nhận được một ngoại lệ SqlClient với đầy đủ chi tiết.

5.9 Ghi mã cơ sở dữ liệu bất khả tri

Trong việc phát triển ADO.NET, Mirosoft tạo một kiến trúc truy cập dữ liệu mới sẽ linh hoạt hơn, thực thi tốt hơn và có thể mở rộng dễ dàng hơn những kiến trúc ADO và OLE DB trên nền tảng COM trước. Họ làm điều này bằng việc tạo một mô hình mỗi nguồn dữ liệu phải cung cấp data provider (bộ cung cấp dữ liệu) riêng của nó: một nhóm lớp được quản lý cho phép bạn kết nối với một nguồn dữ liệu cụ thể (chẳng hạn SQL Server, Oracle), thực thi các lệnh và truy xuất dữ liệu. Để bảo đảm rằng những provider này là nhất quán, mỗi provider phải hoàn thành một nhóm giao diện chuẩn. Tuy nhiên, phương pháp này tạo những thách thức chủ yếu cho các nhà phát triển muốn ghi mã nhà provider bất khả tri, chẳng hạn một thường trình cơ sở dữ liệu cơ bản có thể được sử dụng tốt như với nhà cung cấp SQL Server hay nhà cung cấp Oracle. Thông

thường, bạn sử dụng mã provider bất trả tri bởi vì bạn không chắc chắn loại cơ sở dữ liệu nào mà phiên bản cuối cùng của một ứng dụng sẽ sử dụnghay bởi vì bạn dự đoán nhu cầu di chuyển đến một cơ sở dữ liệu khác trong tương lai.

Ghi chú

Bạn có muốn một cách viết mã cơ sở dữ liệu không hạn mức đối với một nguồn dữ liệu cụ thể? Thách thức này trở nên dễ dàng hơn nhiều trong .NET 2.0.

.NET 2.0 lấy những bước chính để dễ dàng tạo mã cơ sở dữ liệu phổ biến bằng việc giới thiệu một mô hình factory mới. Một mô hình factory là một mẫu mà một lớp chịu trách nhiệm thể hiện các lớp khác). Trong mô hình này, bạn có thể sử dụng một nhóm cung cấp cơ sở dữ liệu để tạo các kết nối ADO.NET, các lệnh và nhiều kiểu đối tượng khác được yêu cầu đối với một cơ sở dữ liệu cụ thể. Factory tự động trả về kiểu đối tượng bạn cần cho nguồn dữ liệu của bạn (chẳng hạn một SqlCommand hay OracleCommand), nhưng khi bạn ghi mã của mình, bạn sẽ không lo lắng về những chi tiết này. Thay vào đó, bạn ghi các lệnh chung không liên quan đến những chi tiết cụ thể của nguồn dữ liệu.

5.9.1 Bạn làm điều đó như thế nào?

Trong mã provider bất trả khi, bạn vẫn sử dụng tất cả những đối tượng được gõ như thế. Tuy nhiên, mã của bạn thu thập những đối tượng này bằng việc sử dụng các giao diện chung. Ví dụ, mỗi đối tượng lệnh, dù được sử dụng cho SQL Server hay Oracle, thực hiện giao diện IDbCommand chung, bảo đảm một nhóm các phương thức và các thuộc tính cơ bản.

Ghi chú

Bởi vì mã provider bất khả tri cố càng phổ biến càng tốt, vì thế nó khó tối ưu một cơ sở dữ liệu chính xác hơn. Kết quả là kỹ thuật này không phù hợp vì hầu hết các ứng dụng kinh doanh qui mô lớn.

Mã provider bất khả tri được tổ chức để bạn định rõ kiểu cơ sở dữ liệu bạn đang sử dụng, thường bằng việc đọc một số thông tin từ một file cấu hình. Bạn sử dụng thông tin này để truy xuất một DbProviderFactory cho cơ sở dữ liệu của bạn. Đây là một ví dụ minh họa chuỗi factory được mã hóa cứng:

```
Dim Factory As String = "System.Data.SqlClient"
```

```
Dim Provider As DbProviderFactory

Provider = DbProviderFactories.GetFactory(Factory)
```

Ở ví dụ này, mã sử dụng phương thức GetFactory( ) dùng chung của lớp System.Data.Common.DbProviderFactories. Nó định rõ một chuỗi nhận dạng tên provider. Ví dụ, nếu bạn sử dụng chuỗi System.Data.SqlClient, phương thức GetFactory( ) trả về một đối tượng System.Data.SqlClient.SqlClientFactory. Lớp DbProviderFactories có thể tạo các factory cho tất cả những provider dữ liệu được bao gồm với .NET, bởi vì chúng được định hình rõ ràng trong file cấu hình machine.config trên máy tính hiện hành. Về cơ bản, mẩu tin cấu hình yêu cầu lớp DbProviderFactories tạo một SqlClientFactory khi lập trình viên chuyển chuỗi chính xác "System.Data.SqlClient". Nếu bạn phát triển provider cho riêng mình, cũng có thể đăng ký nó để làm việc theo cách này.

Đối tượng SqlClientFactory có các phương thức được cài sẵn để tạo tất cả các đối tượng được sử dụng bởi nhà cung cấp SQL Server. Tuy nhiên, mã của bạn có thể hoàn toàn phổ biến. Thay vì tương tác với kiểu lớp SqlClientFactory cụ thể, nó sẽ sử dụng lớp cơ sở DbProviderFactory. Theo cách đó mã của bạn có thể làm việc với bất kỳ loại nào của DbProviderFactory và do đó hỗ trợ bất kỳ provider cơ sở dữ liệu nào.

Một khi bạn có DbProviderFactory, bạn có thể tạo các kiểu đối tượng ADO.NET được gõ khác, sử dụng một nhóm các phương thức chung bằng việc sử dụng các phương thức CreateXxx( ). Bao gồm:

```
CreateConnection( )

CreateCommand( )

CreateParameter( )

CreateDataAdapter( )

CreateCommandBuilder( )
```

Tất cả những phương thức này cần một phiên bản về provider của đối tượng chúng định danh.

Để hiểu rõ hơn về mã cơ sở dữ liệu chung hoạt động như thế nào, bạn cần tiến hành kiểm tra một ví dụ hoàn chỉnh có thể chuyển đổi từ một provider dữ liệu sang một provider dữ liệu tạm thời. Trước hết, bạn cần tạo một file cấu hình ứng dụng lưu trữ tất cả những chi tiết về provider cụ thể. Để làm điều này, hãy tạo một ứng dụng console và mở file app.config. Hãy thêm vào ba xác lập sau đây nhằm định rõ tên factory, chuỗi kết nối cho cơ sở dữ liệu và truy vấn để thực thi:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="Factory" value="System.Data.SqlClient" />
    <add key="Connection" value=
  "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=SSPI" />
    <add key="Query" value="SELECT * FROM Orders" />
  </appSettings>
</configuration>
```

Ví dụ này sử dụng provider SQL Server để kết nối cơ sở dữ liệu Northwind và truy xuất một danh sách về tất cả những mẩu tin trong bảng Orders.

Bây giờ, bạn có thể truy xuất thông tin file cấu hình và sử dụng nó với lớp DbProviderFactories để tạo mỗi đối tượng provider ADO.NET bạn cần. Trong ví dụ 5.9, truy vấn được thực thi, một DataSet được lấp đầy, và một danh sách các trị OrderID được hiển thị trong cửa sổ console.

Ví dụ 5.9 Sử dụng DbProviderFactories để ghi mã cơ sở dữ liệu bất khả tri.

```
Imports System.Data.Common
Imports System.Configuration

Module GenericDatabaseTest

    Public Sub Main( )
        ' Get all the information from the configuration file.
        Dim Factory, Connection, Query As String
        Factory = ConfigurationManager.AppSettings("Factory")
        Connection = ConfigurationSettings.AppSettings("Connection")
        Query = ConfigurationManager.AppSettings("Query")

        ' Get the factory for this provider.
        Dim Provider As DbProviderFactory
        Provider = DbProviderFactories.GetFactory(Factory)

        ' Use the factory to create a connection.
        Dim con As DbConnection = Provider.CreateConnection( )
```

```
        con.ConnectionString = Connection

        ' Use the factory to create a data adapter
        ' and fill a DataSet.
        Dim Adapter As DbDataAdapter = Provider.CreateDataAdapter
        Adapter.SelectCommand = Provider.CreateCommand( )
        Adapter.SelectCommand.Connection = con
        Adapter.SelectCommand.CommandText = Query
        Dim ds As New DataSet
        Adapter.Fill(ds. "Orders")

        ' Display the retrieved information.
        For Each Row As DataRow In ds.Tables("Orders").Rows
            Console.WriteLine(Row("OrderID"))
        Next
    End Sub


End Module
```

Hầu hết đây là một mẫu logic truy cập dữ liệu. Phần thú vị duy nhất là bạn có thể chuyển đổi từ một provider sang một provider khác mà không sửa đổi bất kỳ mã nào hay biên dịch lại. Bạn chỉ cần sửa đổi thông tin nhà cung cấp và chuỗi kết nối trong file cấu hình. Ví dụ, thực hiện những thay đổi cho file cấu hình để truy cập cùng bảng thông qua giao diện provider OLE DB chậm hơn:

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>
  <appSettings>
    <add key="Factory" value="System.Data.OleDb" />
    <add key="Connection" value=
  "Provider=SQLOLEDB;Data Source=localhost;Initial Catalog=Northwind;Integrated
  Security=SSPI" />
    <add key="Query" value="SELECT * FROM Orders" />
  </appSettings>

</configuration>
```

Sau khi lưu file cấu hình, bạn có thể chạy lại ứng dụng. Nó cũng hoạt động tốt, hiển thị cùng danh sách các mẩu tin thứ tự.

5.9.2 Bạn sẽ gặp phải những thách thức nào trong việc viết các chương trình cơ sở dữ liệu bất khả thi?

Phương thức factory mới là một bước tiến lớn cho những người muốn ghi mã provider bất khả tri. Tuy nhiên, vẫn còn một số vấn đề (một số không quan trọng và một số quan trọng hơn) vẫn giữ nguyên. Bao gồm:

Xử lý các lỗi

Mỗi provider cơ sở dữ liệu đều có đối tượng ngoại lệ của nó (giống như SqlException và OracleException) và những đối tượng này không phái sinh từ một lớp cơ sở chung. Điều đó có nghĩa là không có cách nào để viết một trình xử lý ngoại lệ bắt được những ngoại lệ cơ sở dữ liệu. Tất cả những gì bạn có thể làm là viết trình xử lý ngoại lệ bắt được đối tượng Exception.

Tính năng provider cụ thể

Một số tính năng không hiển thị thông qua các giao diện chung. Ví dụ, SQL Server có khả năng thi hành các truy vấn FOR XML trả về các tài liệu XML. Để thực thi kiểu truy vấn này, bạn hãy sử dụng phương thức SqlCommand.ExecuteXmlReader( ). Thật không may, đây không phải là một phương thức lệnh chuẩn, vì thế không có cách nào truy cập nó thông qua giao diện IDbCommand.

Xử lý các thông số

Một số provider (như SQL Server) nhận ra các thông số lệnh qua tên của chúng. Những provider khác (như OLE DB) nhận ra các thông số lệnh theo thứ tự, xuất hiện của chúng. Những khác biệt không quan trọng như vậy có thể cản trở việc lập trình provider bất trả tri.

5.10 Sử dụng New XPathDocument và XPathNavigator

.NET cung cấp một dãy tùy chọn để giải quyết XML trong các khoảng trống tên System.Xml. Một sự lựa chọn phổ biến là XmlDocument cho phép bạn định hướng XML trong bộ nhớ như một bộ thu thập các đối tượng mắc nối. Để việc thực thi hiệu quả hơn, các lớp XmlWriter và XmlReader cung cấp một cách hữu hiệu hơn để đọc và ghi một stream của XML. Thật không may, cũng không có giải pháp nào hoàn hảo. XmlDocument tận dụng quá nhiều bộ nhớ và việc định hướng cấu trúc của nó yêu cầu quá nhiều mã. Hơn nữa, bởi vì XmlDocument được dựa trên một chuẩn bên thứ ba (XML DOM hay mô hình đối tượng tài liệu), thật khó để cải tiến nó mà không phá vỡ tính tương thích. Mặt khác, XmlWriter và XmlReader quá hạn chế, buộc bạn phải truy cập thông tin từ lúc bắt đầu đến kết thúc. Chúng cũng gây khó khăn cho một nhà phát

triển cung cấp một giao diện XML cho dữ liệu không phải là XML.

Ghi chú

XPathDocument được sửa đổi xác lập một chuẩn mới cho việc phân ngữ XML trong .NET.

.NET 2.0 đưa ra một giải pháp với System.XmlXPath.XPathDocument. XPathDocument là một đầu đọc XML dựa trên con trỏ cố trở thành giao diện XML duy nhất mà bạn cần sử dụng. Nó tạo thuận lợi cho bạn trong việc di chuyển từ bất kỳ vị trí nào trong một tài liệu và cản trở cấp độ khi sử dụng với các chuẩn XML chẳng hạn như XQuery, XPath, XSLT và hợp chuẩn XML Schema.

5.10.1 Bạn làm điều đó như thế nào?

Để sử dụng một XPathDocument, bạn bắt đầu bằng việc tải tài liệu từ một stream, XmlReader, hay URI (có thể chứa một đường dẫn file hay địa chỉ Internet). Để tải nội dung, bạn có thể sử dụng phương thức Load() hay một đối số khởi tạo, cả hai đều làm việc trong cùng cách. Ở ví dụ minh họa này, XPathDocument được lấp đầy với nội dung từ một file cục bộ:

```
Dim Doc As New XPathDocument("c:\MyDocument.xml")
```

Để thật sự di chuyển xung quanh một XPathDocument, bạn cần tạo một XPathNavigator bằng việc gọi phương thức CreateNavigator( )

```
Dim Navigator As XPathNavigator = Doc.CreateNavigator( )
```

XPathNavigator bao gồm một nhóm phương thức phong phú cho việc định hướng cấu trúc của tài liệu XML. Một số phương thức bao gồm:

MoveToRoot( )

Nhảy đến gốc hay phần tử tài liệu chứa tất cả phần tử khác

MoveToID( )

Di chuyển đến một phần tử có một ID cụ thể, như được nhận dạng với thuộc tính ID.

