Chương 21

Các thủ tục lưu trữ ADO

- Thi hành các thủ tục lưu trữ.
- Tìm hiểu cách thi hành các thủ tục lưu trữ cho trở lại dữ liệu.
- Tìm hiểu cách chuyển các tham số tới thủ tục lưu trữ.
- Tìm hiểu cách sử dụng các tham số OUTPUT.

Thi hành các thủ tục lưu trữ

Rất có thể bạn sẽ dùng các thủ tục đã lưu trữ trong một ứng dụng CSDL bạn phát triển. Bạn có thể dùng các thủ tục lưu trữ để cho trở lại dữ liệu hoặc thi hành các truy vấn hành động. Các thủ tục lưu trữ có thể chấp nhận các tham số như các tham số nhập liệu, tham số xuất liệu hoặc cả hai.

Bạn có thể sử dụng các thủ tục lưu trữ vì nhiều nguyên nhân khác nhau. Các thủ tục lưu trữ thường hiệu quả hơn SQL động tĩnh. Các thủ tục lưu trữ lưu SQL trên máy chủ trung tâm thay vì lập lại SQL khắp chương trình của bạn. Chúng được biên dịch và lưu trữ sự thi hành có kế hoạch trên máy chủ, vì vậy chúng tải và thi hành nhanh. Bạn có thể đưa ra quyết định về SQL nào thi hành bên trong thủ tục lưu trữ dựa vào các tham số được chuyển vào.

Bạn sẽ thực thi một số thủ tục lưu trữ trong bước khởi động (jumpstart) này bằng cách dùng ADO.NET. Bạn sẽ tìm hiểu nhiều phương thức chuyển các tham số khác nhau tới các thủ tục lưu trữ. Bạn sẽ tìm hiểu cách truy tìm dữ liệu từ thủ tục lưu trữ. Bạn sẽ thi hành câu lệnh UPDATE trong thủ tục lưu trữ bằng ADO.NET. Bạn cũng sẽ hiểu cách cho trở lại thông tin từ một thủ tục lưu trữ bằng cách dùng các tham số OUTPUT. Hình 1 trình bày màn hình mẫu mà bạn sẽ sử dụng trong bước khởi động này.

Hình 1: *Bạn sẽ dùng màn hình này để thi hành nhiều thủ tục lưu trữ khác nhau.*

Các bước

Ở mẫu đầu tiên này, bạn sẽ tìm hiểu cách chuyển tham số tới một thủ tục lưu trữ và truy tìm các bản ghi từ thủ tục lưu trữ đó. Bạn sẽ dùng đối tượng OleDbCommand và OleDbDataReader. Bạn có thể tìm hiểu thêm về đối tượng OleDbCommand trong bước khởi động có tên "Đối tượng ADO.NET Connection và ADO.NET Command". Đối tượng OleDbDataReader là một cấu trúc giống Forward-Only Recordset trong các đối tượng ADO cũ.

1. Nạp file StoredProcs.sln được chứa trong folder FirstStep.

2. Nhấp đôi nút Execute ở góc phải trên để đưa ra thủ tục sự kiện Click.

3. Gõ mã được trình bày bên dưới vào.

```
Private Sub btnOrders_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnOrders.Click
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim strConn As String
    Dim strSQL As String

    ' Build the connection string
    strConn = ConnectStringBuild()

    strSQL = "CustOrdersOrders @CustomerID=?"

    ' Setup a Command object
    oCmd = New OleDb.OleDbCommand()
    With oCmd
        ' Create New Connection
        .Connection = _
         New OleDb.OleDbConnection(strConn)
        ' Open Connection
        .Connection.Open()
        ' Set SQL string
        .CommandText = strSQL
        ' .CommandType = CommandType.StoredProcedure
```

```
            ' Add parameters
            .Parameters.Add("CustID", _
             Data.OleDb.OleDbType.Char, 5)
            .Parameters("CustID").Value = txtCustID.Text
            ' Execute stored procedure
            oDR = _
.ExecuteReader(CommandBehavior.SequentialAccess)
    End With

    lstOrders.Items.Clear()
    lstOrders.Refresh()
    Do While oDR.Read()
        lstOrders.Items.Add( _
         oDR.Item("OrderID").ToString())
    Loop
    oDR.Close()

    ' Close the Connection
    oCmd.Connection.Close()
End Sub
```

Đầu tiên bạn khai báo đối tượng OleDbCommand và OleDbDataReader. Hai đối tượng này được dùng để đệ trình SQL và cho tập dữ liệu trở lại chương trình. Bạn xây dựng một chuỗi kết nối bằng cách gọi hàm ConnectStringBuild(). Chuỗi SQL chứa tên của thủ tục lưu trữ, CustOrdersOrders, cùng với tên của tham số khi nó xuất hiện trong thủ tục lưu trữ. Bạn đặt một dấu chấm hỏi như là một trình giữ chỗ cho giá trị bạn sẽ chuyển tới tham số ấy. Thủ tục lưu trữ CustOrdersOrders được chuyển một ID khách hàng đặc biệt từ bảng Customers trong CSDL Northwind và sẽ cho trở lại tất cả các đơn đặt hàng kết hợp cho khách hàng đó.

Bạn sẽ tạo một thể nghiệm mới về đối tượng kết nối và gán nó vào đặc tính Connection của đối tượng lệnh. Sau đó bạn xác lập đặc tính CommandText là chuỗi SQL chứa tên thủ tục lưu trữ và tham số. Đừng xác lập đặc tính CommandType là StoredProcedure, nếu không mẫu này sẽ không hoạt động.

Để chuyển giá trị tới tham số trong câu lệnh SQL, bạn phải gọi ra phương thức Add trên tập hợp Parameters để tạo tham số mới. Bạn chuyển tới cấu tử Add tên bạn muốn gọi tham số này, kiểu dữ liệu tham số và chiều dài dữ liệu bạn sẽ chuyển vào. Sau khi bạn tạo tham số này, bạn xác lập đặc tính Value của đặc tính này là dữ liệu bạn nhận trong hộp văn bản txtCustID.

Bạn thi hành thủ tục lưu trữ bằng cách dùng phương thức Execute của đối tượng Command. Nó sẽ cho trở lại đối tượng OleDbDataReader được điền với tập kết quả của các đơn đặt hàng của khách hàng được cho trở lại từ thủ tục lưu trữ.

Bây giờ bạn có thể lặp qua đối tượng OleDbDataReader bằng cách dùng phương thức Read và truy tìm mỗi cột từ tập kết quả bằng cách dùng phương thức Item của đối tượng OleDbDataReader. Như vậy bạn vừa tìm hiểu về cách truy tìm tập kết quả từ một thủ tục lưu trữ và nạp nó vào hộp danh sách.

Tạo chuỗi kết nối

Trước khi bạn có thể chạy mẫu này, bạn sẽ phải tạo một hàm tổng quát sẽ được gọi bởi mẫu này và nhiều mẫu bạn sẽ tạo trong bước khởi động này.

1. Mở cửa sổ mã cho form này, tìm một dòng trống bất kỳ nơi đâu bên ngoài thủ tục.

2. Gõ mã bên dưới vào để tạo hàm sẽ xây dựng chuỗi kết nối này.

```
Public Function ConnectStringBuild() As String
   Dim strConn As String
```

```
    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User ID=sa;"
    strConn &= "Password=;"

    Return strConn
End Function
```

Phương thức ConnectStringBuild giả định bạn đã cài đặt SQL Server và có thể truy xuất được nó. Chuỗi nhà cung cấp ở mã trên có tham số Data Source được xác lập là (local), là máy cục bộ của bạn. Bạn có thể phải sửa đổi tham số này để trỏ tên của máy SQL Server của bạn. Bạn cũng có thể phải sửa đổi tham số User ID thành một User ID khác và có thể phải thay đổi tham số Password để có thể đăng nhập chính xác máy chủ này.

Nếu bạn không truy xuất được SQL Server, thay vào đó, có thể bạn phải sử dụng CSDL Northwind Access. CSDL Northwind được phân phối với Microsoft Access. Nếu bạn muốn kết nối với CSDL Access này, bạn sẽ dùng chuỗi nhà cung cấp sau.

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Northwind\Northwind.mdb"
```

Đương nhiên, bạn sẽ phải thay đổi đường dẫn ở trên thành đường dẫn hợp lệ cho CSDL Northwind này.

Bây giờ bạn thấy chương trình này hoạt động.

1. Chạy chương trình.
2. Nhấp nút Execute.
3. Bây giờ bạn sẽ thấy một danh sách các số hiệu đơn đặt hàng xuất hiện trong hộp danh sách.

Sử dụng đặc tính CommandType

Thay vì đặt tham số ngay vào chuỗi SQL như bạn đã làm ở thí dụ trước, bạn có thể bỏ qua nó. Thực hiện các bước sau để xem nó hoạt động như thế nào.

1. Chép mã từ thủ tục sự kiện Click của nút Execute. Bạn sẽ chỉ chép mã bên trong thủ tục sự kiện, chứ không phải bên trong khai báo thủ tục sự kiện.

2. Nhấp đôi nút Execute 2.

3. Dán mã vào thủ tục sự kiện btnOrders2_Click.

4. Đổi dòng sau:

```
strSQL = "CustOrdersOrders @CustomerID=?"
```

thành:

```
strSQL = "CustOrdersOrders"
```

5. Bạn cũng phải bỏ chú thích dòng xác lập CommandType:

```
oCmd.CommandType = CommandType.StoredProcedure
```

Bạn có thể để nguyên phần mã còn lại khi thủ tục này làm việc.

6. Chạy chương trình và nhấp nút Execute 2 và bạn sẽ thấy tập các lệnh đặt hàng giống hệt xuất hiện trong hộp danh sách.

Ở phiên bản này bạn đã phải dùng đặc tính CommandType, trong khi ở thí dụ đầu tiên bạn không sử dụng. Thực ra, sử dụng phiên bản mã này tốt hơn nhiều khi gọi các thủ tục lưu trữ so với phiên bản đầu tiên. Điều này cho thấy rõ bạn đang gọi một thủ tục lưu trữ vì bạn sẽ xác lập đặc tính CommandType cụ thể.

Không phải sử dụng các tham số

Không nhất thiết bạn phải sử dụng tập hợp Parameters để thi hành thủ tục lưu trữ này. Bạn chỉ cần chuyển giá trị của tham số ngay bên trong câu lệnh SQL.

1. Nhấp đôi nút Execute 3.

2. Gõ mã sau vào thủ tục sự kiện Click cho nút btnOrders3.

```
Private Sub btnOrders3_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnOrders3.Click
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim strConn As String
    Dim strSQL As String

    ' Build the connection string
    strConn = ConnectStringBuild()

    strSQL = "CustOrdersOrders @CustomerID='" & _
     txtCustID.Text & "'"

    ' Setup a Command object
    oCmd = New OleDb.OleDbCommand()
    With oCmd
        .Connection                    =                    New
OleDb.OleDbConnection(strConn)
        .Connection.Open()
        .CommandText = strSQL
        oDR = _

.ExecuteReader(CommandBehavior.SequentialAccess)
```

```
    End With

    lstOrders.Items.Clear()
    Do While oDR.Read()
            lstOrders.Items.Add( _
             oDR.Item("OrderID").ToString())
    Loop
    oDR.Close()

    ' Close the Connection
    oCmd.Connection.Close()
End Sub
```

Bây giờ bạn sẽ chuyển giá trị của tham số ngay bên trong câu lệnh SQL. Bạn chỉ cần đặt câu lệnh SQL này vào đặc tính CommandText và đệ trình SQL bằng cách dùng phương thức Execute. Bạn không cần tạo các tham số vào tập hợp Parameters.

Cập nhật dữ liệu bằng cách dùng các thủ tục lưu trữ

Bạn có thể sử dụng các thủ tục lưu trữ để cập nhật dữ liệu cũng như truy tìm dữ liệu. Ở mẫu kế tiếp, bạn sẽ thi hành một thủ tục lưu trữ mới để chấp nhận một số tham số và thực hiện câu lệnh UPDATE.

Các bước

1. Trước khi bạn có thể chạy thí dụ này, bạn sẽ phải nạp thủ tục lưu trữ có tên CustUpdateRegion mà chúng ta vừa tạo cho thí dụ này.

2. Mở Query Analyzer của SQL Server.

3. Kết nối với CSDL Northwind.

4. Nạp file Storedprocs.sql được định vị ở folder FirstStep cho bước khởi động này.

5. Thi hành câu lệnh CREATE PROCEDURE này để xây dựng thủ tục lưu trữ trong CSDL Northwind. Thủ tục lưu trữ được trình bày bên dưới.

```
CREATE PROCEDURE CustUpdateRegion
   @RowsUpdated int OUTPUT,
   @Region nvarchar(15),
   @Country nvarchar(15)
AS
UPDATE Customers
SET Region = @Region
WHERE Country = @Country

SELECT @RowsUpdated = @@rowcount
GO
```

Thủ tục lưu trữ ở trên sẽ cho phép bạn cập nhật cột Region trong bảng Customers dựa vào tham số bạn chuyển vào. Cột Region sẽ được thay đổi cho tất cả các khách hàng ở trong một mã vùng nào đó mà bạn cũng chuyển vào thủ tục lưu trữ này. Ngoài việc chuyển dữ liệu vào, bạn sẽ sử dụng tham số OUTPUT, @RowsUpdated, để truy tìm một số dòng có ảnh hưởng. Bạn thực hiện điều này bằng cách lựa @@ROWCOUNT từ SQL Server và đặt nó vào tham số @RowsUpdated.

Tạo thủ tục cập nhật

Bây giờ bạn tạo thủ tục sẽ thi hành thủ tục lưu trữ này bằng cách chuyển vào Region và Country.

1. Nhấp đôi nút Execute ADO.

2. Viết mã được trình bày bên dưới vào thủ tục sự kiện btnUpdate_Click.

```
Private Sub btnUpdate_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnUpdate.Click
    Dim oCmd As OleDb.OleDbCommand
    Dim oParam As OleDb.OleDbParameter
    Dim strConn As String
    Dim strSQL As String

    ' Build the connection string
    strConn = ConnectStringBuild()

    ' Build the call to the stored procedure
    strSQL = "CustUpdateRegion"

    ' Setup a Command object
    oCmd = New OleDb.OleDbCommand()
    With oCmd
        .Connection =
         New OleDb.OleDbConnection(strConn)
        .Connection.Open()
        .CommandText = strSQL
        .CommandType = CommandType.StoredProcedure

        ' Create the OUTPUT parameter
        oParam = .Parameters.Add("RowsUpdated", _
         Data.OleDb.OleDbType.Integer)
        oParam.Direction = ParameterDirection.Output
        ' Create the Region Parameter
        .Parameters.Add("Region", _
         Data.OleDb.OleDbType.VarChar, 15)
        .Parameters("Region").Value = txtRegion.Text
        ' Create the Country Parameter
        .Parameters.Add("Country", _
         Data.OleDb.OleDbType.VarChar, 15)
        .Parameters("Country").Value = txtCountry.Text

        ' Execute the Update
        .ExecuteNonQuery()

        ' Retrieve the RowsUpdated OUTPUT value
        txtRowsUpdated.Text = _
         .Parameters("RowsUpdated").Value.ToString()
```

```
            ' Close the Connection
            .Connection.Close()
        End With
    End Sub
```

Sau khi bạn tạo tham số đầu tiên, RowsUpdated, đối tượng Parameter được cho trở lại từ phương thức Add. Bạn sử dụng đối tượng tham số này để xác lập hướng là ParameterDirection.Output. Bạn sẽ báo ADO tạo một trình giữ chỗ cho tham số sẽ được trở lại từ thủ tục lưu trữ. Hai tham số khác được tạo giống như bạn đã làm ở thí dụ trước đây. Để thi hành thủ tục lưu trữ này, bạn sẽ gọi ra phương thức ExecuteNonQuery trên đối tượng OleDbCommand. Bạn sẽ dùng phương thức ExecuteNonQuery khi bạn không mong đợi dữ liệu được cho trở lại từ thủ tục lưu trữ.

1. Sau khi lệnh đã được thi hành, bạn có thể sử dụng tập Parameters để truy tìm giá trị của tham số OUTPUT.

2. Chạy chương trình và nhấp nút Execute ADO để xem bao nhiêu dòng được cập nhật trên bảng này sau khi thi hành thủ tục lưu trữ.

Sử dụng các đối tượng SQL*

Thay vì sử dụng đối tượng OleDbCommand và OleDbConnection, bạn có thể sử dụng đối tượng SQLClient.SqlCommand và SQLClient.SqlConnection mới. Các đối tượng này bỏ qua tầng OLE DB truyền thống và truyền thông trực tiếp với SQL Server. Bạn sẽ thấy giao diện rất giống giữa hai tập đối tượng. Thử chúng bằng các bước sau.

1. Nhấp đôi nút Execute SQL.

2. Viết mã sau vào thủ tục sự kiện btnSQL_Click.

```
Private Sub btnSQL_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs, Handles btnSQL.Click
    Dim oCmd As SqlClient.SqlCommand
    Dim oParam As SqlClient.SqlParameter
    Dim strConn As String
    Dim strSQL As String

    ' Build the connection string
    strConn = SQLConnectStringBuild()

    ' Build the call to the stored procedure
    ' If more than one parameter, you must
    ' use ? placeholders
    strSQL = "CustUpdateRegion"

    ' Setup a Command object
    oCmd = New SqlClient.SqlCommand()
    With oCmd
        .Connection                    =                    New
SqlClient.SqlConnection(strConn)
        .Connection.Open()
        .CommandText = strSQL
        .CommandType = CommandType.StoredProcedure

        ' Create the OUTPUT parameter
        oParam = .Parameters.Add( _
            New   SqlClient.SqlParameter("@RowsUpdated",
_
            Data.SqlClient.SqlDbType.Int))
        oParam.Direction = ParameterDirection.Output

        ' Create the INPUT parameters
        .Parameters.Add( _
            New SqlClient.SqlParameter("@Region", _
            Data.SqlClient.SqlDbType.VarChar, 15))
        .Parameters("@Region").Value = txtRegion.Text

        .Parameters.Add( _
            New SqlClient.SqlParameter("@Country", _
            Data.SqlClient.SqlDbType.VarChar, 15))
        .Parameters("@Country").Value = txtCountry.Text

        ' Execute the Update
        .ExecuteNonQuery()
```

```
            ' Retrieve # of rows updated
            txtRowsUpdated.Text =
.Parameters("@RowsUpdated").Value.ToString()

            ' Close the Connection
            .Connection.Close()
        End With
    End Sub
```

Khi bạn khảo sát mã vừa gõ vào, nó trông giống như ở thí dụ ADO. Thay vì sự khác nhau duy nhất là tiền tố ADO trên tất cả các đối tượng, thì bạn sử dụng tiền tố SQL.

Xây dựng chuỗi kết nối SQL

Các đối tượng SQL sử dụng định dạng chuỗi kết nối hơi khác với chuỗi nhà cung cấp OLE DB. Bạn phải thêm hàm sau vào form này để bạn có thể sử dụng chuỗi kết nối thích hợp cho thí dụ vừa qua.

```
Public Function SQLConnectStringBuild() As String
   Dim strConn As String

   strConn = "Server=(local);"
   strConn &= "Database=Northwind;"
   strConn &= "User ID=sa;"
   strConn &= "Password=;"

   Return strConn
End Function
```

Trong chuỗi kết nối SQL, bạn chỉ định tên máy chủ bằng cách dùng Server= parameter và CSDL để kết nối bằng cách dùng Database= parameter. User ID và Password có thể được chỉ định như được trình bày ở trên hoặc sử dụng UID= and PWD=.

Bây giờ bạn chạy chương trình một lần nữa, nhấp nút Execute SQL và bạn sẽ nhận các kết quả giống như nút Execute ADO. Sự khác nhau duy nhất là bạn thông qua các nhà cung cấp SQL địa phương thay vì nhà cung cấp OLE DB.

Việc sử dụng các thủ tục lưu trữ là cách để trình SQL với CSDL back end (phía sau) hiệu quả. Tuy nhiên, một trong các trở ngại là mã không đọc được khi bạn phải xem ở hai vị trí để thực sự thấy tất cả mã. Một lưu ý khác, thay vì sử dụng tham số OUTPUT để cho trở lại số dòng bị ảnh hưởng bởi thủ tục lưu trữ, bạn cũng có thể sử dụng đặc tính RecordsAffected trên đối tượng OleDbCommand và SQLCommand. Nếu bạn biết bạn sẽ chỉ sử dụng SQL Server, bạn nên sử dụng các đối tượng SQL vì chúng hiệu quả hơn các đối tượng ADO một ít.

Tóm tắt

Ở chương này, bạn đã tìm hiểu cách gọi các thủ tục lưu trữ của ADO.NET. Bạn đã tìm hiểu cách truy tìm dữ liệu và cập nhật dữ liệu bằng các thủ tục lưu trữ. Bạn cũng đã hiểu cách cho trở lại dữ liệu từ tham số OUTPUT.

Câu hỏi ôn tập

1. Tại sao bạn sử dụng các thủ tục lưu trữ trong một ứng dụng?

2. Có hay không: Có sự khác nhau nào giữa chuỗi kết nối OleDb và chuỗi kết nối SqlClient?

3. Nếu bạn chuyển tên tham số và một dấu chấm hỏi như trình giữ chỗ, bạn có cần xác lập đặc tính CommandType trên đối tượng OleDbCommand không?

4. Bạn phải lập tiền tố các tham số gì khi sử dụng đối tượng SqlClient.SqlParameter?

Bài tập

- Tạo một thủ tục lưu trữ để truy tìm tất cả các cột cho một ID sản phẩm riêng biệt được chuyển vào nó.
- Tạo một form để cho phép bạn nhập liệu ID sản phẩm, chuyển ID đó vào thủ tục lưu trữ và hiển thị các kết quả trong control DataGrid.

Trả lời câu hỏi ôn tập

1. Chúng nhanh hơn, tập trung cho bảo trì hơn, dễ chuyển tham số vào.

2. Đúng.

3. Bạn không thể xác lập CommandType nào.

4. Dấu (@).

Chương 22

XML

- Tìm hiểu về XML.
- Tại sao XML quan trọng.
- Tìm hiểu cách hiển thị file XML.
- Tìm hiểu về XSL.

XML là gì?

XML (Extensible Markup Language – Ngôn ngữ đánh dấu mở rộng) là một ngôn ngữ mô tả được tạo nên bằng các tag giống như HTML. Không giống HTML, các tag của XML có thể được bạn định nghĩa để mô tả một kiểu dữ liệu bất kỳ. Không có các tag được định nghĩa sẵn, thay vào đó bạn tạo các tag của riêng bạn dựa vào kiểu dữ liệu bạn muốn mô tả.

Chúng ta hãy khảo sát một thí dụ về chuỗi XML tiêu biểu. Listing 1 trình bày một trong các tag này thể hiện như thế nào.