MoveToNext( )

Di chuyển đến nút kế tiếp tại cùng mức (về mặt kỹ thuật được gọi là anh em).

MoveToPrevious( )

Di chuyển đến nút trước tại cùng mức (về mặt kỹ thuật được gọi là anh em)

MoveToFirstChild( )

Di chuyển xuống một mức đến nút đầu tiên được chứa bởi nút hiện hành.

MoveToParent( )

Di chuyển lên một mức đến bố chứa nút hiện hành.

Một khi bạn định vị trên một phần tử, bạn có thể đọc tên phần tử từ thuộc tính Name. Bạn có thể truy xuất nội dung text được chứa từ thuộc tính Value.

Bây giờ, bạn đã biết nhiều về điều này, thật đáng để thử một ví dụ minh họa cơ bản. Chúng ta sẽ sử dụng một tài liệu XML chứa một catalog sản phẩm dựa trên ASP.NET Commerce Starter Kit của Microsoft. File XML này có cấu trúc được trình bày ở ví dụ 5.10.

Ví dụ 5.10 XML mẫu đối với một catalog sản phẩm

```
<?xml version="1.0" standalone="yes"?>
<Products>
  <Product>
    <ProductID>356</ProductID>
    <ModelName>Edible Tape</ModelName>
    <ModelNumber>STKY1</ModelNumber>
    <UnitCost>3.99</UnitCost>
    <CategoryName>General</CategoryName>
  </Product>
  <Product>
    <ProductID>357</ProductID>
    <ModelName>Escape Vehicle (Air)</ModelName>
    <ModelNumber>P38</ModelNumber>
    <UnitCost>2.99</UnitCost>
    <CategoryName>Travel</CategoryName>
  </Product>
  ...
</Products>
```

Ví dụ 5.11 tải tài liệu này, tạo một XPathNavigator và di chuyển qua các nút, tìm kiếm một phần tử <ModelName> cho mỗi <Product>. Khi phần tử đó được tìm thấy, trị của nó được hiển thị.

Ví dụ 5.11 Định hướng một tài liệu XML với XPathNavigator

```
Imports System.Xml.XPath
Imports System.Xml

Module XPathNavigatorTest

    Sub Main( )
        ' Load the document.
        Dim Doc As New XPathDocument( _
          My.Computer.FileSystem.CurrentDirectory & _
          "\ProductList.xml")

        ' Navigate the document with an XPathNavigator.
        Dim Navigator As XPathNavigator = Doc.CreateNavigator( )

        ' Move to the root <Products> element.
        Navigator.MoveToFirstChild( )

        ' Move to the first contained <Product> element.
        Navigator.MoveToFirstChild( )

        ' Loop through all the <Product> elements.
        Do
            ' Search for the <ModelName> element inside <Product>
            ' and display its value.
            Navigator.MoveToFirstChild( )
            Do
                If Navigator.Name = "ModelName" Then
                    Console.WriteLine(Navigator.Value)
                End If
            Loop While Navigator.MoveToNext( )

            ' Move back to the <Product> element.
```

```
            Navigator.MoveToParent( )
        Loop While Navigator.MoveToNext( )
    End Sub


End Module
```

Khi bạn chạy mã này, bạn sẽ thấy một màn hình với một danh sách các tên mô hình cho tất cả những sản phẩm.

Thật thú vị, XPathNavigator cũng cung cấp việc phân loại cho các trị dữ liệu. Thay vì truy xuất trị hiện hành như một chuỗi bằng việc sử dụng thuộc tính Value, bạn có thể sử dụng một trong những thuộc tính tự động chuyển đổi trị sang kiểu dữ liệu khác. Các thuộc tính được hỗ trợ bao gồm:

ValueAsBoolean

ValueAsDateTime

ValueAsDouble

ValueAsInt

ValueAsLong

Để tiến hành kiểm tra điều này, bạn có thể ghi lại vòng lặp trong ví dụ 5.11 để nó chuyển đổi price (giá) sang một trị gấp đôi rồi hiển thị tổng số với thuế giá trị gia tăng:

```
Do
    If Navigator.Name = "ModelName" Then
        Console.WriteLine(Navigator.Value)
    ElseIf Navigator.Name = "UnitCost" Then
        Dim Price As Double = Navigator.ValueAsDouble * 1.15
        Console.WriteLine(vbTab & "Total with tax: " & Math.Round(Price, 2))
    End If
Loop While Navigator.MoveToNext( )
```

5.10.2 Những cách khác để tìm kiếm một tài liệu XML với XPathNavigator

Bạn có thể chọn một phần của tài liệu XML để làm việc trong một XPathNavigator. Để lựa chọn phần này, bạn hãy sử dụng các phương thức Select( ) hay SelectsingleNode( ) của lớp XPathNavigator. Cả hai

phương thức này yêu cầu một biểu thức XPath nhận dạng các nút bạn muốn truy xuất.

Ví dụ, mã sau lựa chọn phần tử <ModelName> cho mỗi sản phẩm trong hạng mục Tools:

```
' Use an XPath expression to get just the nodes that interest you
' (in this case, all product names in the Tools category).
Dim XPathIterator As XPathNodeIterator
XPathIterator = Navigator.Select ( _
  "/Products/Product/ModelName[../CategoryName='Tools']")

Do While (XPathIterator.MoveNext( ))
    ' XPathIterator.Current is an XPathNavigator object pointed at the
    ' current node.
    Console.WriteLine(XPathIterator.Current.Value)
Loop
```

Thủ thuật

Những ví dụ minh họa trong phần này sử dụng một tài liệu XML không có khoảng trống tên. Tuy nhiên, các khoảng trống tên thường được sử dụng trong việc lập trình các viễn cảnh cho phép chương trình của bạn nhận dạng duy nhất kiểu tài liệu nó quy chiếu. Nếu tài liệu của bạn sử dụng các khoảng trống tên, bạn cần sử dụng lớp XmlNamespaceManager và ghi lại các biểu thức XPath để sử dụng một tiền tố khoảng trống tên. Nếu bạn muốn một ví dụ minh họa về kỹ thuật này, hãy tham khảo các mẫu có thể tải xuống được đối với phần này, trình bày một minh họa với một hạng mục sản phẩm sử dụng các khoảng trống tên XML.

Giới thiệu XPath

Cú pháp XPath cơ bản sử dụng một hệ thống ký hiệu giống như đường dẫn để mô tả các vị trí trong một tài liệu. Ví dụ, đường dẫn /Products/Product/ModelName cho biết một phần tử ModelName được lồng trong một phần tử Product, rồi phần tử này được lồng trong một phần tử gốc Products. Đây là một đường dẫn tuyệt đối (nó bắt đầu với một dấu suyệt đơn, tiêu biểu cho gốc tài liệu).

Bạn cũng có thể sử dụng các đường dẫn tương đối, vấn tìm các nút với tên được cho bất kể chúng ở đâu. Các đường dẫn liên quan bắt đầu với hai dấu suyệt. Ví dụ, //ModelName sẽ tìm tất cả các phần tử

ModelName bất kể chúng ở đâu trong hệ thống cung cấp tài liệu. Những ký tự đường dẫn khác bạn có thể sử dụng bao gồm dấu chấm (.) ám chỉ nút hiện tại; hai dấu chấm (..) để di chuyển lên một mức; và dấu hoa thị (*) để lựa chọn bất kỳ nút nào.

XPath thật sự đem lại thú vị khi bạn bắt đầu thêm các điều kiện lọc. Các điều kiện lọc được bổ sung cho một đường dẫn trong dấu ngoặc vuông. Ví dụ, biểu thức XPath //Productc[CategoryName="Tools"] tìm tất cả các phần tử Product chứa một phần tử CategoryName với text "Tools". Bạn có thể sử dụng toàn bộ dãy toán tử logic, chẳng hạn như nhỏ hơn hay lớn hơn (< và >) hoặc không bằng (!=). Để biết thêm thông tin về thế giới tuyệt vời của XPath, hãy tra cứu XML trong một Nushell (O'Reilly).

Lưu ý

XPathNavigator có thể hiệu chỉnh chịu những thay đổi mở rộng và các tính năng được trình bày ở phần kế tiếp (mục 5.11) không làm việc trong cấu trúc cuối cùng mà chúng ta đã kiểm tra. Mặc dù nó được mong đợi để trở lại, những tính năng này đôi khi được cắt ngay cả ở giai đoạn trễ này. Nếu việc tạo mã mô hình thay đổi, bạn sẽ tìm thấy mã được cập nhật trong các ví dụ có thể tải xuống.

5.11 Hiệu chỉnh một tài liệu XML với XPathNavigator

XPathNavigator là giao diện XML được lựa chọn cho các ứng dụng Visual Basic 2005. Và trong .NET 2.0, nó không chỉ hoạt động như một kiểu xem trong dữ liệu XML chỉ đọc mà nó còn cho phép bạn thay đổi các tài liệu XML, chẳng hạn như bằng việc sửa đổi nội dung text, chèn các phần tử mới hay xóa một nhánh các nút.

Ghi chú

Tính năng tốt nhất của XPathNavigator là phần hỗ trợ mới của nó đối với việc hiểu chỉnh và chèn nội dung.

5.11.1 Bạn làm điều đó như thế nào?

Ở phần trước, mục 5.10, bạn đã biết cách tải dữ liệu XML vào trong một XPathDocument rồi duyệt qua và tìm kiếm thông qua nó bằng cách sử dụng một XPathNavigator. Nếu bạn muốn thay đổi, bạn vẫn bắt đầu với cùng XPathDocument. Bí quyết là bạn cũng sử dụng hai phương thức bổ sung trong XPathNavigator:

SetValue( )

Phương thức này chèn một trị mới trong phần tử hiện tại, thay thế trị đang hiện hữu.

DeleteCurrent( )

Phương thức này xóa nút hiện tại khỏi tài liệu.

Hãy nhớ rằng bạn đã có hai lựa chọn cơ bản cho việc tạo XPathNavigator:

Sử dụng phương thức XPathDocument.CreateNavigator( )

Phương thức này trả về XPathNavigator cho toàn bộ tài liệu. Sau đó, bạn có thể di chuyển đến phần tài liệu bạn muốn thay đổi.

Sử dụng phương thức XPathDocument.Select( ) với một biểu thức XPath.

Phương thức này trả về một XPathNodeIterator cho phép bạn di chuyển qua các kết quả của bạn, truy xuất một XPathNavigator cho mỗi nút được lựa chọn.

Ví dụ 5.12 sửa đổi tài liệu XML được trình bày ở ví dụ 5.10. Nó làm gia tăng toàn bộ giá cả lên 10% rồi xóa các nút không rơi vào hạng mục Tools. Cuối cùng, nó hiển thị tài liệu XML đã được thay đổi.

Ví dụ 5.12 Sửa đổi một tài liệu XML với XPathNavigator

```
Imports System.Xml.XPath
Imports System.Xml

Module XPathNavigatorTest

    Sub Main( )
        ' Load the document.
        Dim Doc As New XPathDocument(My.Computer.FileSystem.CurrentDirectory & _
          "\ProductList.xml")

        ' Use the XPathNavigator to make updates.
        Dim XPathIterator As XPathNodeIterator = Doc.Select("//UnitCost")

        ' Increase the price by 10%.
        For Each Editor As XPathNavigator In XPathIterator
            Editor.SetValue((1.1 * Editor.ValueAsDouble).ToString( ))
        Next
```

```
        ' Delete nodes that aren't in the Tools category.
        XPathIterator = Doc.Select("/Products/Product[CategoryName!='Tools']")
        For Each Editor As XPathNavigator In XPathIterator
            Editor.DeleteCurrent( )
        Next

        ' Show changes.
        XPathEditor.MoveToRoot( )
        Console.WriteLine(XPathEditor.OuterXml)
    End Sub

End Module
```

Khi bạn chạy ứng dụng này, XML đối với tài liệu đã thay đổi được hiển thị trong cửa sổ console. Bạn cũng có thể mở file ProductList_new.xml nơi những thay đổi đã được lưu.

Trong nhiều trường hợp, bạn không chỉ muốn thay đổi một trị mà bạn sẽ cần một cách để chèn những phần tử mới hay toàn bộ các phần. XPathNavigator bao gồm một nhóm các phương thức để chèn các phần tử và các thuộc tính mới trong một phần. Tuy nhiên, cách dễ nhất để thêm một khối XML là sử dụng một XmlWriter. Nếu bạn đã làm việc với XML và .NET trước đây, có lẽ bạn nhận ra XmlWriter. XmlWriter thường được sử dụng để viết nội dụng XML trực tiếp vào một file trong các ứng dụng .NET 1.x. Sự khác biệt trong .NET 2.0 là XPathNavigator cho phép bạn sử dụng XmlWriter ghi trực tiếp vào XPathDocument trong bộ nhớ của bạn.

Tất cả những gì bạn cần làm là bắt đầu bằng việc gọi một trong những phương thức XPathEditor trả về một XmlWriter. Những phương thức này bao gồm:

AppendChild( )

Thêm một phần tử mới bên trong phần tử hiện tại, sau tất cả các phần tử con đang hiện hữu.

PrependChild( )

Thêm một phần tử mới trong phần tử hiện hành, trước bất cứ phần tử con nào đang hiện hữu.

InsertAfter( )

Thêm một phần tử mới sau phần tử hiện hành (và cùng mức)

InsertBefore( )

Thêm một phần tử mới ngay trước phần tử hiện hành (và cùng mức).

Ví dụ 5.13 sử dụng phương thức AppendChild( ) để thêm một sản phẩm mới vào tài liệu XML danh sách sản phẩm.