```
<People>
    <Name>
        <id>999-99-9999</id>
        <Firstname>Paul</Firstname>
        <Lastname>Sheriff</Lastname>
    </Name>
    <Name>
      <id>111-11-1111</id>
      <Firstname>Michael</Firstname>
      <Lastname>Krasowski</Lastname>
    </Name>
    <Name>
        <id>222-22-2222</id>
        <Firstname>Keith</Firstname>
        <Lastname>Nash</Lastname>
    </Name>
</People>
```

***Listing 1:** Chuỗi XML tiêu biểu.*

Cấu trúc XML

Chuỗi XML được trình bày ở Listing 1, một tag đã được tạo có tên <People> để mô tả toàn bộ nội dung của dữ liệu. Sau đó chúng ta cần có ba dòng mã, vì vậy ở đây đã tạo một số tag <Name> để chứa đựng ba trường; <id>, <Firstname> và <Lastname>. Chúng được gọi là các **Element**.

XML được tạo nên bởi:

- Các Tag
- Các Element
- Các Attribute
- Các Entity

- Các Comment

Các Tag

Một Tag là một dấu định danh của một phần tử (element). Biểu tượng nhỏ hơn (<) và lớn hơn (>) được dùng để nhận dạng tag. Bạn tự đặt tên tag. Thí dụ:

```
<First>John</First>
```

Các nguyên tắc đặt tên tag của XML rất đơn giản. Bên dưới là một số nguyên tắc đặt tên tag:

- Phải chứa một chữ.
- Phải bắt đầu bằng một chữ hoặc một gạch dưới.
- Gạch dưới không thể ở bên trong bất kỳ nơi nào khác.
- Có thể chứa các số, các chữ hoặc dấu hai chấm (:).
- Các tag không thể có một tag bên trong nó.
- Tên không có kí tự trắng.
- Có thể sử dụng dấu gạch nối và một dấu chấm.
- Cố tạo các tài liệu XML cho người ta dễ đọc.
- Nhạy kiểu chữ!

Các Element

Một phần tử (element) là một Tag với dữ liệu nào đó. Bạn đặt dữ liệu này ở giữa tag bắt đầu và tag dừng. Bạn có thể tạo các phần tử trống không có dữ liệu bằng cách dùng một trong hai cú pháp sau:

```
<City></City>
```

hoặc

```
<City/>
```

Một tài liệu XML phải có một phần tử gốc.

Attribute

Thuộc tính (attribute) được kết hợp với một phần tử tag. Nó được dùng để định tính thêm dữ liệu trong một phần tử. Bạn có thể nghĩ rằng điều này giống như một đặc tính của đối tượng. Thí dụ, nó có thể giống như sau:

```
<Employee id="a1234">
```

Entity

Một thực thể (entity) chính là tài liệu XML. Nó thường cư trú trên đĩa ở định dạng ASCII.

Comment

Chú thích (comment) được dùng cung cấp tư liệu cho XML. Bạn có thể tạo các chú thích trên một dòng, trên nhiều dòng và có thể ở sau một tag bất kỳ trong tài liệu.

```
<!-- This is a comment -->
```

Cấu trúc tài liệu XML

Một tài liệu XML sẽ có các phần tử sau:

- XML Declaration (Khai báo)
- Root Element (Phần tử gốc)
- Các Element và các Attribute

XML Declaration nhận dạng thực thể như một tài liệu XML. Nó sẽ nhận dạng phiên bản đặc tả XML mà tài liệu này tham gia. Khai báo được tùy chọn, nhưng phải có.

```
<?xml version="1.0"?>
```

Root Element là phần tử trên đầu để nhận dạng tài liệu. Bạn chỉ có thể có một phần tử gốc cho mỗi tài liệu XML. Trong Listing 1, phần tử <People> là phần tử gốc. Các phần tử có thể có phần tử Parent hoặc một hay nhiều phần tử Child và Sibling. Trong Listing 1, <Name> là cha của <ID>, <Firstname> và <Lastname>. <Firstname> là con của <Name>, <Lastname> là anh em của <Firstname> và <ID>.

XML dựa trên Element so với Attribute

Listing 1 là một thí dụ về XML dựa trên phần tử. Listing 2 trình bày một thí dụ về XML dựa trên thuộc tính.

```
<People>
   <Name id='999-99-9999' FirstName='Paul'
```

```
    LastName='Sheriff' />
  <Name id='111-11-1111' FirstName='Michael'
    LastName='Kreecweki' />
  <Name id='222 22 2222' FirstName='Keith'
    LastName='Nash' />
</People>
```

***Listing 2:** XML sử dụng các Attribute.*

Ở XML trên bạn định nghĩa mỗi phần tử bằng một tên, một dấu bằng (=) và một giá trị bên trong dấu trích dẫn đơn. Bạn dừng tag bằng cách dùng một />. Các định dạng trong listings 1 và 2 đều tương đương.

DTD và Schema

Dù các định dạng ở trên dường như chúng ta dễ đọc và hiểu, nhưng một chương trình máy tính cần có định nghĩa về các phần tử dữ liệu khác nhau. Nếu bạn sẽ truyền thông với một ứng dụng mà bạn chưa biết rõ, thì hệ thống sẽ cung cấp cho bạn mô tả về dữ liệu mà nó sẽ gửi bằng XML. Định nghĩa này sẽ ở định dạng Document Type Definition (DTD) hoặc Schema. DTD định dạng kiểu cổ, khó đọc không rõ ràng lắm. Schema là một XML dựa trên ngôn ngữ mô tả và sẽ trở thành chuẩn thực tế vào các năm sắp tới.

Bên dưới là một mẫu về DTD, sẽ mô tả XML ở Listing 3:

```
<!ELEMENT People(name)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT Firstname (#PCDATA)>
<!ELEMENT Lastname (#PCDATA)>
<!ELEMENT Name (id, Firstname, Lastname)>
```

***Listing 3:** Một Document Type Definition (DTD) mẫu.*

Một thí dụ về Schema của cùng XML sẽ giống như sau:

```
<Schema xmlns="schemas-microsoft-com:xml-data">
   <ElementType name="id"/>
   <ElementType name="Firstname"/>
   <ElementType name="Lastname"/>
   <ElementType name="Name" model="open">
     <element type="id"/>
     <element type="Firstname"/>
     <element type="Lastname"/>
   </ElementType>
   <ElementType name="People" model="closed">
     <element type="Name" maxOccurs="*"/>
   </ElementType>
</Schema>
```

***Listing 4:** Một định nghĩa Schema mẫu.*

Định dạng Schema được coi là dễ đọc hơn khi nó thể hiện giống XML để mô tả rõ hơn.

HTML

HTML (HyperText Markup Language – Ngôn ngữ đánh dấu siêu văn bản) là một ngôn ngữ hoàn toàn khác với XML. HTML có một tập các tag được sử dụng giới hạn. Nó không kiểm tra kiểu chặt chẽ và không dễ dàng mở rộng được như XML. HTML được phác họa như một cơ chế hiển thị và là một cách để trình bày dữ liệu. XML được phác họa như một cách để mô tả dữ liệu. Do đó cả hai hỗ trợ lẫn nhau.

Tại sao XML quan trọng?

XML sẽ rất quan trọng trong một vài năm tới khi nó trở thành chuẩn để trao đổi dữ liệu. Nhà doanh nghiệp đang tận dụng ngôn ngữ mềm dẻo, chuẩn và mạnh này. Bạn sẽ bắt đầu thấy một định dạng XML thông dụng để trao đổi các hóa đơn, thanh toán thẻ tín dụng, chuyển giao hồ sơ y khoa, vân vân.

XML có thể mô tả các bảng có quan hệ dễ dàng và thậm chí còn có thể mô tả các mối quan hệ giữa các bảng. XML sẽ thay thế EDI như một chuẩn vì EDI mất quá lâu mới trở thành một chuẩn, quá nhiều người muốn tùy biến riêng, vân vân. XML có thể thêm các đuôi mở rộng mà không phải thay đổi cấu trúc cơ sở.

Sử dụng XML

XML thường được dùng để gửi từ thành phần này tới thành phần khác trong ứng dụng bậc *n* như các chuỗi thu thập trên mạng là một hoạt động rất hiệu quả. XML cũng có thể được dùng để gửi dữ liệu trên giao diện HTTP. Nó rất tốt khi bạn cần gửi dữ liệu từ web site này tới web site khác.

XML trong các sản phẩm

XML được dùng nhiều trong các sản phẩm, chẳng hạn như Internet Explorer, SQL Server 7 và 2000. Oracle cũng hỗ trợ XML. Từ ADO 2.5 cũng hỗ trợ các luồng XML và .NET lưu trữ và truyền thông hầu như riêng với XML.

XML so với EDI

EDI (Electronic Data Interchange – Trao đổi dữ liệu điện tử) là một chuẩn đã được thiết kế cách nay nhiều năm để trao đổi dữ liệu giữa các máy tính. Vấn đề ở EDI là nó rất phụ thuộc vào nhà cung cấp và thậm chí phụ thuộc vào cả ứng dụng. Ngoài ra, nó thực sự không được coi là một nền chéo, sử dụng nó cũng nặng nề và rất phức tạp. Nó còn phải chịu lệ phí sử dụng rất mắc của ủy ban tiêu chuẩn EDI. XML có thể thay thế hoàn toàn

EDI khi nó có thể hỗ trợ tất cả các định dạng EDI mà không bị một trong các khuyết điểm này.

Các tài nguyên XML

Có một số tài nguyên trên XML mà bạn nên tìm kiếm trên Internet. Bên dưới là danh sách các tài nguyên và mô tả của chúng.

URL	Mô tả
www.w3c.org/xml	Rất phức tạp, rất "khó chịu".
Http://msdn.microsoft.com/xml	Nhiều công cụ, nhưng rất ít phụ thuộc Microsoft.
www.xml.com	Thông tin chung tốt, các mẫu, các công cụ, các bài báo, vân vân.
www.xml.org	Thông tin chung tốt, các mẫu, các công cụ, các bài báo.

Hiển thị XML

Bằng cách dùng Internet Explorer 5.0, bạn có thể hiển thị tài liệu XML mà không cần phải định dạng và nó sẽ hiển thị bằng cách dùng kiểu hiển thị cây. Có một file mẫu được gọi là **AuthorsNoXSL.xml** được định vị trong folder các mẫu của chương này.

Hình 1: *XML hiển thị bằng IE 4.0 hoặc phiên bản sau như một cấu trúc cây.*

XML của file này trông giống như được trình bày ở bên dưới. Bạn thấy nó giống hệt những gì được trình bày trong IE.