Ví dụ 5.1.3 Sử dụng phương thức AppendChild( ) để thêm một phần tử mới vào một tài liệu XML.

```
Imports System.Xml.XPath
Imports System.Xml

Module XPathNavigatorTest

    Sub Main( )
        ' Load the document.
        Dim Doc As New XPathDocument(My.Computer.FileSystem.CurrentDirectory & _
          "\ProductList.xml")

        ' Create a new product.
        Dim XPathEditor As XPathNavigator = Doc.CreateEditor( )
        XPathEditor.MoveToRoot( )
        XPathEditor.MoveToFirstChild( )

        ' Use the XmlWriter to add a new <Product> complete with
        ' all child elements.
        Dim Writer As XmlWriter = XPathEditor.AppendChild

        ' Insert the opening <Product> tag.
        Writer.WriteStartElement("Product", _
          "http://www.ibuyspy.com/ProductCatalog")

        ' The WriteElementString( ) method inserts a whole element at once.
        Writer.WriteElementString("ProductID", "999")
        Writer.WriteElementString("ModelName", "Rubber Pants")
        Writer.WriteElementString("ModelNumber", "NOZ999")
        Writer.WriteElementString("UnitCost", "12.99")
        Writer.WriteElementString("CategoryName", "Clothing")
```

```
        ' Insert the closing </Product> tag and close the writer.
        Writer.WriteEndElement( )
        Writer.Close( )

        ' Show changes.
        XPathEditor.MoveToRoot( )
        Console.WriteLine(XPathEditor.OuterXml)
    End Sub

End Module
```

Việc chạy ví dụ 5.13 sẽ tạo XML sau, được hiển thị trong cửa sổ console và được lưu vào file XML mới được tạo:

```
...
  <Product>
    <ProductID>999</ProductID>
    <ModelName>Rubber Pants</ModelName>
    <ModelNumber>NOZ999</ModelNumber>
    <UnitCost>12.99</UnitCost>
    <CategoryName>Clothing</CategoryName>
  </Product>
...
```

Ghi chú

Bạn có thể tạo nhiều đối tượng hiệu chỉnh và định hướng để làm việc với cùng XPathDocument. Tuy nhiên, các bộ soạn thảo không thực hiện bất kỳ việc khóa, vì thế bạn không thể hiệu chỉnh một XPathDocument trên nhiều chuỗi cùng một lúc trừ khi bạn sử dụng những phương pháp bảo vệ riêng của mình.

5.11.2 Hợp chuẩn XML

Cả XPathNavigator và XmlWriter buộc bạn ghi XML hợp chuẩn. Tuy nhiên, cũng quan trọng để kiểm tra tài liệu XML nhằm bảo đảm chúng có phù hợp với các nguyên tắc cụ thể hay không. Công cụ tốt nhất đối với tác vụ này là một tài liệu giản đồ XML định rõ các phần tử, cấu trúc, các kiểu dữ liệu và những hạn định đối với một tài liệu.

Chuẩn giản đồ thật sự ở ngoài phạm vi chương này. Tuy nhiên, giả sử bạn có một giản đồ cho XML của bạn, bạn có thể hợp chuẩn tài liệu của bạn bất cứ lúc nào bằng việc gọi XPathNavigator.CheckValidity( ). Phương thức này trả về true nếu tài liệu phù hợp với giản đồ. Đây là cách thực hiện nó:

```
' Load the document.
Dim Doc As New XPathDocument("c:\ProductList.xml")

' (Make updates).

' Load the schema.
' Technically, you can load a collection of schemas,
' one for each namespace in the document that you want to validate.
Dim Schemas As New XmlSchemaSet( )
Schemas.Add("http://www.ibuyspy.com/ProductCatalog", "c:\ProductListSchema.xsd")
Schemas.Compile( )

' Validate with the schema.
' Instead of submitting a null reference (Nothing), you can supply
' a delegate that points to a callback method that will be triggered
' every time an error is found when the validation check is performed.
Dim Valid As Boolean
Valid = Doc.CreateNavigator( ).CheckValidity(Schemas, Nothing)
```

Chương 6

Các dịch vụ nền .NET 2.0

Ở những chương trước, bạn đã tìm hiểu về những thay đổi sâu xa nhất trong .NET 2.0, bao gồm những tính năng mới trong các ứng dụng web Windows Forms, ASP.NET và truy cập dữ liệu ADO.NET. Thật ra, các nhà phát triển Microsoft đã làm việc chăm chỉ trong việc vận dụng và điều chỉnh tốt toàn bộ thư viện lớp .NET. Nếu bạn quan sát chung quanh, bạn sẽ tìm thấy các thành phần, các kiểu và các namespace mới phát sinh ở mọi nơi.

6.1 Theo dõi các sự kiện dễ dàng

Khi một điều nào đó gặp sự cố trong ứng dụng của bạn, user hiếm khi ở một tư thế để xử lý sự cố. Thay vì hiển thị một hộp thông báo chi tiết, điều quan trọng hơn là bảo đảm tất cả những chi tiết được ghi lại ở một nơi nào đó không thay đổi, vì thế bạn có thể kiểm tra chúng sau đó để cố tìm ra sự cố. Ở những phiên bản trước của .NET, việc theo dõi thì đơn giản nhưng vô vị. Trong VB 2005, hoạt động này trở nên dễ dàng hơn nhờ có đối tượng My.Application.Log.

6.1.1 Bạn làm điều đó như thế nào?

Bạn có thể sử dụng đối tượng My.Application.Log mới để ghi nhanh một file XML, một file text thông thường, hay sổ theo dõi sự kiện Windows.

Để ghi một thông báo theo dõi với My.Application.Log, đơn giản bạn cần sử dụng phương thức WriteEntry( ). Bạn cung cấp một thông điệp chuỗi làm thông số đầu tiên và hai thông số nữa. Thông số thứ hai là kiểu sự kiện (event) cho biết thông báo có thay thế cho thông tin hay không, một cảnh báo, một lỗi và...Thông số thứ ba là một đối tượng ngoại lệ, các chi tiết của nó cũng sẽ được sao chép vào trong mục theo dõi.

Ghi chú

Khi một điều nào đó tồi tệ xảy ra trong ứng dụng của bạn, bạn muốn có một cách dễ dàng để ghi nó vào một file hay sổ theo dõi sự kiện. Đừng xem nhiều hơn nữa đối tượng My.Application.Log.

Để tiến hành kiểm tra điều này, tạo và chạy ứng dụng console trong ví dụ 6.1, vốn ghi ghi một chuỗi text ngắn vào sổ theo dõi.

Ví dụ 6.1 Theo dõi đơn giản

```
Module LogTest

    Sub Main( )
      My.Application.Log.WriteEntry("This is a test!", _
        TraceEventType.Information)
    End Sub

End Module
```

Rõ ràng, mã theo dõi rất đơn giản nhưng các mục nhập theo dõi được ghi lại ở đâu? Tất cả đều dựa vào cấu hình trong ứng dụng của bạn. .NET sử dụng các trace listenen là những lớp chuyển dụng lắng nghe các thông báo theo dõi rồi sao chép chúng đến một vị trí khác (chẳng hạn như một file, sổ theo dõi sự kiện, và...). Khi bạn gọi phương thức WriteEntry( ) mục nhập được ghi vào một nhóm bộ nghe hiện hành (được hiển thị thông qua bộ thu thập My.Application.TraceSource). Theo mặc định, những bộ nghe này bao gồm FileLogTraceListener ghi vào một file theo dõi user. File này được lưu trữ dưới một thư mục user cụ thể (được định rõ bởi các biến môi trường APPDATA của user) trong một thư mục con của form [CompanyName]\[ProductName]\[FileVersion], CompanyName, ProductName, và FileVersion ám chỉ thông tin được định rõ trong ứng dụng tập hợp. Ví dụ, nếu Windows user JoeM chạy ứng dụng LogTestApp, file theo dõi sẽ được tạo trong một thư mục chẳng hạn như c:Documents và Settings\JoeM\Application Data\MyCompany\ LogTestApp\1.0.0.0\ LogTestApp.log.

Một khi bạn đã tìm thấy thư mục phù hợp, bạn có thể mở file theo dõi (logfile) trong Notepad để kiểm tra các nội dung text. Bạn sẽ tìm thấy những thông tin sau:

Ghi chú

Để cấu hình thông tin thu thập, hãy nhấp dúp vào hạng mục My Project trong Solution Explorer, chọn tab Application và nhấp nút Assembly Information.

```
Microsoft.VisualBasic.MyServices.Log.WindowsFormsSource    Information

    0    This is a test!
```

Số 0 đại diện cho thông tin kiểu theo dõi. Các mục nhập kế tiếp ghép dữ liệu vào file theo dõi này, dữ liệu không bao giờ bị xóa (trừ khi bạn xóa file bằng thao tác tay).

6.1.2 Theo dõi những vị trí khác như thế nào?

.NET bao gồm một số bộ nghe theo dõi được cài trước mà bạn có thể sử dụng. Chúng bao gồm:

DefaultTraceListener

Bộ nghe này ghi thông tin vào trong phần gỡ rối của cửa sổ trong Visual Studio. Về cơ bản nó hữu ích trong khi đang kiểm tra.

FileLogTraceListener

Bộ nghe này ghi thông tin vào file theo dõi ứng dụng có tên là [AssemblyName].log. Vị trí mặc định của file theo dõi tùy thuộc vào các xác lập môi trường của user và thông tin ứng dụng.

EventLogTraceListener

Bộ nghe này ghi thông tin vào sổ theo dõi sự kiện Windows.

XmlWriteTraceListener

Bộ nghe này ghi thông tin vào một file trong dạng thức XML. Bạn định rõ vị trí file nên được lưu trữ ở đâu. Nếu cần thiết, thư mục sẽ được tạo tự động.

Theo mặc định, mọi ứng dụng Visual Basic mới mà bạn tạo bắt đầu sự hoạt động của nó với hai bộ nghe : một DefaultTraceListnener và một FileLogTraceListener. Để thêm vào các bộ nghe mới, bạn cần sửa đổi file cấu hình ứng dụng. Trong Visual Studio, bạn có thể nhấp đúp hạng mục app.config trong Solution Explorer. Thông tin bộ nghe được định rõ trong hai phần nhỏ của phần <system.diagnostics>.

Trong phần nhỏ <sharedListeners>, bạn định rõ bộ nghe theo dõi mà bạn muốn có tùy chọn để sử dụng, hãy định rõ bất kỳ các thuộc tính cấu hình liên quan nào và gán một tên mô tả. Đây là một ví dụ minh họa việc định rõ một bộ nghe mới để viết dữ liệu XML vào một file theo dõi:

Ghi chú

Hãy nhớ rằng sau khi ứng dụng được cài đặt, file app.config được đổi tên để có tên ứng dụng, thêm vào phần đuôi .config.

```
<sharedListeners>
  <add name="MyXmlLog" type="System.Diagnostics.XmlWriterTraceListener"
    initializeData="c:\MyLog.xml" />
</sharedListeners>
```

Trong phần nhỏ <source>, bạn đinh danh bộ nghe theo dõi mà bạn muốn sử dụng, lựa chọn một danh sách <sharedListeners>:

```
<source name="Microsoft.VisualBasic.MyServices.Log.WindowsFormsSource">
  <listeners>
    <add name="Xml"/>
  </listeners>
</source>
```

Sự tách biệt giữa phần <sharedListeners> và phần <sources> cho phép bạn nhanh chóng chuyển đổi các bộ nghe ngẫu nhiên, mà không cản trở các xác lập cấu hình của chúng.

Bây giờ, bạn có thể chạy lại ứng dụng được trình bày ở ví dụ 6.1. Lúc này nó sẽ ghi thông báo vào một file XML có tên là MyLog.xml trong thư mục gốc C:. Các nội dung như sau (với thông tin biểu đồ được xóa để có khả năng đọc tốt hơn):

```
<E2ETraceEvent>
  <System>
    <EventID>0</EventID>
    <Type>0</Type>

    <TimeCreated SystemTime="2004-07-26T16:14:04.7533392Z" />
    <Source Name="Microsoft.VisualBasic.MyServices.Log.WindowsFormsSource" />
    <Execution ProcessName="LogSample.vshost" ProcessID="3896" ThreadID="8" />
    <Computer>FARIAMAT</Computer>
  </System>
  <ApplicationData>
    <System.Diagnostics>
```

```
        <Message>This is a test!</Message>
        <Severity>Information</Severity>
      </System.Diagnostics>
    </ApplicationData>
  </E2ETraceEvent>
```

Ví dụ 6.2 trình bày một minh họa file cấu hình hoàn chỉnh. Nó cho phép can file, can sổ theo dõi sự kiện và can sổ theo dõi XML. Hãy chú ý rằng EventLogTraceListener được điều chỉnh tốt với một bộ lọc bảo đảm chỉ các thông báo lỗi được theo dõi.

Ví dụ 6.2 Ghi dữ liệu vào ba bộ nghe theo dõi khác nhau

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>

    <!— Enable all three trace listeners
       (from the <sharedListeners> section). —>
    <sources>
      <source name="Microsoft.VisualBasic.MyServices.Log.WindowsFormsSource"
       switchName="DefaultSwitch">
        <listeners>
          <add name="FileLog"/>
          <add name="EventLog"/>
          <add name="Xml"/>
        </listeners>
      </source>
    </sources>
    <switches>
      <add name="DefaultSwitch" value="Information" />
    </switches>

    <!— Define three trace listeners that you might want to use. —>
    <sharedListeners>
      <add name="FileLog" type="System.Diagnostics.FileLogTraceListener"
       initializeData="FileLogWriter" delimiter=";" />
      <add name="EventLog" type="System.Diagnostics.EventLogTraceListener"
       initializeData="MyApplicationLog">
```

```
        <filter type="System.Diagnostics.SeverityFilter" initializeData="Error" />
      </add>
      <add name="Xml" type="System.Diagnostics.XmlWriterTraceListener"
       initializeData="c:\SampleLog.xml" delimiter=";" />
    </sharedListeners>
  </system.diagnostics>
</configuration>
```

Bây giờ, bạn có thể sử dụng cùng ứng dụng đơn giản để ghi cùng lúc các file theo dõi thông thường, file theo dõi XML và một mục nhập trong sổ theo dõi sự kiện Windows có tên là Applications.

Thật không may, không có bất kỳ .NET API mức cao nào để truy xuất thông từ một sổ theo dõi. Nếu thông tin được theo dõi trong một file, bạn có thể sử dụng các lớp FileStream và StreamReader từ namespace System.IO để đọc file một dòng tại một thời điểm. Nếu bạn đã nhập thông tin trong sổ theo dõi sự kiện Windows, bạn sẽ cần dựa vào lớp EventLog, bạn có thể tìm thấy nó trong namespace System.Diagnostics.

Ghi chú

Sổ theo dõi sự kiện (envent log) chỉ là một danh sách các thông báo được lưu trữ bởi hệ điều hành trong một khoảng thời gian cụ thể. Để xem sổ theo dõi sự kiện, hãy chọn Event Viewer từ phần Administrative Tools của Control Panel.

6.2 Ping một máy tính khác

Internet là một mạng động trong đó các máy tính xuất hiện và biến mất mà không cần cảnh báo. Một kiểm tra ứng dụng đơn giản có thể luôn thực hiện để kiểm tra nếu một máy tính có thể tiến đến là gởi một thông điệp đối chuyển. Về mặt kỹ thuật, đối chuyển tương tự như việc hỏi một máy tính khác "bạn có ở đó không?" để nhận được câu trả lời của nó, đối chuyển gởi một kiểu thông báo đặc biệt qua giao thức Internet mức thấp gọi là ICMP (Internet Control Message Protocol).

Ghi chú

Cần tìm ra nếu một máy tính có thể tiến đến qua Internet? Với lớp Ping mới, bạn có thể đưa ra đề nghị đơn giản này mà không làm rối mã xuống mức thấp.

Việc gởi một thông điệp đối chuyến bằng việc sử dụng các lớp tìm thấy trong các namespace System.Net là một sự thách thức và đòi hỏi hàng chục câu mã cấp thấp xử lý các socket thô. Trong .NET 2.0, có một giải pháp đơn giản hơn với lớp Ping mới trong namespace System.Net.NetworkingInformation.

6.2.1 Bạn làm điều đó như thế nào?

Để đối chuyền một máy tính, bạn hãy sử dụng phương thức Ping( ) của đối tượng My.Computer.Network. Phương thức này cho phép bạn truy cập thuận tiện vào tính năng đối chuyền tối thiểu. Phương thức Ping( ) trả về true hay False tùy thuộc vào nó có nhận được một sự hồi đáp từ máy tính bạn đang cố liên lạc hay không.

Ghi chú

Windows bao gồm một tiện ích gọi là ping.exe mà bạn có thể sử dụng để đối chuyền các máy tính khác tại dòng lệnh.

Ví dụ 6.3 sử dụng phương thức này để liên lạc với web server www.yahoo.com.

Ví dụ 6.3 Đối chuyền một máy tính từ xa

```
Module PingTest

    Sub Main( )
        Dim Success As Boolean

        ' Try to contact www.yahoo.com (wait 1000 milliseconds at most,
        ' which is the default if you don't specify a timeout).
        Success = My.Computer.Network.Ping("www.yahoo.com", 1000)
        Console.WriteLine("Did the computer respond? " & Success)
    End Sub

End Module
```

Khi bạn gọi Ping( ), bạn sẽ chỉ định hai thông số: URL hay địa chỉ IP cho máy tính mà bạn đang cố liên lạc (chẳng hạn www.microsoft.com hay 123.4123.4), và tùy chọn khoảng thời gian chờ đợi tối đa tính bằng mili giây. Một khi hạn mức này được tiến đến, lời đề nghị hết giờ và phương thức Ping( ) trả về False để cho biết sự thất bại.

Lưu ý

Một thông điệp đối chuyền là một kiểm tra cấp thấp không nhất thiết phải phù hợp với tính có sẵn của các dịch vụ trên một máy tính cụ thể. Ví dụ, cho dù bạn có thể ping www.yahoo.com, điều đó không có nghĩa là các trang web cơ chế tìm kiếm có sẵn và làm việc chính xác. Tương tự, các web server hay những bức tường lửa thường chối từ các thông điệp đối chuyền đã hạn chế khả năng một người nào đó tấn công sự từ chối dịch vụ bằng việc làm đầy máy tính với hàng triệu yêu cầu giả tạo. Đối với lý do đó, nếu bạn đối chuyền www.microsoft.com, bạn sẽ không nhận được một sự hồi đáp, cho dù bạn có thể vẫn lướt website của chúng bằng việc sử dụng địa chỉ đó.

6.2.2 Lấy nhiều thông tin hơn từ máy tính từ xa như thế nào?

Đối tượng My.Computer.Network không trả về bất kỳ thông tin bổ sung nào về các kết quả của kiểm tra đối chuyền. Ví dụ, bạn không tìm ra để nhận một hồi đáp, mất bao lâu. đó là một số liệu thống kê chủ yếu được sử dụng bởi một số ứng dụng, chẳng hạn như phần mềm đồng đẳng, nhằm sắp xếp tốc độ kết nối của các máy tính khác nhau.

Để có thêm thông tin, bạn cần trực tiếp điều khiển lớp Ping trong namespace System.Net.NetworkingInformation. Nó trả về một đối tượng PingResult với một số mẩu thông tin, bao gồm thời gian tận dụng cho một sự hồi đáp. Đoạn mã sau đây đặt phương thức này vào việc kiểm tra. Giả sử rằng bạn đã nhập namespace System.Net. Networking Information:

```
Dim Pinger As New Ping

Dim Reply As PingReply = Pinger.Send("www.yahoo.com")

Console.WriteLine("Time (milliseconds): " & Reply.RoundTripTime)

Console.WriteLine("Exact status: " & Reply.Status.ToString( ))

Console.WriteLine("Adress contacted: " & Reply.Address.ToString( ))
```

Đây là một số kết xuất mẫu:

```
Time (milliseconds): 61

Exact status: Success

Adress contacted: 216.109.118.78
```

Lớp Ping cũng cung cấp một phương thức SendAsync( ) mà bạn có thể sử dụng để đối chuyển một máy tính không trì hoãn mã của bạn (bạn có thể xử lý sự hồi đáp ở một chuỗi khác khi một chỉ lệnh gọi lại phát sinh) và những phiên bản quá tải khác của phương thức Send( ) cho phép bạn xác lập các tùy chọn cấp thấp (giống như số chặn mà thông điệp đối chuyển sẽ truyền qua trước khi hết hạn).

6.3 Lấy thông tin về kết nối mạng

Một số ứng dụng cần điều chỉnh cách chúng làm việc như thế nào dựa trên một kết nối mạng có được hiển thị hay không. Ví dụ, hãy tưởng tượng một công cụ báo cáo kinh doanh chạy trên máy tính xách tay của một giám đốc kinh doanh đang đi du lịch. Khi máy tính xách tay được cắm vào mạng, ứng dụng cần chạy trong một chế độ kết nối để truy xuất thông tin nó cần đến, chẳng hạn như một danh sách về các sản phẩm, trực tiếp từ một cơ sở dữ liệu hay dịch vụ Web. Khi máy tính xách tay ngắt kết nối ra khỏi mạng, ứng dụng cần làm giảm một chế độ ngắt kết nối vô hiệu hóa các tính năng nào đó hay tụt lại trên một dữ liệu hơi cũ hơn được lưu trữ trong một file cục bộ. Để quyết định về chế độ nào được sử dụng, một ứng dụng cần một phương pháp nhanh để xác định trạng thái của máy tính hiện tại. Nhờ vào đối tượng My.Computer.Network mới, tác vụ này trở nên dễ dàng.

Ghi chú

Bạn cần tìm xem máy tính của bạn hiện tại có đang online (trực tuyến) hay không? Với My Class, việc kiểm tra này chỉ là một thuộc tính đơn giản.

6.3.1 Bạn làm điều đó như thế nào?

Đối tượng My.Computer.Network cung cấp một thuộc tính IsAvailable đơn giản cho phép bạn xác định liệu máy tính hiện tại có kết nối mạng hay không. Thuộc tính IsAvailable trả về TRue miễn là ít nhất một trong những giao diện mạng cấu hình được kết nối và nó phục vụ như một kiểm tra nhanh để xem máy tính có trực tuyến hay không. Để tiến hành kiểm tra điều này, hãy nhập mã sau trong một ứng dụng console:

```
If My.Computer.Network.IsAvailable Then

    Console.WriteLine("You have a network interface.")

End If
```

Nếu bạn muốn biết thêm thông tin, bạn có trở lại các namespace System.Net và System.Net.NetworkInformation, các namespace này cung

cấp nhiều chi tiết làm nổi bật tốt hơn. Ví dụ, để truy xuất và hiển thị địa chỉ IP đối với máy tính hiện tại, bạn có thể sử dụng lớp System.Net.Dns bằng cách gõ nhập mã này:

```
' Retrieve the computer name.
Dim HostName As String = System.Net.Dns.GetHostName( )
Console.WriteLine("Host name: " & HostName)

' Get the IP address for this computer.
' Note that this code actually retrieves the first
' IP address in the list, because it assumes the
' computer only has one assigned IP address
' (which is the norm).
Console.WriteLine("IP: " & _
   System.Net.Dns.GetHostByName(HostName).AddressList(0).ToString( ))
```

Đây là kết xuất bạn có thể thấy:

```
Host name: FARIAMAT

IP: 192.168.0.197
```

Ngoài ra, bây giờ bạn có thể truy xuất ngay cả các thông tin chi tiết hơn về kết nối mạng của bạn không có sẵn trong những phiên bản trước của .NET. Để làm điều đó, bạn cần sử dụng lớp System.Net.Network Information.IPGlobalProperties mới tiêu biểu cho hoạt động mạng trên một mạng IP chuẩn.

Lớp IPGlobalProperties cung cấp một số phương thức cho phép bạn truy xuất các đối tượng khác nhau, mỗi đối tượng cung cấp các số liệu thống kê đối với một kiểu hoạt động mạng cụ thể. Ví dụ, nếu bạn quan tâm đến mọi sự lưu thông qua kết nối mạng của bạn, bạn sử dụng TCP, bạn có thể gọi IPGlobalProperties.GetTcpIv4Statistics( ). Đối với đa số người, đây là thước đo mạng hữu ích nhất. Mặt khác, nếu bạn đang sử dụng một mạng IPv6 thế hệ kế tiếp, bạn cần sử dụng IPGlobalProperties.GetTcpIPv6Statistics( ). Những phương thức khác tồn tại để giám sát sự lưu thông sử dụng các giao thức UPD hay ICMP. Rõ ràng là bạn sẽ cần biết một phần nhỏ về nối mạng để thoát khỏi những phương thức này.

Thủ thuật

IP (Internet Protocol) là khối kiến tạo chính của đa số các mạng và Internet. Nó nhận dạng duy nhất các máy tính với một địa chỉ IP gồm bốn phần và

cho phép bạn gởi một gói tin cơ bản từ một máy tính này sang một máy tính khác (không có bất kỳ điểm nào như sửa lỗi, điều khiển lưu trình hay quản lý kết nối). Nhiều giao thức mạng khác, chẳng hạn như TCP (Transmission Connection Protocol) được cài đặt ở phần đầu nền tảng chính IP và các giao thức khác vẫn được cài đặt ở phần đầu của TCP (chẳng hạn HTTP, ngôn ngữ của Web).

Mã sau đây truy xuất các số liệu thống kê chi tiết về lưu thông mạng. Giả sử rằng bạn đã nhập namespace System.Net.Network Information:

```
Dim Properties As IPGlobalProperties = IPGlobalProperties.GetIPGlobalProperties( )
Dim TcpStat As TcpStatistics
TcpStat = Properties.GetTcpIPv4Statistics( )

Console.WriteLine("TCP/IPv4 Statistics:")
Console.WriteLine("Minimum Transmission Timeout... : " & _
  TcpStat.MinimumTransmissionTimeOut)
Console.WriteLine("Maximum Transmission Timeout... : " & _
  TcpStat.MaximumTransmissionTimeOut)

Console.WriteLine("Connection Data:")
Console.WriteLine("  Current ..... : " & _
  TcpStat.CurrentConnections)
Console.WriteLine("  Cumulative ..... : " & _
  TcpStat.CumulativeConnections)
Console.WriteLine("  Initiated ..... : " & _
  TcpStat.ConnectionsInitiated)
Console.WriteLine("  Accepted ..... : " & _
  TcpStat.ConnectionsAccepted)
Console.WriteLine("  Failed Attempts ..... : " & _
  TcpStat.FailedConnectionAttempts)
Console.WriteLine("  Reset ..... : " & _
  TcpStat.ResetConnections)

Console.WriteLine( )
Console.WriteLine("Segment Data:")
Console.WriteLine("  Received ..... : " & _
```

```
    TcpStat.SegmentsReceived)
Console.WriteLine("  Sent ..... : " & _
    TcpStat.SegmentsSent)
Console.WriteLine("  Retransmitted ..... : " & _
    TcpStat.SegmentsResent)
```

Đây là kết xuất bạn có thể thấy:

```
TCP/IPv4 Statistics:
Minimum Transmission Timeout... : 300
Maximum Transmission Timeout... : 120000
Connection Data:
  Current ..... : 6
  Cumulative ..... : 29
  Initiated ..... : 10822
```

Ghi chú

Các số liệu thống kê được lưu giữ kể từ lúc thiết lập nối kết. Điều đó có nghĩa là mỗi khi bạn ngắt kết nối hay khởi động lại máy tính, bạn phải xác lập lại các số liệu thống kê nối mạng.