```
<?xml version="1.0"?>
<AuthorsData>
   <Author>
      <au_id>172-32-1176</au_id>
      <au_fname>Johnson</au_fname>
      <city>Menlo Park</city>
      <au_lname>White</au_lname>
   </Author>
   <Author>
      <au_id>213-46-8915</au_id>
      <au_fname>Marjorie</au_fname>
```

```
        <city>Oakland</city>
        <au_lname>Green</au_lname>
    </Author>
...
</AuthorsData>
```

XSL và XML

XSL (eXtensible Stylesheet Language) là ngôn ngữ trông giống như XML và được dùng để biến đổi từ định dạng này sang định dạng khác.

Sử dụng tờ mẫu XSL

Để định dạng XML thành HTML, bạn sẽ dùng tài liệu e**X**tensible **S**tyle **L**anguage (XSL). Ngôn ngữ này trông giống XML, nhưng sẽ xử lý các chỉ lệnh về cách truy tìm các phần của tài liệu XML. Sau đó bạn có thể truy tìm các phần này và đặt chúng vào tài liệu HTML.

***Hình 2:** Sử dụng tờ mẫu XSL mà bạn có thể định dạng XML theo bất kỳ cách nào bạn muốn.*

Để tạo màn hình này trong IE, bạn sử dụng cùng XML bạn đã thấy ở phần trước, nhưng bạn áp dụng tag style sheet (tờ mẫu) XSL ở trên nó.

```
<?xml version='1.0'?>
<?xml:stylesheet type="text/xsl" href="Authors.xsl" ?>
<AuthorsData>
<Author>
<au_id>172-32-1176</au_id>
<au_fname>Johnson</au_fname>
```

```
<city>Menlo Park</city>
<au_lname>White</au_lname>
</Author>
...
```

Style sheet trông giống như sau:

```
<?xml version="1.0"?>
<xsl:stylesheet      xmlns:xsl="http://www.w3.org/TR/WD-
xsl">
  <xsl:template match="/">
    <HTML>
      <BODY>
        <H3>Authors</H3>
        <TABLE BORDER="1">
        <TR>
        <TH>Last Name</TH>
        <TH>First Name</TH>
        <TH>City</TH>
        </TR>
        <xsl:for-each select="AuthorsData/Author">
           <TR>
           <TD><xsl:value-of select="au_lname"/></TD>
           <TD><xsl:value-of select="au_fname"/></TD>
           <TD><xsl:value-of select="city"/></TD>
           </TR>
        </xsl:for-each>
        </TABLE>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```

Tóm tắt

XML là một định dạng rất tốt để diễn tả dữ liệu. Bạn có thể tạo các tag riêng, nó có thể được gửi qua giao thức HTTP và sẽ nhanh chóng trở thành chuẩn để trao đổi dữ liệu. Bạn sẽ cần biết một chút về XML để sử dụng .NET. Nhưng hầu như .NET xử lý tất cả chi tiết cho bạn.

Câu hỏi ôn tập

1. XML viết tắt của các chữ gì?
2. XSL viết tắt của các chữ gì?
3. XML sẽ thay thế chuẩn gì?
4. Schema là gì?

Bài tập

- Tạo tài liệu XML, <Dog>, với các phần tử cho các trường sau: Breed, Color, Height, Weight, Disposition
- Tạo một số dòng dữ liệu cho tài liệu Dog này và thử hiển thị nó bằng IE.
- Tạo một schema cho tài liệu Dog này.

Trả lời câu hỏi ôn tập

1. eXtensible Markup Language.
2. eXtensible Style Language.
3. EDI.
4. Một định nghĩa về cách tài liệu XML sẽ thể hiện.

Chương 23

Xử lý XML

- Tìm hiểu về trình phân tách DOM trong .NET.
- Tìm hiểu về cách nạp tài liệu XML.
- Tìm hiểu cách lựa các phần tử riêng biệt bằng Xpath.
- Tìm hiểu cách đọc nội dung phần tử và các thuộc tính.
- Tìm hiểu cách cập nhật nội dung phần tử và các thuộc tính.

Sử dụng System.XML trong VB.NET

Các tài liệu XML đã trở thành cách rất tốt để chuyển dữ liệu từ ứng dụng này sang ứng dụng khác. Dù truyền thông từ một DLL với DLL khác, từ một EXE này với một EXE khác hoặc thậm chí từ một máy chủ này với một máy chủ khác, XML dễ dàng truyền qua lại một cách đơn giản và hiệu quả. Trong VB6,

bạn sử dụng đối tượng Microsoft.XMLDOM để xử lý các tài liệu XML. Trong .NET, đối tượng tương đương được gọi là System.Xml.XmlDocument. Ở chương này, bạn sẽ tìm hiểu cách làm việc với các tài liệu XML trong .NET Framework.

Trình phân tách XML

Ứng dụng .NET nhập liệu XML từ một trong các nguồn khác nhau. Ứng dụng client truyền các tham số tới máy chủ bằng tài liệu XML. Một máy chủ ứng dụng phân tách và xử lý nội dung của tài liệu XML. Sau khi tiến trình của server hoàn tất, client nhận tài liệu XML.

Trước đây, bạn dựa vào thành phần Microsoft.XMLDOM để xử lý các tài liệu XML. Thành phần này thường được gọi ra từ Active Server Pages, Visual Basic và Internet Explorer. May mắn, .NET Framework cung cấp một đối tượng gốc hiệu suất cao mà bạn có thể sử dụng để phân tách và xử lý các tài liệu XML.

Ở chương này, bạn sẽ tìm hiểu cách nạp tài liệu XML vào .NET và hiển thị nội dung của nó. Bạn sẽ thấy các so sánh giữa System.Xml mới và Microsoft.XMLDOM cũ. Các so sánh này làm cho hiểu rõ về cách XML làm việc trong .NET Framework. Hình 1 trình bày tài liệu XML mẫu mà bạn sẽ dùng ở khắp chương này.


```
- <contact>
  - <name>
      <first>John</first>
      <last>Smith</last>
    </name>
  - <address city="PleasantVille" state="CA" zip="92222">
      <street>1 Main Street</street>
      <street>Apt. #5</street>
    </address>
    <website>www.johnsmith.com</website>
    <email>john@smith.com</email>
    <email>John.Smith@smith.com</email>
  </contact>
```

***Hình 1:** Một tài liệu XML mẫu chứa thông tin Contact có một số phần tử và một số thuộc tính về một trong các phần tử.*

Nạp tài liệu XML

Khi bạn nạp và phân tách tài liệu XML, bạn lập thể nghiệm một thư viện đối tượng biết cách nạp và phân tách tài liệu. Sau khi đối tượng được lập thể nghiệm, bạn có thể cho nó bắt đầu làm việc. Thực hiện các bước sau:

1. Lập thể nghiệm trình phân tách XML.
2. Nạp một tài liệu XML riêng biệt.
3. Hiển thị dữ liệu được định vị trong tài liệu XML.

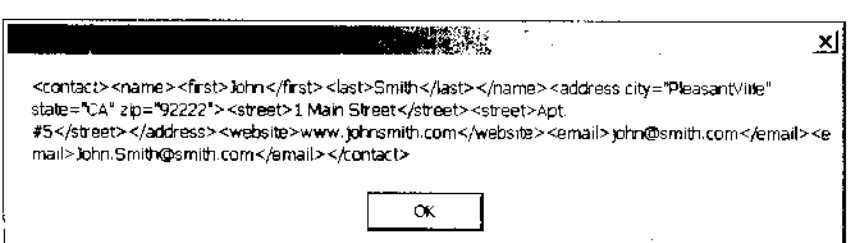
Nạp tài liệu XML

Trong Visual Basic.NET, bạn lập thể nghiệm trình phân tách tài liệu XML được cung cấp với .NET Framework. Các mô hình đối tượng bạn sử dụng ở đây sẽ giống với bất kỳ ngôn ngữ thân thiện .NET nào, gồm cả VB.NET. Ở thí dụ này, bạn lập thể nghiệm đối tượng System.Xml.XmlDocument.

Đối tượng này cung cấp tính năng cơ bản về nạp và phân tách nội dung tài liệu XML. Sau khi đối tượng được lập thể nghiệm, bạn có thể nạp tài liệu XML. Một khi tài liệu được nạp, bạn có thể hiển thị nó với người dùng.

```
Private Sub btnLoad_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnLoad.Click
    Dim xd As System.Xml.XmlDocument

    xd = New System.Xml.XmlDocument()

    xd.Load(GetFileNameAndPath())

    MessageBox.Show(xd.InnerXml)
    MessageBox.Show(xd.InnerText)
End Sub
```

Một trong các thay đổi về cú pháp trong .NET là ở cách bạn tham chiếu nội dung của tài liệu. Thay vì tham chiếu đặc tính XML, bây giờ bạn tham chiếu đặc tính InnerXml, (hình 2).



Hình 2: *Đặc tính XML cho trở lại toàn bộ tài liệu XML.*

Để tham chiếu dữ liệu không có các tag XML, sử dụng đặc tính InnerText (hình 3).