```
  Accepted ..... : 41
  Failed Attempts ..... : 187
  Reset ..... : 2271

Segment Data:
  Received ..... : 334791
  Sent ..... : 263171
  Retransmitted ..... : 617
```

6.3.2 Thế còn...

..... những vấn đề kết nối khác chẳng hạn như một bộ định tuyến ngắt kết nối, mạng hay thay đổi hay một bức tường lửa hạn chế truy cập đến vị trí khác mà bạn cần như thế nào?

Các số liệu thống kê kết nối mạng không cho bạn bất kỳ thông tin nào về phần còn lại của mạng (mặc dù bạn có thể cố đối chuyển một máy tính ở một nơi nào khác trên mạng, như đã được mô tả ở phần trước). Nói cách khác, ngay cả khi một kết nối mạng có sẵn, không có cách nào bảo

đảm nó đang làm việc. Vì lý do đó, bất cứ khi nào bạn cần truy cập một tài nguyên qua mạng, dù đó là một dịch vụ web, cơ sở dữ liệu hay ứng dụng chạy trên một máy tính khác, bạn cần gói chỉ lệnh của bạn trong một mã xử lý ngoại lệ chính xác.

6.4 Tải lên và tải xuống các File với FTP

Những phiên bản trước của .NET không bao gồm bất kỳ công cụ nào cho FTP (File Transfer Protocol), một giao thức phổ biến được sử dụng để chuyển các file đến và đi từ một web server. Kết quả là bạn hoặc phải mua một thành phần của nhóm thứ ba hay ghi riêng cho mình (theo nguyên tắc thì dễ dàng nhưng trong thực tế khó thực hiện chính xác).

Ghi chú

Bạn cần tải lên các file đến một site FTP hay tải xuống nội dung đang hiện hữu? Phần hỗ trợ mới có sẵn trong VB 2005.

Trong .NET 2.0, một lớp FtpWebRequest mới lấp đầy khoảng hở. Tuy nhiên, lớp FtpWebRequest có những tính phức tạp riêng của nó, vì thế các lập trình viên Microsoft đơn giản hóa hoạt động cho các nhà phát triển VB thậm chí nhiều hơn bằng việc mở rộng đối tượng My.Computer.Network để cung cấp hai phương thức truy cập nhanh cho các tác vụ FTP cơ bản hoàn tất. Có UploadFile( ) gởi một file đến một server từ xa và DownloadFile( ) truy xuất một file và lưu trữ nó cục bộ.

6.4.1 Bạn làm điều đó như thế nào?

Dù bạn sử dụng lớp FtpWebRequest hay đối tượng My.Computer.Network, toàn bộ tương tác FTP trong .NET không được ấn định. Điều đó có nghĩa là bạn kết nối với site FTP, thực hiện một tác vụ đơn giản (giống như việc chuyển một file hay truy xuất một thư mục liệt kê) rồi ngắt kết nối. Nếu bạn cần thực hiện một tác vụ khác, bạn cần kết nối lại. Thật may mắn, tiến trình kết nối và đăng nhập này được xử lý tự động bởi .NET Framework.

Cách dễ nhất để sử dụng FTP trong một ứng dụng VB là thực hiện điều này thông qua đối tượng My.Computer.Network. Nếu bạn sử dụng các phương thức FTP của nó, bạn không bao giờ cần phải lo lắng về các chi tiết gây nhàm chán trong việc mở, đóng và đọc các stream. Để tải xuống một file, thông tin tối thiểu mà bạn cần là URL hướng đến site FTP và đường dẫn hướng đến file cục bộ. Đây là một ví dụ minh họa:

```
My.Computer.Network.DownloadFile( _

    "ftp://ftp.funet.fi/pub/gnu/prep/gtk.README", "c:\readme.txt")
```

Lệnh này truy xuất file trên site FTP ftp.funet.fi trên đường dẫn /pub/gnu/prep/gtk.README và sao chép nó đến file cục bộ c:\readme.txt.

Việc tải lên sử dụng các thông số giống nhau, nhưng đảo ngược.

```
My.Computer.Network.UploadFile("c:\newfile.txt", _
"ftp://ftp.funet.fi/pub/newfile.txt")
```

Lệnh này sao chép file cục bộ newfile.txt từ thư mục c: \ đến site FTP ftp.funet.fi, trong thư mục từ xa \pub.

Cả DownloadFile( ) và UploadFile( ) hỗ trợ một số quá tải sử dụng các thông số bổ sung, bao gồm các điều kiện (thông tin user và mật khẩu mà bạn có thể cần để đăng nhập vào một server) và một thông số hạn mức thời gian để xác lập lượng thời gian tối đa bạn sẽ chờ đợi một sự hồi đáp trước khi từ bỏ (theo mặc định là một ngàn mili giây).

Thật không không may, các phương thức DownloadFile( ) và UploadFile( ) không quá tráng kiện trong các kiến tạo beta của Visual Basic 2005 và các phương thức có thể thất bại. Một tùy chọn làm việc tốt hơn là lớp FtbWebRequest phức tạp hơn. Không chỉ làm cho ứng dụng đáng tin cậy hơn mà nó còn lấp đầy một số khoảng hở nổi bật trong hỗ trợ FTP được cung cấp bởi My.Network.Computer. Bởi vì FtbWebRequest cho phép bạn thực thi bất kỳ lệnh FTP nào, cho nên bạn có thể sử dụng nó để truy xuất các danh sách liệt kê thư mục, lấy thông tin file và nhiều thứ khác nữa.

Ghi chú

Internet Exlorer có trình duyệt FTP được cài sẵn riêng của nó. Chỉ cần gõ một URL hướng đến một FTP site (giống như ftp://ftp.microsoft.com) vào trong thanh địa chỉ IE để duyệt những gì có ở đó. Bạn có thể sử dụng công cụ này để kiểm chứng mã của bạn đang làm việc chính xác.

Để sử dụng FtbWebRequest, bạn cần thực hiện một số bước. Trước tiên, hãy chuyển URL hướng đến site FTP sang phương thức WebRequest.Create( ) chia sẻ:

```
Dim Request As FtpWebRequest
Request = CType(WebRequest.Create("ftp://ftp.microsoft.com/MISC"), _ FtpWebRequest)
```

Phương thức WebRequest.Create( ) kiểm tra URL và trả về kiểu đối tượng WebRequest thích hợp. Bởi vì các FTP URL luôn luôn bắt đầu với chương trình ftp://, cho nên phương thức Create( ) sẽ trả về một đối tượng FtbWebRequest mới.

Một khi bạn đã có FtbWebRequest, bạn cần lựa chọn tác vụ FTP nào bạn muốn thực hiện bằng việc xác lập thuộc tính FtbWebRequest.Method

với text của lệnh FTP. Đây là một ví dụ minh họa cho việc truy xuất thông tin thư mục với lệnh LIST:

```
Request.Method = "LIST"
```

Một khi bạn đã lựa chọn tác vụ FTP mà bạn muốn thực hiện, bước cuối cùng là thực thi lệnh và đọc sự hồi đáp. Phần nhạy bén là sự hồi đáp được trả về cho bạn như một stream của text. Bạn phải có bổn phận di chuyển thông qua khối text này từng dòng với một StreamReader và phân ngữ thông tin.

Ví dụ, mã sau đây đọc qua một danh sách liệt kê thư mục được trả về và hiển thị mỗi dòng ở một cửa sổ Console:

```
Dim Response As FtpWebResponse = CType(Request.GetResponse( ), FtpWebResponse)
Dim ResponseStream As Stream = Response.GetResponseStream( )
Dim Reader As New StreamReader(ResponseStream, System.Text.Encoding.UTF8)

Dim Line As String
Do
    Line = Reader.ReadLine( )
    Console.WriteLine(Line)
Loop Until Line = ""
```

Kết xuất như sau:

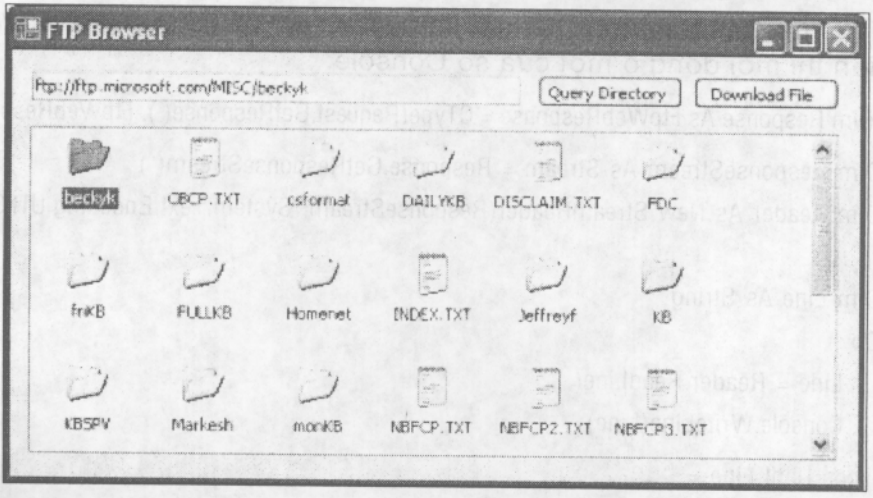
```
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   beckyk
-r-xr-xr-x   1 owner   group         15749    Apr    8     1994   CBCP.TXT
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   csformat
dr-xr-xr-x   1 owner   group         0        Aug    1     2002   DAILYKB
-r-xr-xr-x   1 owner   group         710 A    pr     12    1993   DISCLAIM.TXT
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   FDC
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   friKB
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   FULLKB
dr-xr-xr-x   1 owner   group         0        Jul    3     2002   Homenet
-r-xr-xr-x   1 owner   group         97       Sep    28    1993   INDEX.TXT
...
```

Rõ ràng là nếu bạn muốn thu thập các mẩu thông tin cá nhân (như kích thước file) hay tách biệt các file ra khỏi các thư mục, bạn sẽ cần thực hiện thêm công việc phân ngữ text được trả về bởi StreamReader.

Cuối cùng, khi bạn kết thúc với đề nghị và hồi đáp FTP, bạn cần đóng các stream:

```
Reader.Close( )

Response.Close( )
```

Để đặt tất cả trong một ngữ cảnh, bạn cần xem xét một ứng dụng FTP đơn giản. Hình 6.11 trình bày một ứng dụng mẫu được bao gồm với các mẫu có thể download với chương này.



Hình 6.1 Một ứng dụng trình duyệt FTP đơn giản

Ứng dụng Windows này bao gồm các điều khiển sau:

- Một textbox nơi bạn có thể gõ nhập một URL hướng đến một file hay thư mục trong site FTP.
- Một Button có tên là Query Directory truy xuất các folder và các file tại một URL được cho. Tác vụ này đòi hỏi lớp FtbWebRequest.
- Một Button có tên là Download File tải xuống file tại một URL được cho. Tác vụ này sử dụng phương thức My.Computer.Network.DownloadFile().
- Một FolderBrowserDialog cho phép bạn lựa chọn một folder mà file được tải xuống sẽ được lưu.
- Một ListView hiển thị danh sách liệt kê file và thư mục cho URL. Danh sách này được làm mới mỗi khi bạn nhấp vào nút Query Directory. Ngoài ra, mỗi khi bạn nhấp chọn một hạng mục trong ListView, thông tin đó tự động được thêm vào URL trong text box (hộp văn bản). Điều này cho phép bạn duyệt

nhanh qua một site FTP, lùi xuống một số tầng trong cấu trúc thư mục và lựa chọn file gây chú ý cho bạn.

Ví dụ 6.4 chỉ ra mã cho form trình duyệt FTP.

Ví dụ 6.4 Form trình duyệt FTP

```
Public Class FtpForm
    Inherits System.Windows.Forms.Form

    ' Stores the path currently shown in the ListView.
    Private CurrentPath As String

    Private Sub cmdQuery_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles cmdQuery.Click
        ' Check the URI is valid.
        Dim RequestUri As Uri = ValidateUri(txtFtpSite.Text)
        If RequestUri Is Nothing Then Return

        ' Clear the ListView.
        listDir.Items.Clear( )

        ' Create a new FTP request using the URI.
        Dim Request As FtpWebRequest
        Request = CType(WebRequest.Create(RequestUri), FtpWebRequest)

        ' Use this request for getting full directory details.
        Request.Method = "LIST"
        Request.UsePassive = False

        Dim Response As FtpWebResponse
        Dim ResponseStream As Stream
        Dim Reader As StreamReader
        Try
            ' Execute the command and get the response.
            Response = CType(Request.GetResponse( ), FtpWebResponse)
            Debug.WriteLine("Status: " & Response.StatusDescription)

            ' Read the response one line at a time.
```

```
        ResponseStream = Response.GetResponseStream( )
        Reader = New StreamReader(ResponseStream, System.Text.Encoding.UTF8)
        Dim Line As String
        Do
            Line = Reader.ReadLine( )
            If Line <> "" Then
                Debug.WriteLine(Line)

                ' Extract just the file or directory name from the line.
                Dim ListItem As New ListViewItem(Line.Substring(59).Trim( ))
                If Line.Substring(0, 1) = "d" Then
                    ListItem.ImageKey = "Folder"
                Else
                    ListItem.ImageKey = "File"
                End If
                listDir.Items.Add(ListItem)
            End If
        Loop Until Line = ""

        ' Operation completed successfully. Store the current FTP path.
        CurrentPath = RequestUri.ToString( )

    Catch Ex As Exception
        MessageBox.Show(Ex.Message)

    Finally
        ' Clean up.
        Reader.Close( )
        Response.Close( )

    End Try
End Sub

Private Sub cmdDownload_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles cmdDownload.Click
```

```
' Check the URI is valid.
Dim RequestUri As Uri = ValidateUri(txtFtpSite.Text)
If RequestUri Is Nothing Then Return

' Prompt the user to choose a destination folder.
' Default the file name to the same file name used on the FTP server.
dlgSave.FileName = Path.GetFileName(txtFtpSite.Text)
If dlgSave.ShowDialog( ) <> Windows.Forms.DialogResult.OK Then
    Return
End If

' Create a new FTP request using the URI.
Dim Request As FtpWebRequest
Request = CType(WebRequest.Create(RequestUri), FtpWebRequest)

' Use this request for downloading the file.
Request.UsePassive = False
Request.Method = "RETR"

Dim Response As FtpWebResponse
Dim ResponseStream, TargetStream As Stream
Dim Reader As StreamReader
Dim Writer As StreamWriter
Try
    ' Execute the command and get the response.
    Response = CType(Request.GetResponse( ), FtpWebResponse)
    Debug.WriteLine("Status: " & Response.StatusDescription)
    Debug.WriteLine("File Size: " & Response.ContentLength)

    ' Create the destination file.
    TargetStream = New FileStream(dlgSave.FileName, FileMode.Create)
    Writer = New StreamWriter(TargetStream)

    ' Write the response to the file.
    ResponseStream = Response.GetResponseStream( )
    Reader = New StreamReader(ResponseStream, System.Text.Encoding.UTF8)
```

```
            Writer.Write(Reader.ReadToEnd( ))

        Catch Err As Exception
            MessageBox.Show(Err.Message)

        Finally
            ' Clean up.
            Reader.Close( )
            Response.Close( )
            Writer.Close( )
        End Try
    End If

End Sub

Private Function ValidateUri(ByVal uriText As String) As Uri
    Dim RequestUri As Uri
    Try
        ' Check that the string is interpretable as a URI.
        RequestUri = New Uri(uriText)

        ' Check that the URI starts with "ftp://"
        If RequestUri.Scheme <> Uri.UriSchemeFtp Then
            RequestUri = Nothing
        End If
    Catch
        RequestUri = Nothing
    End Try

    If RequestUri Is Nothing Then
        MessageBox.Show("Invalid Uri.")
    Else

    End If
    Return RequestUri
End Function
```

```
    Private Sub listDir_SelectedIndexChanged(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles listDir.SelectedIndexChanged
        ' When a new item is selected in the list, add this
        ' to the URI in the text box.
        If listDir.SelectedItems.Count <> 0 Then
            CurrentPath = CurrentPath.TrimEnd("/")
            txtFtpSite.Text = CurrentPath & "/" & listDir.SelectedItems(0).Text
        End If
    End Sub