***Hình 3:** Đặc tính InnerText.*

Việc sử dụng đặc tính InnerXml và InnerText giống mô hình đối tượng DHTML trong Internet Explorer. Trong DHTML, bạn truy xuất nội dung HTML của một phần tử bằng đặc tính InnerHTML. Nội dung văn bản của phần tử được tham chiếu bằng đặc tính InnerText.

XPath

XPath cung cấp một ngôn ngữ truy vấn cho các tài liệu XML, giống như SQL (Structure Query Language – Ngôn ngữ truy vấn cấu trúc) cung cấp một ngôn ngữ truy vấn CSDL quan hệ của bạn. Từ vựng XPath cho phép bạn lựa các nút (node) riêng biệt để đáp ứng tiêu chuẩn nhất định. Khi bạn mở một tài liệu XML trong .NET, bạn thường thi hành một truy vấn XPath để truy tìm một tập con các nút (node) được chứa trong tài liệu. Hãy xét một tài liệu chứa một contact client như được trình bày ở listing sau.

```
<contact>
  <name>
    <first>John</first>
    <last>Smith</last>
  </name>
  <address city="PleasantVille" state="CA" zip="92222">
    <street>1 Main Street</street>
    <street>Apt. #5</street>
```

```
    </address>
    <website>www.johnsmith.com</website>
    <email>john@smith.com</email>
    <email>John.Smith@smith.com</email>
</contact>
```

Để truy tìm dữ liệu trong nút Address, bạn có thể chỉ định truy vấn XPath sau.

```
/contact/address
```

Thay vì định vị phần tử bằng một vị trí theo thứ tự, truy vấn XPath có trực giác hơn nhiều. Bằng cách này, mã của bạn vẫn mềm dẻo và dễ dùng như cấu trúc tài liệu XML tiến triển suốt thời gian của ứng dụng. Bạn không muốn mã của bạn phải sửa đổi nếu một phần tử được thêm vào tài liệu trước địa chỉ.

XPath thích hợp với mã của bạn sau khi bạn nạp tài liệu XML. Sau đó bạn có thể áp dụng XPath để truy tìm tập con của các phần tử trong tài liệu. Các phần tử này có thể được xử lý bởi ứng dụng của bạn. Các bước chủ yếu được liệt kê sau.

1. Lập thể nghiệm trình phân tách XML.
2. Nạp tài liệu XML riêng biệt.
3. Định nghĩa XPath.
4. Lựa phần tử khớp với XPath.
5. Xử lý phần tử so khớp.

Sử dụng truy vấn XPath

Tiến trình áp dụng XPath trong VB.NET không có gì khác với kinh nghiệm của bạn trong VB6. Một khi tài liệu XML của bạn được nạp, bạn sẽ có thể truy tìm phần tử con riêng biệt. Phần tử này có thể được áp dụng vào tiến trình của bạn.

```
Private Sub btnSelectSingleNode_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) _
 Handles btnSelectSingleNode.Click
    Dim xd As System.Xml.XmlDocument
    Dim strXPath As String
    Dim oNode As System.Xml.XmlNode

    ' Create new XmlDocument
    xd = New System.Xml.XmlDocument()
    ' Load File
    xd.Load(GetFileNameAndPath())

    strXPath = "/contact/name/first"
    ' Get a Single Node
    oNode = xd.SelectSingleNode(strXPath)

    MessageBox.Show(oNode.InnerXml)
    MessageBox.Show(oNode.OuterXml)
End Sub
```

Sự khác nhau giữa đặc tính XML trong XMLDOM và đặc tính InnerXml trong System.Xml hóa ra rõ ràng khi bạn lựa một nút với XPath. Nội dung của InnerXml chứa nội dung của phần tử mà không có tên tag của phần tử.

```
John
```

Nếu bạn đang tìm cả nội dung và tên tag, bạn sẽ tham chiếu đặc tính OuterXml.

```
<first>John</first>
```

Nếu bạn đã làm việc với DHTML trong Internet Explorer, cú pháp này sẽ giống nhau. Vấn đề tương tự phát sinh khi bạn xây dựng một trang DHTML. Bạn cần chọn tham chiếu riêng nội dung phần tử hoặc phần tử và cả nội dung của nó.

NodeList

Trong các thí dụ trên, bạn đã chỉ truy tìm một nút bằng cách dùng truy vấn XPath. Trường hợp XPath khớp với nhiều nút sẽ giải quyết làm sao? Có thể truy tìm nhiều hơn một nút so khớp. Chỉ cần thay thế phương thức SelectSingleNode() bằng phương thức SelectNodes() khi bạn mong đợi XPath so khớp với hơn một nút. Giá trị trở lại của phương thức SelectNodes() là một NodeList, là một mảng các nút.

Khi bạn áp dụng XPath để cho trở lại hơn một phần tử, bạn sẽ phải tạo một số thay đổi so với mã trước đây. Gọi phương thức SelectNodes() thay vì gọi phương thức SelectSingleNode(). NodeList kết quả phải được vòng qua bằng vòng lặp For. Sau là các bước bạn sẽ thực hiện để xử lý NodeList:

1. Nạp tài liệu XML riêng biệt.
2. Định nghĩa XPath.
3. Lựa các phần tử khớp với XPath.
4. Lặp đi lặp lại qua tất cả các nút so khớp.

Sử dụng lớp NodeList

Làm việc với một NodeList trong System.Xml giống với XMLDOM. Các phương thức gọi đều giống nhau. Các tên đối

tượng cũng đều giống nhau. Nhưng đặc tính bạn sử dụng để xác định số lượng các phần tử so khớp thay đổi.

```
Private Sub btnSelectNodes_Click( _
 ByVal sender As System.Object, _
 ByVal      e      As      System.EventArgs)      Handles
btnSelectNodes.Click
    Dim xd As System.Xml.XmlDocument
    Dim strXPath As String
    Dim oNodeList As System.Xml.XmlNodeList
    Dim intLoop As Integer

    xd = New System.Xml.XmlDocument()
    xd.Load(GetFileNameAndPath())

    strXPath = "/contact/email"

    ' Get Node Collection
    oNodeList = xd.SelectNodes(strXPath)

    MessageBox.Show(oNodeList.Count)

    For intLoop = 0 To oNodeList.Count - 1

MessageBox.Show(oNodeList.Item(intLoop).InnerXml)
    Next
End Sub
```

Đặc tính Count cho bạn biết số lượng các Node so khớp trong NodesList. Nhớ là chỉ mục của mục NodeList bắt đầu từ vị trí zêrô. Điều này có nghĩa là số lượng đó trừ đi một bằng biên trên của mảng.

Khi bạn chạy thí dụ này, bạn sẽ thấy một thông điệp chứa số lượng các nút so khớp. Kế tiếp, vòng lặp For lặp đi lặp lại qua mỗi Node và hiển thị nội dung. Bằng cách này, các ứng dụng của bạn xử lý các Node so khớp của XPath.

Đọc thuộc tính

Các tài liệu XML cấu trúc dữ liệu thành các phần tử và các thuộc tính. Ở hình 1, thành phố, tiểu bang và mã vùng đều có như các thuộc tính của phần tử Address. Ở phần này bạn sẽ tìm hiểu cách đọc các thuộc tính theo lập trình.

Đọc thuộc tính

Các thuộc tính được quản lý trong VB.NET khác với trong VB6. Thay vì xét các phần tử và các thuộc tính khác nhau, System.Xml coi cả hai chúng đều ngang nhau. Bạn lựa một thuộc tính với XPath và nó được cho trở lại như một node.

```
Private Sub btnReadAttribute_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) _
 Handles btnReadAttribute.Click
    Dim xd As System.Xml.XmlDocument
    Dim strXPath As String
    Dim oNode As System.Xml.XmlNode

    xd = New System.Xml.XmlDocument()
    xd.Load(GetFileNameAndPath())

    strXPath = "/contact/address/@city"
    oNode = xd.SelectSingleNode(strXPath)

    MessageBox.Show(oNode.OuterXml)
    MessageBox.Show(oNode.InnerXml)
End Sub
```

Để truy xuất một thuộc tính từ XPath, bạn lập tiền tố tên thuộc tính với dấu @. Nó truyền thông rõ ràng với trình phân tách mà bạn sẽ không tìm tên phần tử mà là tên thuộc tính. Khi bạn lựa nút này, bạn có thể xem InnerXml để tìm giá trị của thuộc tính.

Nếu bạn truy xuất đặc tính OuterXml, bạn nhận tên thuộc tính và giá trị thuộc tính cùng với nhau. Nói chung, bạn sẽ không cần kết hợp thông tin này trong khi xử lý. Hầu như bạn thường sử dụng đặc tính InnerXml của một nút chứa thuộc tính.

Cập nhật phần tử

Các tài liệu XML đều có thể cập nhật. Ứng dụng .NET sửa đổi nội dung của một tài liệu XML. Thông tin mới có thể xuất phát từ form giao diện người dùng hoặc thiết bị khác. Ở phần này bạn sẽ tìm hiểu cách cập nhật giá trị của một phần tử.

Tiến trình sửa đổi một phần tử với System.Xml giống như XMLDOM, nhưng các tên đặc tính khác nhau. Thay vì tham chiếu đặc tính Text, bạn sẽ tham chiếu đặc tính InnerText.

```
Private Sub btnUpdateElement_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) _
 Handles btnUpdateElement.Click
    Dim xd As System.Xml.XmlDocument
    Dim strXPath As String
    Dim oNode As System.Xml.XmlNode

    xd = New System.Xml.XmlDocument()
    xd.Load(GetFileNameAndPath())
    MessageBox.Show(xd.InnerXml)

    strXPath = "/contact/name/first"
    oNode = xd.SelectSingleNode(strXPath)

    MessageBox.Show(oNode.OuterXml)

    oNode.InnerText = "Roberto"

    MessageBox.Show(oNode.OuterXml)
    MessageBox.Show(xd.InnerXml)
End Sub
```

Bằng cách này, bạn có thể sửa đổi nội dung của một phần tử bất kỳ trên tài liệu. Một ứng dụng có thể nhận một XML Document như dữ liệu nhập. Tiến trình có thể sửa đổi một hoặc nhiều phần tử. Tài liệu đã cập nhật có thể được cho trở lại client hoặc được chuyển tới một máy chủ khác.

Cập nhật thuộc tính

Các tài liệu XML cấu trúc dữ liệu thành các phần tử và các thuộc tính. Bạn đã tìm hiểu cách cập nhật nội dung phần tử. Ở phần này, bạn sẽ tìm hiểu cách cập nhật nội dung của một thuộc tính.

Nhớ là tiến trình đọc một thuộc tính với System.Xml khác với đọc một thuộc tính với XMLDOM. Tiến trình cập nhật giá trị của một thuộc tính cũng hơi khác. Sau khi bạn lựa một thuộc tính như một nút với XPath, bạn sửa đổi đặc tính InnerText của nút.

```
Private Sub btnUpdateAttribute_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) _
 Handles btnUpdateAttribute.Click
    Dim xd As System.Xml.XmlDocument
    Dim strXPath As String
    Dim oNode As System.Xml.XmlNode

    xd = New System.Xml.XmlDocument()
    xd.Load(GetFileNameAndPath())
    MessageBox.Show(xd.InnerXml)

    strXPath = "/contact/address/@city"
    oNode = xd.SelectSingleNode(strXPath)
    MessageBox.Show(oNode.OuterXml)

    oNode.InnerText = "Lake Forest"

    MessageBox.Show(oNode.OuterXml)
    MessageBox.Show(xd.InnerXml)
End Sub
```

Lưu ý, thuộc tính được quản lý như một nút giống như một phần tử được quản lý. Một khi bạn định vị thuộc tính với XPath, bạn có thể hiển thị nó hoặc sửa đổi giá trị của nó. Để xác định ứng xử chính xác của thí dụ này, bạn sẽ thấy bốn cửa sổ hộp thông điệp hiển thị tiến trình của mỗi bước tiến hành.

XMLTextReader/XMLTextWriter

Thay vì lớp XMLDocument, bạn cũng có thể sử dụng lớp XMLTextReader và XMLTextWriter để đọc và ghi XML. Hai lớp này rất tốt khi tài liệu XML lớn hơn và không thích hợp với bộ nhớ. Đối tượng XMLTextReader sẽ đọc mỗi lần một khối lượng bộ đệm vào bộ nhớ để bạn có thể xử lý XML mỗi lần một ít.

Đọc XML bằng cách dùng XMLTextReader

Đọc XML vào bộ nhớ, đầu tiên bạn tạo đối tượng XMLTextReader mới bằng cách chuyển vào một chuỗi URL hoặc tên file. Sau đó bạn có thể dùng phương thức Read để đọc mỗi lần một Node. Khi bạn lặp qua nút, bạn có thể xác định kiểu nút, dù nó có các thuộc tính hay không và bạn có thể đọc văn bản từ mỗi nút.

```
Const conURL As String = _
 "http://localhost/PKXMLReaderApp/Contact.xml"

Private Sub btnRead_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnRead.Click
    Dim oXML As XmlTextReader
    Dim strXML As String

    Try
        'oXML = New XmlTextReader(conURL)
        oXML = New XmlTextReader(GetFileNameAndPath())
```

```
        Do While oXML.Read()
            Select Case oXML.NodeType
                'Display beginning of element
                Case XmlNodeType.Element
                    strXML &= "<" + oXML.Name
                    If oXML.HasAttributes Then
                        While
oXML.MoveToNextAttribute()
                            ' Display attribute name
                            ' and value.
                            strXML &= String.Format( _
                             " {0}='{1}'", _
                             oXML.Name, oXML.Value)
                        End While
                    End If
                    strXML &= ">"

                'Display the text in each element
                Case XmlNodeType.Text
                    strXML &= oXML.Value

                'Display end of element
                Case XmlNodeType.EndElement
                    strXML &= "</" + oXML.Name
                    strXML &= ">"

            End Select
        Loop
        txtXML.Text = strXML

    Catch oException As Exception
        MessageBox.Show(oException.Message)

    End Try
End Sub
```

Ghi XML bằng cách dùng XMLTextWriter

Để ghi ra toàn bộ tài liệu XML bằng cách dùng lớp XMLTextWriter, bạn có thể sử dụng phương thức WriteString như được trình bày ở mã bên dưới. Bạn cũng có thể ghi từng phần tử bằng cách dùng các phương thức Write* khác.

```
Private Sub btnWrite_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnWrite.Click
    Dim oXML As XmlTextWriter

    oXML = New XmlTextWriter(GetPath() & "Test.xml", _
     System.Text.Encoding.Unicode)

    oXML.WriteString(txtXML.Text)
End Sub
```

DataSet và XML

Lớp DataSet lưu trữ tất cả dữ liệu được gửi tới nó như một tài liệu XML. Nó cũng có các phương thức vừa đọc lẫn ghi XML vào một file.

Ghi vào file

Để ghi vào một file, đầu tiên bạn phải cư trú đối tượng DataSet bằng cách dùng bất kỳ các phương tiện nào bạn muốn. Chẳng hạn, bạn có thể dùng đối tượng OleDbDataAdapter để nhận dữ liệu từ SQL Server. Một khi dữ liệu ở trong đối tượng DataSet bạn có thể sử dụng phương thức WriteXml để ghi tài liệu này vào file như được trình bày ở mã bên dưới. Bạn cũng có thể cho nó phát sinh schema cho bảng này bằng cách dùng phương thức WriteXmlSchema.

```
Private Sub btnWrite_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnWrite.Click
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oDS As DataSet
    Dim strSQL As String
    Dim strConn As String
```

```
    Dim intLoop As Integer

    Me.Cursor = Cursors.WaitCursor

    ' Get Connection String
    strConn = ConnectStringBuild()

    ' Build SQL String
    strSQL = "SELECT * "
    strSQL &= "FROM Products"

    oDS = New DataSet()
    Try
        ' Create New Data Adapter
        oAdapter = _
         New OleDb.OleDbDataAdapter(strSQL, strConn)
        ' Fill Data Set From Adapter
        ' and assign table name
        oAdapter.Fill(oDS, "Products")

        ' Fill Grid
        grdProducts.PreferredColumnWidth = _
         DataGrid.AutoColumnSize
        grdProducts.DataSource = oDS.Tables("Products")
        ' Put data into controls
        txtXML.Text = oDS.GetXml()
        txtSchema.Text = oDS.GetXmlSchema()

        ' Write Data to a File
        oDS.WriteXml(GetPath() & "Products.xml")
        oDS.WriteXmlSchema(GetPath() & _
         "ProductsSchema.xml")

    Catch oException As Exception
        MessageBox.Show(oException.Message)

    End Try

    Me.Cursor = Cursors.Default
End Sub
```

Đọc từ một file

Bạn có thể xây dựng lại DataSet bằng cách đọc ở một tài liệu XML bằng cách dùng phương thức ReadXmlSchema và ReadXML.

```
Private Sub btnRead_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRead.Click
    Dim oDS As DataSet

    Me.Cursor = Cursors.WaitCursor

    oDS = New DataSet()

    ' Read data from file, get schema FIRST!
    oDS.ReadXmlSchema(GetPath() & "ProductsSchema.xml")
    oDS.ReadXml(GetPath() & "Products.xml")

    ' Fill Grid
    grdProducts.DataSource = Nothing
    grdProducts.PreferredColumnWidth = _
     DataGrid.AutoColumnSize
    grdProducts.DataSource = oDS.Tables("Products")

    ' Put data into controls
    txtXML.Text = oDS.GetXml()
    txtSchema.Text = oDS.GetXmlSchema()

    Me.Cursor = Cursors.Default
End Sub
```

Để truy tìm XML như một chuỗi từ DataSet bạn sử dụng phương thức GetXml và GetXmlSchema.

Sử dụng lớp StringReader

Ở một điểm nào đó, bạn có thể phải tạo một DataSet từ một chuỗi XML mà bạn đã nạp vào bộ nhớ. Nếu bạn phải xét tất cả các phương thức quá tải cho phương thức ReadXML trên đối

tượng DataSet, bạn sẽ không thấy, mà nó chỉ chấp nhận chuỗi. Tất cả chúng đòi hỏi sự sắp xếp đối tượng nào đó phải có tên file văn bản hoặc một tên file. Tuy nhiên, nếu bạn thực sự tìm ra bạn sẽ thấy một trong số chúng là TextReader. Hơn nữa, TextReader đọc từ đĩa, nhưng một trong các lớp thừa kế từ TextReader là StringReader. Đây là lớp bạn cần sử dụng để thực hiện điều này.

```
Private Sub btnLoad_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnLoad.Click
    Dim ds As DataSet
    Dim sr As System.IO.StringReader
    Dim strXML As String

    strXML &= "<NewDataSet>"
    strXML &= "<Products>"
    strXML &= "<ProductID>1</ProductID>"
    strXML &= "<ProductName>Chai</ProductName>"
    strXML &= "<SupplierID>1</SupplierID>"
    strXML &= "<CategoryID>1</CategoryID>"
    strXML &= "<QuantityPerUnit>10 boxes x 20
     bags</QuantityPerUnit>"
    strXML &= "<UnitPrice>18</UnitPrice>"
    strXML &= "<UnitsInStock>39</UnitsInStock>"
    strXML &= "<UnitsOnOrder>0</UnitsOnOrder>"
    strXML &= "<ReorderLevel>10</ReorderLevel>"
    strXML &= "<Discontinued>false</Discontinued>"
    strXML &= "</Products>"
    strXML &= "<Products>"
    strXML &= "<ProductID>2</ProductID>"
    strXML &= "<ProductName>Chang</ProductName>"
    strXML &= "<SupplierID>1</SupplierID>"
    strXML &= "<CategoryID>1</CategoryID>"
    strXML &= "<QuantityPerUnit>24 - 12 oz
     bottles</QuantityPerUnit>"
    strXML &= "<UnitPrice>19</UnitPrice>"
    strXML &= "<UnitsInStock>17</UnitsInStock>"
    strXML &= "<UnitsOnOrder>40</UnitsOnOrder>"
    strXML &= "<ReorderLevel>25</ReorderLevel>"
    strXML &= "<Discontinued>false</Discontinued>"
    strXML &= "</Products>"
    strXML &= "</NewDataSet>"

    sr = New System.IO.StringReader(strXML)
```

```
        ds = New DataSet()
        ds.ReadXml(sr)

        grdProducts.DataSource = ds.Tables("Products")
    End Sub
```

Tóm tắt

.NET Framework cung cấp việc phân tách XML gốc bằng hợp ngữ System.Xml và bằng ADO.NET DataSet. Bạn có thể nạp, lựa và sửa đổi nội dung của một tài liệu XML. Bạn sử dụng đối tượng nào tùy thuộc vào độ lớn của tài liệu XML và nơi XML sẽ phát xuất và nơi nó sẽ tới.

Câu hỏi ôn tập

1. Bạn dùng lớp nào để đọc toàn bộ tài liệu cùng một lần vào bộ nhớ?

2. Bạn dùng lớp nào để nạp tài liệu mỗi lần bằng các chunk (đoạn)?

3. Bạn dùng phương thức nào để truy tìm XML từ DataSet?

4. Bạn dùng phương thức nào để truy tìm XML Schema từ DataSet?

5. Bạn dùng lớp nào để đặt chuỗi XML mà có trong bộ nhớ vào DataSet?

Bài tập

Sử dụng tài liệu Dog bạn đã tạo ở chương trước, viết mã bằng cách dùng bất kỳ kĩ thuật nào ở chương này để đọc XML và đặt nó vào hộp văn bản trên một form.