End Class
```

Mã phức tạp nhất được tìm thấy ở ví dụ này xảy ra trong trình xử lý sự kiện trên nút cmdQuery, nút này truy xuất một danh sách liệt kê thư mục, phân ngữ ra thông tin quan trọng và cập nhật ListView.

6.5 Kiểm tra thành phần nhóm của User hiện hành

.NET Framework luôn luôn cung cấp các lớp bảo mật cho phép bạn truy xuất thông tin cơ bản về account của user hiện hành. Đối tượng My.User mới được cung cấp bởi Visual Basic 2005 tạo thuận lợi hơn các phiên bản trước đây nhằm truy cập thông tin này.

Ghi chú

Tìm ra ai đang sử dụng ứng dụng của bạn và các nhóm mà một user bí ẩn thuộc về.

6.5.1 Bạn làm điều đó như thế nào?

Các ứng dụng thường cần kiểm tra ai đang chạy ứng dụng. Ví dụ, có lẽ bạn muốn hạn chế một số tính năng đối với các nhóm nào đó, chẳng hạn như các nhà quản lý Windows. Bạn có thể hoàn thành điều này với đối tượng My.User.

Đối tượng My. User cung cấp hai thuộc tính chính vốn trả về thông tin về user hiện hành. Đó là:

IsAuthenticated

Trả về true nếu thông tin user account hiện hành có sẵn trong đối tượng My.User. Lý do duy nhất mà thông tin này sẽ không hiển thị là nếu bạn đã tạo một ứng dụng web cho phép truy cập giấu

tên, hay nếu Windows account hiện hành không được kết hợp với miền ứng dụng.

Username

Trả về tên user hiện hành. Giả sử bạn đang sử dụng một chính sách bảo mật Windows, đây là tên Windows account cho user, trong form ComputerName\UserName hay DomainName\UserName.

Đối tượng My.User cũng cung cấp một phương thức đơn, IsInRole( ). Phương thức này chấp nhận tên của một nhóm (chẳng hạn một chuỗi) rồi trả về true nếu user thuộc về nhóm đó. Ví dụ, bạn có thể sử dụng kỹ thuật này để xác minh user hiện hành là một nhà quản lý Windows trước khi thực hiện một tác vụ nào đó.

Để tiến hành kiểm tra điều này, hãy sử dụng ứng dụng console account ở ví dụ 6.5, ví dụ này trình bày một số thông tin cơ bản về user hiện hành và kiểm tra nếu user là một Administrator.

Ghi chú

Để kiểm tra danh sách nhóm và user cho máy tính hiện hành (hay thay đổi), hãy chọn Computer management từ phần Administrative Tools của Control Panel. Sau đó, mở rộng nút System Tools > Local User and Groups.

Ví dụ 6.5 Kiểm tra sự nhận dạng user hiện hành

```
Module SecurityTest

  Sub Main( )
    ' Use Windows security. As a result, the User object will
    ' provide the information for the currently logged in user
    ' who is running the application.
    My.User.InitializeWithWindowsUser( )

    Console.WriteLine("Authenticated: " & My.User.Identity.IsAuthenticated)
    Console.WriteLine("User: " & My.User.Identity.Username)

    Console.WriteLine("Administrator: " & My.User.IsInRole("Administrators"))
  End Sub
End Module
```

Đây là loại kết xuất bạn sẽ thấy khi chạy việc kiểm tra này:

```
Authenticated: True

User: FARIAMAT\Matthew

Administrator: True
```

6.6 Tạo mật mã thông tin bí mật cho User hiện hành

Các ứng dụng thường cần một phương pháp lưu trữ dữ liệu riêng tư trong profile hay trong bộ nhớ. Giải pháp rõ ràng là tạo mật mã đối xứng, mã hoá dữ liệu của bạn bằng việc sử dụng một loạt byte ngẫu nhiên gọi là một khóa bí mật. Vấn đề là khi bạn muốn giải mật mã dữ liệu chống nghe trộm của bạn, bạn cần sử dụng cùng khóa bí mật mà bạn đã sử dụng để tạo mật mã. Điều này làm phát sinh những vấn đề phức tạp nghiêm trọng. Hoặc bạn cần tìm một nơi an toàn để bảo vệ khóa bí mật của bạn hay bạn cần phải sinh khóa bí mật từ một số thông tin khác, chẳng hạn một mật khẩu do user cung cấp (không an toàn hơn nhiều và có thể làm hỏng toàn bộ khi các user quên mật khẩu của họ).

Ghi chú

Bạn cần một cách nhanh để tạo mật mã thông tin bí mật, không cần phải lo lắng về quản lý khóa? Giải pháp mà từ lâu bạn chờ đợi đã xuất hiện trong.NET 2.0 với lớp ProtectedData.

Giải pháp lý tưởng là yêu cầu hệ điều hành Windows tạo mật mã dữ liệu cho bạn. Để hoàn thành điều này, bạn cần DPAPI (Data Protection API) tạo mật mã dữ liệu sử dụng một khóa đối xứng được dựa trên một mẩu thông tin user cụ thể hay máy tính cụ thể. Theo cách này, bạn không cần phải lo lắng về việc lưu trữ khóa hay sự ủy quyền. Thay vào đó, hệ điều hành xác thực user khi người này đăng nhập. Dữ liệu được lưu trữ bởi một user không thể tự động truy cập được đến các user khác.

Ở những phiên bản trước của .NET, không có lớp quản lý nào cho việc sử dụng DPAPI. Lỗi sơ suất này được chỉnh sửa trong .NET 2.0 với lớp ProtectedData mới trong namespace System.Security.Cryptography.

6.6.1 Bạn làm điều đó như thế nào?

Lớp ProtectedData cung cấp hai phương thức chia sẻ. ProtectData() sử dụng một mảng byte với dữ liệu nguồn và trả về một mảng byte với dữ liệu được tạo mật mã. UnProtectData( ) thực hiện tác vụ ngược lại, sử

dụng một mảng byte được tạo mật mã và trả về một mảng byte với dữ liệu đã giải mật mã.

Thủ thuật duy nhất cho việc sử dụng các phương thức ProtectData( ) và UnprotectData( ) là bạn có thể chỉ tạo mật mã và giải mật mã dữ liệu trong một mảng byte. Điều đó có nghĩa là nếu bạn muốn tạo mật mã chuỗi, các con số hay điều gì khác, bạn chỉ cần ghi nó vào một mảng byte trước khi bạn thực hiện việc tạo mật mã. Để xem tác vụ này thực hiện như thế nào, bạn có thể chạy mã ứng dụng console trong ví dụ 6.6.

Ví dụ 6.6 Lưu trữ một chuỗi text đã tạo mật mã trong một file.

```
Imports System.Security.Cryptography
Imports System.IO

Module ProctedData

    Sub Main( )
        ' Get the data.
        Console.WriteLine("Enter a secret message and press enter.")
        Console.Write(">")
        Dim Input As String = Console.ReadLine( )

        Dim DataStream As MemoryStream
        If Input <> "" Then
            Dim Data( ), EncodedData( ) As Byte

            ' Write the data to a new MemoryStream.
            DataStream = New MemoryStream( )
            Dim Writer As New StreamWriter(DataStream)
            Writer.Write(Input)
            Writer.Close( )

            ' Convert the MemoryStream into a byte array,
            ' which is what you need to use the ProtectData( ) method.
            Data = DataStream.ToArray( )

            ' Encrypt the byte array.
            EncodedData = ProtectedData.Protect(Data, Nothing, _
```

```
            DataProtectionScope.CurrentUser)

        ' Store the encrypted data in a file.
        My.Computer.FileSystem.WriteAllBytes("c:\secret.bin",
         EncodedData, False)
    End If
End Sub

End Module
```

Khi bạn chạy ứng dụng này, bạn sẽ được nhắc nhở để gõ vào một số text, ứng dụng này sẽ tạo mật mã bằng việc sử dụng thông tin user account hiện hành của bạn và lưu trữ trong file secret.bin. Dữ liệu sẽ không thể truy cập được đến bất kỳ user nào khác.

Để kiểm chứng dữ liệu được tạo mật mã, bạn có hai lựa chọn. Bạn có thể mở file và tự tìm kiếm, hay bạn có thể sửa đổi mã để nó đọc dữ liệu trực tiếp từ stream bộ nhớ được tạo mật mã. Mã này thử tác vụ vừa thực hiện và hiển thị một chuỗi của đoạn vô nghĩa như một kết quả:

```
' Verify the data is encrypted by reading and displaying it
' without performing any decryption.
DataStream = New MemoryStream(EncodedData)
Dim Reader As New StreamReader(DataStream)
Console.WriteLine("Encrypted data: " & Reader.ReadToEnd( ))
Reader.Close( )
```

Để giải mật mã dữ liệu, bạn cần đặt nó vào trong một mảng byte rồi sử dụng phương thức UnprotectData( ). Để trích dẫn dữ liệu ra khỏi mảng byte không được tạo mật mã, bạn có thể sử dụng StreamReader. Để bổ sung phần hỗ trợ giải mật mã cho ví dụ trước, hãy chèn mã sau đây, mở file và hiển thị thông điệp bí mật bạn đã gõ vào trước đó:

```
If My.Computer.FileSystem.FileExists("c:\secret.bin") Then
    Dim Data( ), EncodedData( ) As Byte

    EncodedData = My.Computer.FileSystem.ReadAllBytes("c:\secret.bin")
    Data = ProtectedData.Unprotect(EncodedData, Nothing, _
     DataProtectionScope.CurrentUser)

    Dim DataStream As New MemoryStream(Data)
```

```
    Dim Reader As New StreamReader(DataStream)

    Console.WriteLine("Decoded data from file: " & Reader.ReadToEnd( ))
    Reader.Close( )
End If
```

Hãy nhớ rằng bởi vì dữ liệu được tạo mật mã sử dụng user profile hiện hành, bạn có thể giải mật mã dữ liệu bất cứ lúc nào. Sự hạn chế duy nhất là bạn cần phải được đăng nhập vào bên dưới cùng user account.

Chú ý rằng khi bạn bảo vệ dữ liệu, bạn phải lựa chọn một trong những trị từ bảng liệt kê DataProtectionScope. Có hai lựa chọn:

Ghi chú

Bất kể bạn lựa chọn DataProtectionScope nào, thông tin được tạo mật mã sẽ được lưu trữ trong một vùng được bảo vệ đặc biệt của thanh ghi Windows.

LocalMachine

Windows sẽ tạo mật mã dữ liệu với một khóa máy tính cụ thể, bảo đảm không có ai có thể đọc dữ liệu trừ khi họ đăng nhập vào cùng máy tính. Điều này hoạt động tốt đối với các ứng dụng phía server chạy không có sự can thiệp nào của user, chẳng hạn như các dịch vụ Windows và các dịch vụ web.

CurrentUser

Windows sẽ tạo mật mã dữ liệu với khóa user cụ thể để nó không thể truy cập được bất kỳ user nào khác.

Ở ví dụ hiện tại, dữ liệu user cụ thể được lưu trữ. Tuy nhiên, bạn có thể sửa đổi DataProtectionScope để lưu trữ dữ liệu có thể truy cập được đến bất kỳ user nào trên máy tính hiện hành.

6.6.2 Bảo vệ dữ liệu trước khi bạn đặt nó trong một cơ sở dữ liệu như thế nào?

Một khi bạn sử dụng lớp ProtectedData để tạo mật mã dữ liệu của bạn, bạn có thể đặt nó ở bất cứ nơi nào bạn muốn. Ví dụ trước ghi dữ liệu được tạo mã tại một file, nhưng bạn có thể ghi dữ liệu nhị phân vào một bản ghi cơ sở dữ liệu. Để làm như thế, đơn giản là bạn chỉ cần một trường nhị phân trong bảng với đủ chỗ để phù hợp với mảng byte được tạo mật mã. Trong SQL Server, bạn sử dụng kiểu dữ liệu varbinary.

6.7 Mở rộng Console

.NET 1.0 đã giới thiệu lớp Console nhằm cung cấp cho các lập trình viên một cách thuận tiện để tạo các ứng dụng dòng lệnh đơn giản. Phiên bản đầu tiên của Console khác khá thô sơ, với ít phương thức cơ bản hơn chẳng hạn Write( ), WriteLine( ), Read( ), và ReadLine( ). Trong .NET 2.0, các tính năng mới được bổ sung, cho phép bạn xóa cửa sổ, thay đổi các màu sắc tiền cảnh và nền, thay đổi kích thước của cửa sổ và xử lý các phím đặc biệt.

Ghi chú

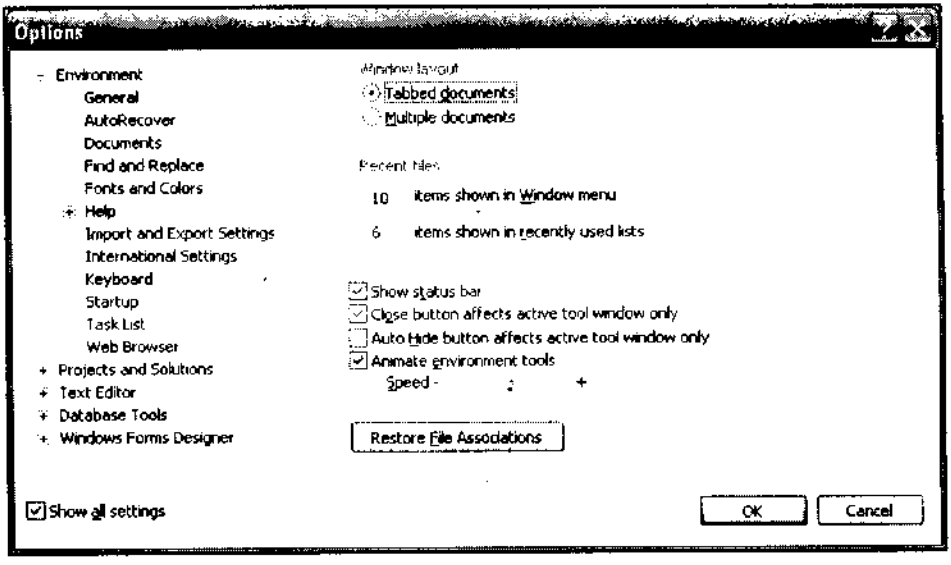
Tối thiểu một lớp Console có các tính năng xử lý bàn phím và ghi màn hình.

6.7.1 Bạn làm điều đó như thế nào?

Cách tốt nhất để biết các tính năng mới là xem chúng hoạt động. Ví dụ 6.7 minh họa một ứng dụng đơn giản, ConsoleText, cho phép user di chuyển một ký tự happy face (khuôn mặt hạnh phúc) chung quanh một cửa sổ console, để lại một dấu vết trong lòng của nó. Ứng dụng chậm mỗi việc nhấn phím, kiểm tra một phím mũi tên có được nhóm hay không và bảo đảm rằng user không di chuyển bên ngoài các giới hạn của cửa sổ.

Lưu ý

Để các tính năng console cấp cao hoạt động, bạn phải vô hiệu hóa cửa sổ Quick Console. Quick Console là một cửa sổ console xuất hiện trong môi trường thiết kế và nó quá nhẹ để hỗ trợ các tính năng như đọc các phím, xác lập các màu sắc và sao chép các ký tự. Để vô hiệu hóa nó, hãy chọn Tools > Options.



Bảo đảm "show all settings checkbox" được đánh dấu kiểm và chọn tab Debugging > General. Sau đó, tắt "Redirect all console output to the Quick Console window".

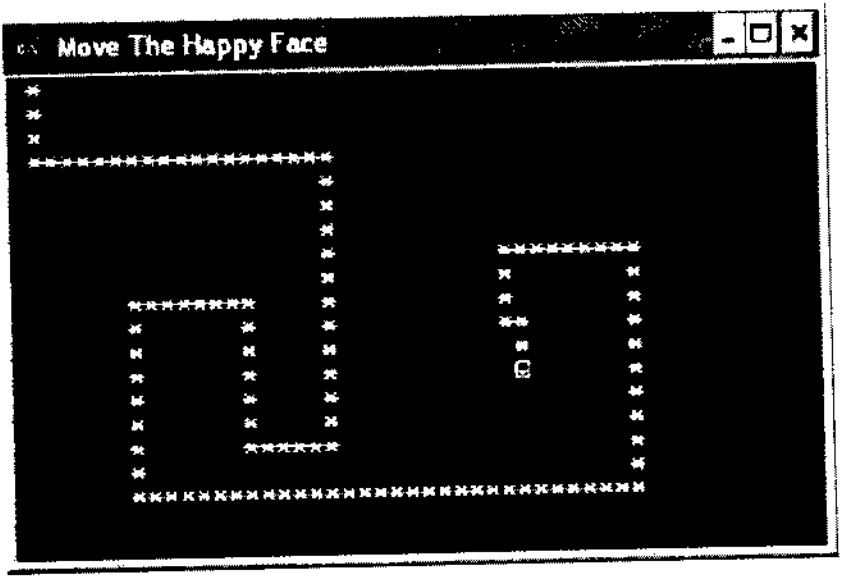
Ví dụ 6.7 Xử lý bàn phím cấp cao với console