Trả lời câu hỏi ôn tập

1. System.XML.XMLDocument.
2. XMLTextReader.
3. GetXml.
4. GetXmlSchema.
5. StringReader.

Chương 24

Gọi các thành phần COM từ .NET

- Tìm hiểu khái niệm về Runtime Callable Wrapper.
- Sử dụng TLBIMP để tạo một Assembly từ một thành phần COM.
- Tham chiếu một thành phần COM trực tiếp từ mã VB.NET.

Di trú một thành phần VB6 COM

Bạn có thể đã từng viết các ứng dụng Visual Basic. Nếu bạn giống như nhiều người phát triển, bạn sẽ lập một bản kiểm kê mã có giá trị vào thời gian đó. Và nếu bạn tiếp tục theo các đề nghị của các chuyên gia về các ngôn ngữ khác nhau, thì mã đó được thành phần hóa (*componentized*). Tức là bằng cách dùng server COM (Component Object Model) — trước đây là ActiveX — để bạn tách ứng dụng thành các chunk của tính năng có thể gọi được.

Khi bạn sẵn sàng chuyển sang .NET, bạn sẽ thấy việc thành phần hóa thực sự mang lại hiệu quả. Có lẽ theo một cách lý tưởng, bạn chỉ cần chuyển đổi và di trú tất cả các thành phần sẵn có sang .NET ngay, rồi sau đó lựa sự phát triển theo mô hình mới. Trong thực tế, mọi thứ sẽ phức tạp hơn thế:

- Bạn sẽ không biết mọi thứ về .NET ngay và sẽ mất thời gian tìm hiểu về các khái niệm lập trình .NET và thực hiện chúng, vì vậy có thể phải tiếp tục phát triển trong COM.

- Mặc dù mã của Visual Basic 6 sẽ di trú sang .NET, nhưng sự di trú không đầy đủ. Có thể bạn có các thành phần không thể di chuyển sang được .NET do khuyết điểm của sự thực hiện hoặc ngôn ngữ.

- Mã bạn di trú sang .NET không thể di trú được tất cả ngay một lần. Bạn sẽ phải di trú và kiểm tra từng thành phần đã di trú.

- Bạn có thể sử dụng các thành phần COM bên thứ ba mà bạn không có mã nguồn.

Hầu hết các người phát triển sẽ chuyển tiếp từ thế giới COM sang thế giới .NET trong một khoảng thời gian hàng tháng hoặc hàng năm. Vì vậy việc thành phần hóa giúp bạn về điều này như thế nào? Có hai khái niệm quan trọng làm cho việc di trú chậm và thận trọng dễ dàng hơn:

- Các thành phần .NET có thể gọi các thành phần COM.

- Các thành phần COM có thể gọi các thành phần .NET.

Sự tương tác hai chiều này là chìa khóa để di trú một ứng dụng được thành phần hóa. Bạn có thể bắt đầu với một thành phần bất kỳ trong ứng dụng của mình và vẫn duy trì các kết nối của nó với các thành phần khác sau khi nó đã được chuyển từ mã COM sang mã .NET.

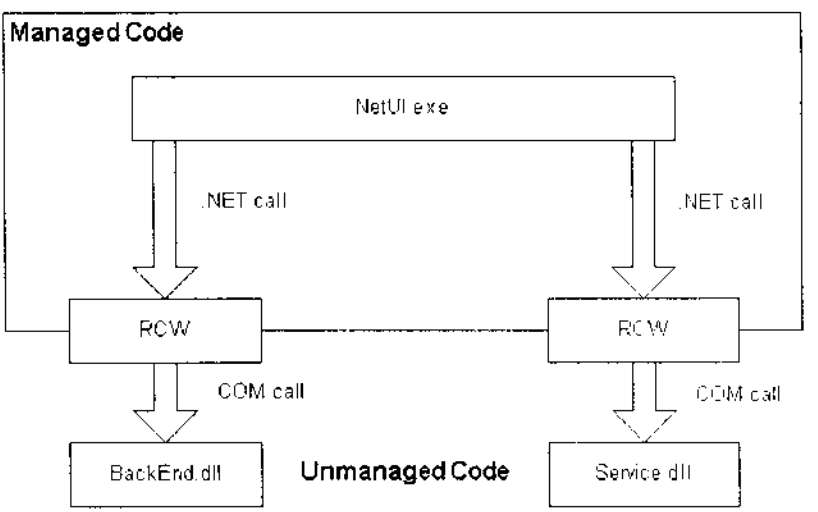
Ở chương này, bạn sẽ tìm hiểu các chi tiết về việc gọi các server COM từ các client .NET. Ở phần "Gọi một .NET Component từ một COM Component," bạn sẽ tìm hiểu về cách gọi theo hướng ngược lại, từ các client COM sang các server .NET.

Unmanaged code và các wrapper có thể gọi vào thời gian chạy

Mã điều tác bên trong .NET Common Language Runtime (CLR) được gọi là *managed code*. Mã này truy xuất tất cả các dịch vụ mà CLR đưa vào bảng, chẳng hạn như tích hợp ngôn ngữ chéo (cross-language integration), bảo mật (security) và hỗ trợ xác định phiên bản, thu thập dữ liệu không thích hợp (garbage collection). Mã không điều tác bên trong CLR được gọi là *unmanaged code*. Theo định nghĩa, tất cả các thành phần COM kế thừa của bạn là unmanaged code. Vì COM được thiết kế trước khi có CLR và mã COM không điều tác bên trong framework được CLR cung cấp, nên nó không sửa đổi được bất kỳ các dịch vụ CLR nào.

Vấn đề pha trộn managed code và unmanaged code trong một ứng dụng là unmanaged code không được nhận biết trong môi trường CLR. Các thành phần managed code không chỉ phụ thuộc vào CLR, chúng còn mong đợi các thành phần khác mà chúng tương tác phụ thuộc vào CLR.

Lối thoát cho tình trạng khó xử này là sử dụng *proxy*. Theo thuật ngữ chung, proxy là một bộ phận của phần mềm nhận các lệnh từ một thành phần, sửa đổi chúng và gửi chúng tới một thành phần khác. Kiểu proxy đặc biệt được dùng khi gọi unmanaged code từ managed code được gọi là Runtime-Callable Wrapper (RCW). Hình 1 trình bày giản đồ về cách các RCW định biên giữa managed code và unmanaged code. Hình này bao gồm một chương trình .NET có tên NetUI.exe, hai thành phần COM có tên BackEnd.dll và Service.dll, và công nghệ cần thiết kết nối chúng.



Hình 1: *Gọi unmanaged code bằng RCW.*

Chuyển đổi Metadata bằng TLBIMP

Đương nhiên, không phải .NET sử dụng metadata đầu tiên để mô tả các giao diện công cộng. Thực ra, các thư viện kiểu COM cũng chứa metadata để mô tả giao diện công cộng với các thành

phần COM. Công việc của RCW là chuyển đổi metadata COM này thành metadata .NET.

Một công cụ để thực hiện việc chuyển đổi này được gọi là tlbimp (type library importer) và nó được cung cấp như một thành phần của .NET Framework SDK (Software Developer Kit). Tlbimp đọc metadata từ thư viện kiểu COM và tạo hợp ngữ CLR so khớp để gọi thành phần COM.

Mẹo:

Nếu bạn xây dựng một ActiveX DLL hoặc ActiveX EXE bằng Visual Basic, thư viện kiểu được nhúng vào trong file DLL hoặc EXE.

Tlbimp là công cụ dòng lệnh có một số tùy chọn cho các thứ như báo hợp ngữ kết quả hoặc phân giải các tham chiếu bên ngoài trong thư viện kiểu. (Gõ **tlbimp /?** tại dấu nhắc để xem tất cả các tùy chọn). Tùy chọn quan trọng nhất là **/out**, cho bạn chỉ định tên cho hợp ngữ .NET kết quả. Thí dụ, để chuyển đổi một ActiveX DLL của Visual Basic có tên support.dll sang hợp ngữ .NET so khớp với tên NETsupport.dll, bạn có thể sử dụng dòng lệnh sau:

```
tlbimp support.dll /out:NETsupport.dll
```

Sử dụng các thành phần COM trực tiếp

Là một người phát triển VB.NET, bạn còn có tùy chọn sử dụng các thành COM trực tiếp. Ít nhất là những gì nó thể hiện, mặc dù về mặt lập trình bạn vẫn sử dụng RCW để nhận các đối tượng trong unmanaged code. Nếu bạn làm việc trong một project

VB.NET, bạn có thể thực hiện các bước sau để thêm tham chiếu với thành phần COM:

1. Chọn Project ➢ Add Reference.
2. Lựa tab COM trong hộp thoại Add Reference.
3. Lựa thư viện kiểu bạn muốn dùng từ danh sách và nhấp Select hoặc sử dụng nút Browse để định vị thành phần không được liệt kê. Các thành phần đã lựa sẽ được thêm vào khung xem danh sách bên dưới trong hộp thoại.
4. Nhấp OK để tạo các RCW cho các thư viện kiểu đã lựa trong project VB.NET của bạn.

Khi bạn thực hiện điều này, bạn sẽ thấy VB.NET thực sự tạo DLL trong folder /Bin của project với tên dẫn xuất từ tên thành phần COM gốc. Thí dụ, nếu bạn tham chiếu phiên bản BackEnd.dll 2.0 bằng cách này, VB.NET sẽ tạo RCW trong file Interop.BackEnd_2_0.dll.

Chọn giữa TLBIMP và Direct Reference

Mặc dù cả hai phương thức sử dụng thành phần COM đều làm việc tốt, nhưng có một số lý do nên chọn phương thức này hơn phương thức kia:

- Để thành phần COM sẽ chỉ được dùng trong một project VB.NET, sử dụng tham chiếu trực tiếp thay vì thực hiện thêm bất kỳ công việc nào khác.

- Nếu thành phần COM được dùng chung giữa nhiều project, sử dụng tlbimp để bạn có thể tạo RCW với một tên riêng biệt ở một vị trí nhất định.

- Nếu bạn cần điều khiển các chi tiết cao cấp của Assembly đã tạo, chẳng hạn như Namespace, số hiệu phiên bản hoặc thông tin đặc trưng, bạn phải sử dụng tlbimp. Phương thức tham chiếu trực tiếp không cho bạn điều khiển các chi tiết này.