```
Module ConsoleTest

    Private NewX, NewY, X, Y As Integer
    Private BadGuyX, BadGuyY As Integer

    Public Sub Main( )
        ' Create a 50 column x 20 line window.
        Console.SetWindowSize(50, 20)
        Console.SetBufferSize(50, 20)

        ' Set up the window.
        Console.Title = "Move The Happy Face"
        Console.CursorVisible = False
        Console.BackgroundColor = ConsoleColor.DarkBlue
        Console.Clear( )

        ' Display the happy face icon.
```

```
    Console.ForegroundColor = ConsoleColor.Yellow
    Console.SetCursorPosition(X, Y)      Console.Write("*")

    ' Read key presses.
    Dim KeyPress As ConsoleKey
    Do
        KeyPress = Console.ReadKey( ).Key

        ' If it's an arrow key, set the requested position.
        Select Case KeyPress
            Case ConsoleKey.LeftArrow
                NewX -= 1
            Case ConsoleKey.RightArrow
                NewX += 1
            Case ConsoleKey.UpArrow
                NewY -= 1
            Case ConsoleKey.DownArrow
                NewY += 1
        End Select

        MoveToPosition( )
    Loop While KeyPress <> ConsoleKey.Escape

    ' Return to normal.
    Console.ResetColor( )
    Console.Clear( )
End Sub

Private Sub MoveToPosition( )
    ' Check for an attempt to move off the screen.
    If NewX = Console.WindowWidth Or NewX < 0 Or _
      NewY = Console.WindowHeight Or NewY < 0 Then
        ' Reset the position.
        NewY = Y
        NewX = X
        Console.Beep( )
```

```
        Else
            ' Repaint the happy face in the new position.
            Console.MoveBufferArea(X, Y, 1, 1, NewX, NewY)

            ' Draw the trail.
            Console.SetCursorPosition(X, Y)
            Console.Write("*")

            ' Update the position.
            X = NewX
            Y = NewY
            Console.SetCursorPosition(0, 0)
        End If
    End Sub

End Module
```

Để tiến hành kiểm tra điều này, hãy chạy ứng dụng và sử dụng các phím mũi tên để di chuyển. Hình 6.2 cho thấy kết xuất của một phiên làm việc ConsoleTest điển hình.



Hình 6.2 Một ứng dụng console thú vị.

Một số phương thức Console mới được sử dụng trong ConsoleTest bao gồm như sau:

Clear( )

Xóa mọi thứ trong cửa sổ console và định vị con trỏ ở góc trái trên cùng.

SetCursorPosition( )

Di chuyển con trỏ đến các tọa độ x và y được chỉ định (được đo từ góc trái trên cùng). Một khi bạn đã di chuyển đến một vị trí mới, bạn có thể sử dụng Console.Write( ) để hiển thị một số ký tự ở đó.

SetWindowSize( ) và SetBufferSize( )

Cho phép bạn thay đổi kích thước của cửa sổ (vùng hiện hành của console) và vùng đệm (các vùng có thể cuộn của console, bằng hay lớn hơn kích thước cửa sổ).

ResetColor( )

Xác lập lại các màu của tiền cảnh và nền theo các mặc định của nó.

Beep( )

Phát ra một tiếng bíp đơn giản thường được sử dụng để báo hiệu nhập liệu không hợp lệ.

Readkey( )

Đọc chỉ một thao tác nhấn phím đơn giản và trả về nó như một đối tượng ConsoleKeyInfo. Bạn có thể sử dụng đối tượng này để dễ dàng biết phím nào được nhấn (bao gồm các thao tác nhấn phím được mở rộng như các phím mũi tên) và những phím khác nào được nhấn giữ tại thời điểm (chẳng hạn Alt, Ctrl hay Shift).

MoveBufferArea( )

Sao chép một phần của cửa sổ console đến một vị trí mới và xóa dữ liệu gốc. Phương thức này cung cấp một cách có hiệu suất cao để di chuyển nội dung các console.

Các thuộc tính Console mới bao gồm:

Title

Xác lập tựa đề cửa sổ

ForegroundColor

Các xác lập màu text sẽ được sử dụng ở lần tới khi bạn sử dụng Console.Write( ) hay Console.WriteLine( )

BackgroundColor

Xác lập màu nền sẽ được sử dụng ở lần kế tiếp khi bạn sử dụng Console.Write( ) hay Console.WriteLine( ). Để ứng dụng vào nền này cho toàn bộ cửa sổ ngay tức khắc, hãy gọi Console.Clear() sau khi bạn xác lập màu nền.

CursorVisible

Ẩn con trỏ nhấp nháy khi xác lập sang False.

WindowHeight và WindowWidth

Trả về hay xác lập những kích thước cho cửa sổ console

CursorLeft và CursorTop

Trả về hay di chuyển vị trí con trỏ hiện tại.

6.7.2 Đọc một ký tự từ một vị trí được chỉ định của cửa sổ

Tiếc thay, lớp Console mới không cung cấp phương pháp nào để làm điều này. Điều đó có nghĩa là nếu bạn muốn mở rộng ví dụ happy face để , user phải định hướng qua một lô các ký tự khác, bạn sẽ cần lưu trữ vị trí của mỗi ký tự trong bộ nhớ (có thể trở nên nhàm chán) để kiểm tra vị trí được yêu cầu sau mỗi việc nhấn phím và ngăn chặn user di chuyển vào trong một khoảng trống đang bị chiếm bởi một ký tự khác.

6.8 Định giờ mã của bạn

Việc định giờ mã không khó. Bạn có thể sử dụng thuộc tính DateTime.Now để giữ ngày và giờ hiện hành ở mức mili giây. Tuy nhiên, phương pháp này không hoàn hảo. Việc khởi tạo đối tượng DateTime mất một khoảng thời gian ngắn và mật độ trễ có thể làm chênh lệch thời gian bạn ghi lại đối với các hoạt động ngắn. Các profiler quan trọng cần một phương pháp tốt hơn, một phương pháp sử dụng các cuộc gọi hệ thống cấp thấp và không có độ trễ.

Ghi chú

Lớp Stopwatch mới cho phép bạn theo dõi mã của bạn thực thi bao nhanh với độ chính xác vô song.

6.8.1 Bạn làm điều đó như thế nào?

Trong .NET 2.0, cách tốt nhất để định giờ mã của bạn là sử dụng một lớp Stopwatch mới trong namespace System.Diagnostics. Lớp Stopwatch thì dễ sử dụng. Tất cả những gì bạn cần làm là tạo một trường hợp thể hiện và gọi phương thức Start( ). Khi bạn kết thúc, hãy gọi Stop( ).

Ví dụ 6.8 minh họa một kỳ kiểm tra đơn giản định giờ một vòng lặp mất bao lâu để hoàn thành. Thời gian đã trôi qua lúc này được hiển thị bằng một số cách khác nhau, với độ chính xác khác nhau.

Ví dụ 6.8 Định giờ một vòng lặp

```
Module TimeCode

  Sub Main( )
    Dim Watch As New Stopwatch( )

    Watch.Start( )

    ' Delay for a while.
    For i As Integer = 1 To 1000000000
    Next

    Watch.Stop( )

    ' Report the elasped time.
    Console.WriteLine("Milliseconds " & Watch.ElapsedMilliseconds)
    Console.WriteLine("Ticks: " & Watch.ElapsedTicks)
    Console.WriteLine("Frequency: " & Stopwatch.Frequency)
    Console.WriteLine("Whole Seconds: " & Watch.Elapsed.Seconds)
    Console.WriteLine("Seconds (from TimeSpan): " & Watch.Elapsed.TotalSeconds)
    Console.WriteLine("Seconds (most precise): " & _
      Watch.ElapsedTicks / Stopwatch.Frequency)
  End Sub

End Module
```

Đây là kết xuất bạn sẽ thấy:

```
Milliseconds 10078
```

```
Ticks: 36075265

Frequency: 3579545

Whole Seconds: 10

Seconds (from TimeSpan): 10.0781705

Seconds (most precise): 10.078170549609
```

Bạn có thể truy xuất thời gian đã trôi qua bằng mili giây từ thuộc tính Stopwatch.ElapseMilliseconds. (Một giây là 1.000 mili giây). Thuộc tính ElapsedMilliseconds trả về một số nguyên 64 bit (Long), làm cho nó cực kỳ chính xác. Nếu truy xuất thời gian như một số giây hay số phút hữu ích hơn, hãy sử dụng thuộc tính Stopwatch.Elapsed vốn trả về một đối tượng TimeSpan.

Mặt khác, nếu bạn muốn độ chính xác lớn nhất, hãy truy xuất số nhịp đã trôi qua từ thuộc tính Stopwatch.ElapsedTicks. Các nhịp Stopwatch đều có ý nghĩa đặc biệt. Khi bạn sử dụng đối tượng TimeSpan hay DateTime, một nhịp đại diện cho 0.0001 của một mili giây. Tuy nhiên, trong trường hợp của một Stopwatch, các nhịp đại diện cho sự gia tăng thời gian có thể đo lường nhỏ nhất và tùy thuộc vào tốc độ của CPU. Để chuyển đổi các nhịp Stopwatch sang giây, hãy chia ElapsedTicks cho Frequency.

6.8.2 Tạm ngừng một bộ định giờ?

Nếu bạn muốn ghi lại tổng số thời gian được tận dụng để hoàn thành đa tác vụ, bạn có thể sử dụng Stop( ) để tạm ngưng một bộ định giờ (timer) và Start( ) để trở lại sau đó. Lúc này, bạn có thể đọc tổng số thời gian được tận dụng cho tất cả các tác vụ bạn đã định giờ từ các thuộc tính Elapsed và ElaspedMilliseconds.

Bạn cũng có thể chạy nhiều bộ định giờ cùng một lúc. Tất cả những gì bạn cần làm là tạo một đối tượng Stopwatch cho mỗi bộ định giờ riêng biệt mà bạn muốn sử dụng.

6.9 Triển khai ứng dụng của bạn với ClickOnce

Một trong những động lực thúc đẩy sự lựa chọn của các ứng dụng dựa trên trình duyệt là các tổ chức không cần triển khai các ứng dụng của họ cho client. Đa số công ty sẵn sàng chấp thuận những hạn chế của HTML để tránh những vấn đề lớn trong việc phân phối các phần cập nhật ứng dụng cho hàng trăm hay hàng ngàn user.

Ghi chú

Bạn muốn tính năng của một ứng dụng client phong phú với sự triển khai dễ dàng của một ứng dụng web? ClickOnce cung cấp một giải pháp mới cho việc triển khai phần mềm của bạn.

Việc triển khai một ứng dụng .NET Client sẽ không bao giờ đơn giản như việc cập nhật một website. Tuy nhiên, .NET 2.0 bao gồm một công nghệ mới gọi là ClickOnce đơn giản hóa việc triển khai một cách sống động.

6.9.1 Bạn làm điều đó như thế nào?

ClickOnce bao gồm một số tính năng đáng lưu ý sau đây:

- ClickOnce có thể tự động tạo một chương trình xác lập bạn có thể phân phối trên một CD hay thông qua mạng hay thông qua một trang web. Chương trình xác lập này có thể cài đặt những điều chủ yếu và tạo các biểu tượng Start menu thích hợp.
- ClickOnce có thể cấu hình ứng dụng của bạn để kiểm tra các phần cập nhật tự động mỗi khi nó khởi động (hoặc định kỳ trong nền). Tùy thuộc vào sở thích của bạn, bạn có thể cho user tùy chọn tải xuống và chạy phiên bản mới được cập nhật hay bạn có thể cài đặt nó vì ép buộc.
- ClickOnce có thể cấu hình ứng dụng của bạn để sử dụng một chế độ chỉ trực tuyến. Trong trường hợp này, user luôn luôn chạy phiên bản mới nhất của ứng dụng từ một trang web URL. Tuy nhiên, bản thân ứng dụng được lưu trữ cục bộ để cải thiện hiệu suất.

ClickOnce được kết hợp chặt chẽ với Visual Studio 2005, cho phép bạn triển khai một ứng dụng ClickOnce và một website bằng việc sử dụng menu lệnh Project > Publish.

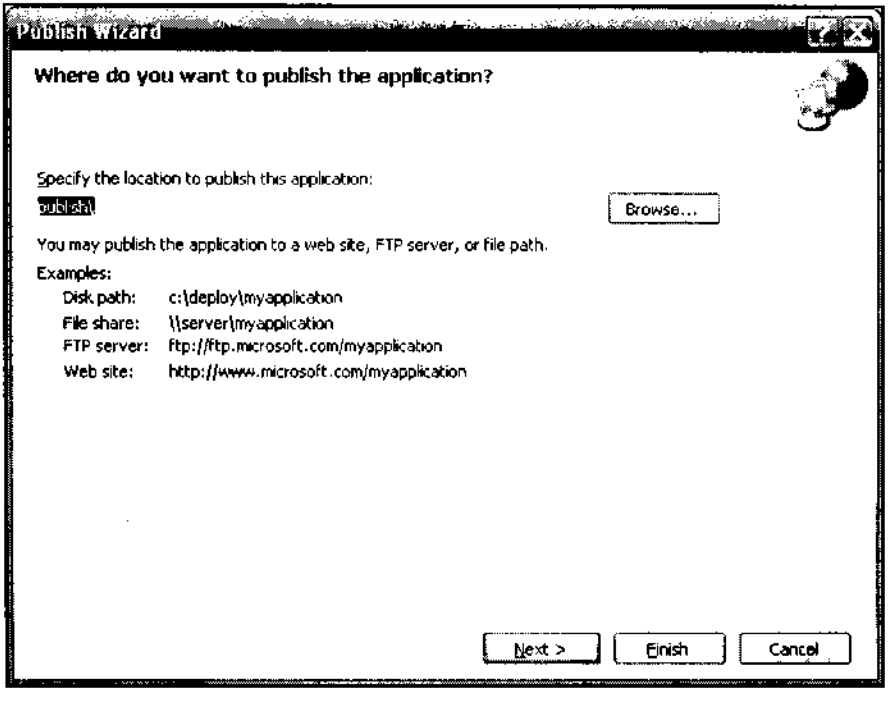
Các bước sau đưa bạn qua tiến trình chuẩn bị dự án để xuất bản:

1. Bằng việc sử dụng Visual Studio 2005, hãy tạo một dự án (project) mới. Một sự chọn lựa tốt là ứng dụng Windows Forms. Trước khi tiếp tục, hãy lưu project.
2. Chọn Build > Publish [ProjectName] (hay nhấp đúp vào project của bạn trong Solution Explorer và chọn Publish). Điều này tạo Publish Wizard cho bạn một cơ hội định rõ hay thay đổi các xác lập khác nhau.

3. Trang thoại đầu tiên của Publish Wizard (lời thoại "Where do you want to publish") nhắc nhở bạn chọn vị trí bạn sẽ phát hành các file được triển khia (xem hình 6.3). Vị trí này là đường dẫn file hay thư mục ảo trên Web server bạn muốn triển khai ứng dụng. Đối với mục kiểm tra đơn giản, hãy sử dụng một URL bắt đầu với http://localhost/ (ám chỉ máy tính hiện hành). Hãy nhấp Next để tiếp tục.

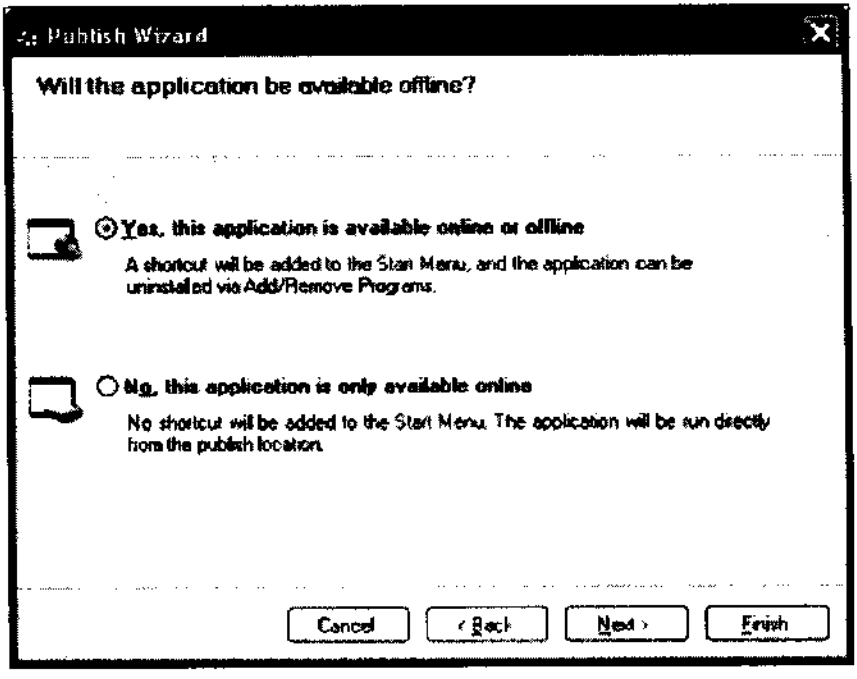
Thủ thuật

Khi Visual Studio xuất bản ứng dụng, nó sẽ tự động tạo một thư mục con có tên là publish trong thư mục ứng dụng hiện hành và nó sẽ ánh xạ điều này đến đường dẫn thư mục ảo bạn đã lựa chọn.



Hình 6.3 Lựa chọn một thư mục triển khai

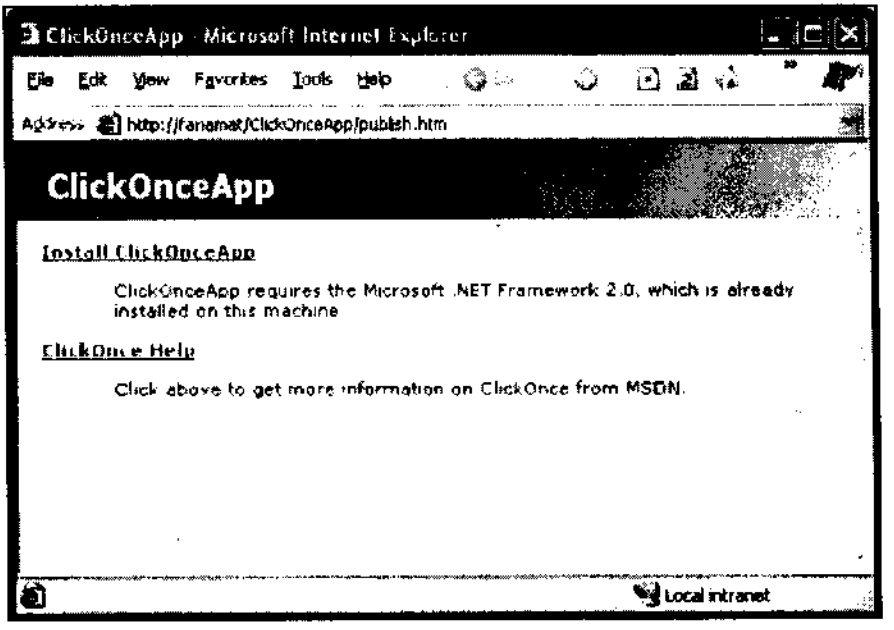
4. Kế tiếp, hãy chọn chế độ cài đặt (xem hình 6.4) bằng cách nhấp vào một trong các nút trên trang thoại "Will the application be available offline". Chọn "Yes, this application is available online or offline". Bằng cách này, xác lập sẽ bổ sung các biểu tượng ứng dụng cho Start menu. Nếu bạn chọn "No, this application is only available online", user sẽ chỉ có thể chạy nó bằng việc lướt đến thư mục ảo đã được xuất bản. Nhấp Next để tiếp tục.



Hình 6.4 Lựa chọn chế độ cài đặt.

5. Publish Wizard bây giờ hiển thị một bảng tóm tắt các xác lập. Hãy nhấp Finish để xuất bản nó (bạn có thể xuất bản một phiên bản được cập nhật bất kỳ lúc nào bằng việc chọn Build Publish [ProjectName] từ menu).

   Một khi wizard kết thúc, trang web ClickOnce đã được tạo tự động sẽ được khởi động như ở hình 6.5. Bằng cách sử dụng trang này, một user có thể nhấp để tải xuống và cài đặt ứng dụng của bạn. Hãy tiến hành kiểm tra nó bằng cách nhấp vào liên kết Install [AppName] .



Hình 6.5 Trang cài đặt ClickOnce

Tiến trình cài đặt chạy không có bất kỳ thông điệp nào, trừ khi nó cần sự đồng ý của user. Ví dụ, trước khi cài đặt có thể bổ sung một biểu tượng cho Start menu, nó cần nhắc nhở user.

Tốt nhất là bây giờ bạn đã có ứng dụng, bạn có thể tận dụng khả năng cập nhật tự động của nó. Để kiểm tra điều này, hãy trở về ứng dụng ở Visual Studio .NET và thay đổi form chính (có lẽ bằng việc thêm một nút mới). Sau đó, gia tăng số phiên bản của ứng dụng. (Để làm điều này, hãy nhấp đúp vào hạng mục My Project trong Solution Explorer, chọn tab Application và nhấp nút AssemblyInfo. Một hộp thoại sẽ xuất hiện cho phép bạn xác lập dữ liệu metadata, bao gồm số phiên bản). Cuối cùng, xuất bản lại ứng dụng.

Khi một phiên bản mới có sẵn trên server, các ứng dụng client sẽ tự động cập nhật, dựa trên các xác lập cập nhật của chúng. Nếu bạn chạy ứng dụng mẫu được cài đặt, nó sẽ kiểm tra các cập nhật khi nó bắt đầu. Trong trường hợp này, nó sẽ phát hiện phiên bản mới và nhắc nhở bạn cài đặt cập nhật.

Mục lục

TỰ HỌC NGÔN NGỮ LẬP TRÌNH VISUAL BASIC 2005

NGUYỄN NAM THUẬN

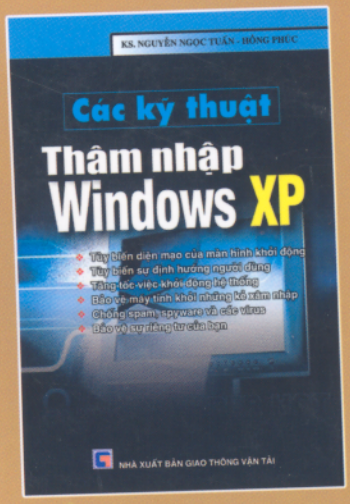
NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

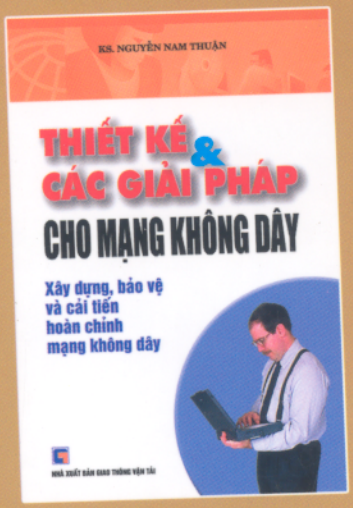
Chịu trách nhiệm xuất bản:
LÊ TỪ GIANG
Biên tập : Hoàng Chí Dũng
Trình bày : Thế Anh
Bìa : Lê Thành

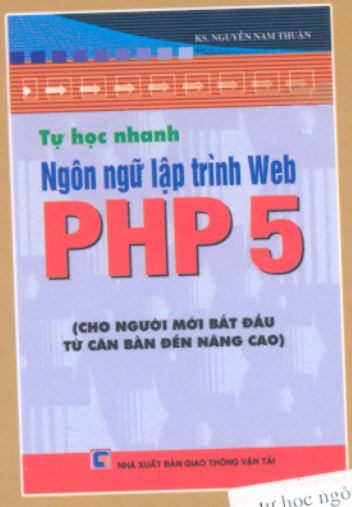
TỔNG PHÁT HÀNH
CÔNG TY CỔ PHẦN VĂN HÓA NHÂN VĂN
Số 1 Trường Chinh - P.11 - Q. Tân Bình – TP. Hồ Chí Minh
ĐT: 9712285 – 9710306 - 8490048 • FAX: 9712286
NHÀ SÁCH NHÂN VĂN • 189 CMT8 - P.7 - Q. Tân Bình – TP. Hồ Chí Minh.
ĐT: 9700420
NHÀ SÁCH NHÂN VĂN • 486 Nguyễn Thị Minh Khai - P.2 - Q. 3 – TP. Hồ Chí Minh.
ĐT: 8396733
SIÊU THỊ SÁCH NHÂN VĂN • 394 Quốc lộ 15 - P. Trung Dũng - TP. Biên Hoa.
ĐT: (061)510959

In 1.000 cuốn, khổ 16 x 24cm. Tại Công ty Cổ phần In Gia Định, số 9D Nơ Trang Long, Q. BT, TP. Hồ Chí Minh - ĐT: 8412644. Giấy đăng ký KHXB số: 49-230/XB-QLXB do Cục xuất bản cấp ngày 03 tháng 03 năm 2005. In xong và nộp lưu chiểu tháng 12 năm 2005.

Tự học
Ngôn ngữ lập trình
Visual Basic 2005







NHAN VAN
CÔNG TY CỔ PHẦN VĂN HÓA NHÂN VĂN
Nhà Sách NHÂN VĂN
*486 Nguyễn Thị Minh Khai, P.2, Quận 3, Thành Phố Hồ Chí Minh - Tel/Fax: 8396733
*Số 01 Trường Chinh, P. 11, Q. Tân Bình
Tel: 9712285 - 9710306 * Fax: 9712286