Sử dụng thành phần COM của .NET

Ở thí dụ sau, bạn sẽ dùng phương thức của thành phần VB6 COM từ bên trong một project .NET. Bạn sẽ dùng tlbimp để chuyển đổi giao diện của thành phần COM thành một hợp ngữ và sau đó sẽ thấy cách bạn có thể xử lý nó như bất kỳ thành phần .NET nào khác từ bên trong managed code. Sau khi bạn thực hiện xong điều này, bạn cũng thấy cách đi theo một lối tắt bằng cách dùng thành phần COM trực tiếp từ một project Visual Basic.NET mà không dùng tlbimp.

Đăng ký thành phần COM

Thành phần COM bạn sẽ dùng trong bài tập này được đặt tên là VB6SQL.DLL. Nó là một ActiveX DLL chứa một lớp có tên Products. Lớp này sử dụng ADO 2.6 và mệnh đề FOR XML của SQL 2000 để truy tìm dữ liệu ở định dạng giống XML của ADO.NET DataTable. Lớp Products cung cấp một phương thức đơn giản, GetDataTableXML, để truy tìm một chuỗi chứa XML xuất liệu.

Lớp mẫu Products khai thác các khả năng được thêm vào ADO 2.6, cho phép bạn gửi xuất liệu của phương thức Execute của đối tượng Command vào đối tượng Stream và sau đó đọc dữ liệu từ luồng đó bằng cách dùng phương thức ReadText. (Nếu bạn quan tâm, đăng ký project VB6 mẫu. Mã đó không phải là điểm trọng tâm của minh họa này).

Để đăng ký thành phần COM, bạn có thể sử dụng một trong hai tùy chọn sau:

- Xây dựng lại DLL bằng cách dùng VB6 trên máy cục bộ của bạn hoặc
- Sử dụng tiện ích dòng lệnh REGSVR32 để đăng ký VB6SQL.DLL từ cửa sổ Command hoặc Start|mục menu Run (thay thế DLLPath bằng vị trí của DLL trên máy tính của bạn):

```
regsvr32 c:\DLLPath\VB6SQL.dll
```

Bạn sẽ thấy một thông điệp được trình bày ở hình 2.

Hình 2: *Thông điệp về sự cài đặt thành công thành phần COM kế thừa.*

Sử dụng TLBIMP để tạo hợp ngữ

Kế tiếp, sử dụng tiện ích tlbimp để tạo một hợp ngữ để cung cấp RCW cho thành phần VB6SQL:

1. Mở cửa sổ dấu nhắc lệnh của Visual Studio.NET bằng cách lựa Start | Programs | Microsoft Visual Studio.NET 7.0 | Visual Studio.NET Tools | Visual Studio.NET Command Prompt.

2. Đổi sang folder chứa VB6SQL DLL.

3. Tại dấu nhắc lệnh, gõ:

```
tlbimp VB6SQL.dll /out:NETVB6SQL.dll
```

Tlbimp sẽ thực hiện công việc chuyển đổi và sau đó in thông điệp xác định thư viện kiểu đã được nhập là tên mới.

Mẹo:

Visual Studio.NET Command Prompt thêm folder /Bin/ của Framework SDK vào đường dẫn để các công cụ SDK, chẳng hạn như tlbimp, có thể được gọi ra mà không phải chỉ định đường dẫn của chúng.

Tạo project kiểm tra

Bây giờ bạn có thể kiểm tra khả năng của ứng dụng Visual Basic.NET để sử dụng một đối tượng từ một thành phần COM di sản (legacy).

1. Mở Visual Studio.Net và chọn New Project từ Start Page.

2. Lựa Visual Basic Project từ khung xem cây ở bên trái của màn hình.

3. Lựa Windows Application như template project.

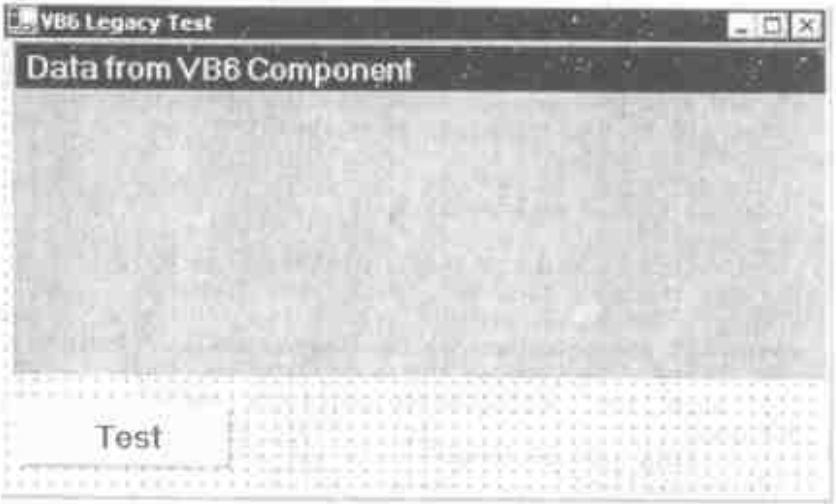
4. Đặt tên của ứng dụng là **VB6SQLTest** và nhấp OK để tạo project.

5. Tô sáng form có tên Form1.vb trong cửa sổ Solution Explorer và đặt lại tên nó là **frmSQLTest.vb**. Xác lập đặc tính **Text** của form là **"VB6 Legacy Test"**.

6. Tạo form được trình bày ở hình 3 bằng cách thêm các control thích hợp và xác lập các đặc tính của các control đó như được trình bày ở bảng 1.

Kiểu Control	Đặc tính	Giá trị
Button	Name	BtnTest
	Text	Test
	Anchor	Bottom, Left
DataGrid	Name	grdDemo
	CaptionText	Data from VB6 Component
	Anchor	Top, Bottom, Left, Right

Bảng 1: *Các control cho frmSQLTest.vb.*



Hình 3: *Thiết kế form kiểm tra.*

7. Để sử dụng RCW cho thành phần VB6SQL COM, chọn Project|Add Reference. Phải chắc chắn tab .NET được lựa và nhấp Browse.

8. Duyệt folder Legacy và lựa thành phần NETVB6SQL.dll. Nhấp Open. Điều này sẽ thêm thành phần ấy vào danh sách Selected Components.

9. Nhấp OK. Nút References trong Solution Explorer sẽ mở rộng để trình bày tham chiếu netvbsql.

Thêm mã để sử dụng thành phần COM

Bây giờ bạn sẵn sàng viết mã sử dụng các thành viên của lớp VB6SQL.

1. Nhấp đôi btnTest và sửa đổi thủ tục btnTest_Click sao cho nó thể hiện như sau:

```
Private Sub btnTest_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnTest.Click
  Dim ovb As New netvb6sql.Products()

  Dim dt As DataTable
  Dim ds As New DataSet()

  ' Use StringReader object to retrieve data.
  ' Get the data from COM, in DataSet format.
  Dim sr As New
System.IO.StringReader(ovb.GetDataTableXML)

  ' Send the StringReader object's data into the
dataset,
  ' filling it with data.
  ds.ReadXml(sr)

  grdDemo.PreferredColumnWidth =
DataGrid.AutoColumnSize
  grdDemo.DataSource = ds.Tables(0)
End Sub
```

Lưu ý, cho mãi tới khi mã này được quan tâm, lớp netvb6sql.Products giống như một lớp .NET bất kỳ khác. Bạn khai báo một thể nghiệm lớp và sử dụng các thành viên của nó như thể nó là một lớp gốc thay vì là một Runtime-Callable Wrapper bao quanh thành phần COM.

Đưa XML vào đối tượng DataTable

Thành phần VB6 di sản cho trở lại dòng truyền XML, được định dạng như thể nó là một đối tượng DataSet đã nối tiếp hóa. Mục đích ở đây là nạp đối tượng DataTable với dữ liệu đó để bạn có thể liên kết khung lưới với dữ liệu bạn dữ liệu bạn đã nạp. Như bạn thấy, việc đưa dòng truyền XML vào đối tượng DataTable không phải là một tác vụ đơn giản. Đầu tiên mã mẫu tạo đối tượng StringReader và nạp giá trị trở lại từ lời gọi phương thức của thành phần COM vào đối tượng đó. Kế tiếp, mã gọi phương thức ReadXML của đối tượng DataSet, chuyển vào đối

tượng StringReader bạn vừa điền vào. Tại sao phải đi qua các bước này? Phương thức ReadXML điền đối tượng DataSet với XML, tạo giản đồ cho các bảng dữ liệu khác nhau bên trong dataset cho bạn. Tuy nhiên, phương thức ReadXML đòi hỏi như tham số của nó một đối tượng nào đó thừa kế từ đối tượng Stream và StringReader thoả mãn đòi hỏi đó, cho phép bạn điền nó với dữ liệu từ một chuỗi. Một khi bạn nạp XML vào đối tượng DataSet, nó là một tác vụ đơn giản để truy tìm DataTable bạn cần và xác lập nó như nguồn dữ liệu của khung lưới.

Để thấy lớp Products hoạt động, thực hiện các bước sau:

1. Press F5 to start the project.

2. Nhấp Test để lập thể nghiệm đối tượng Products và nạp khung lưới với dữ liệu.

3. Đóng form để dừng project.

Sử dụng thành phần COM trực tiếp

Bây giờ bạn đã hiểu cách sử dụng tlbimp, thử một tiến trình thay thế bằng cách dùng thành phần COM trực tiếp.

1. Mở thể nghiệm thứ hai của Visual Studio.Net và chọn New Project từ Start Page.

2. Lựa Visual Basic Project từ khung xem cây ở bên trái của màn hình.

3. Lựa Windows Application là template project.

4. Đặt tên của ứng dụng là **VB6SQLTest2** và nhấp OK để tạo project.

5. Tô sáng form có tên Form1.vb trong cửa sổ Explorer và đặt lại tên nó là **frmSQLTest2.vb**.

6. Chuyển sang thể nghiệm ban đầu của Visual Studio.Net và mở frmSQLTest.vb ở khung xem thiết kế. Nhấp **CTRL+A**, **CTRL+C** để lựa tất cả các control trên form và chép chúng vào Clipboard.

Lưu ý:

Trong Beta 2, bạn không thể chép các control từ một thể nghiệm của VS.NET này tới một thể nghiệm khác của nó. Trong Beta 2, mở các form trong khung xem Code và chép, rồi dán tất cả mã từ thể nghiệm này sang thể nghiệm khác. Điều này vẫn hoạt động trong bản phát hành cuối cùng, nhưng nó chép các control từ khung xem Design có nhiều ý nghĩa hơn. Tuy nhiên, điều đó không hoạt động trong Beta 2.

7. Chuyển trở lại thể nghiệm thứ hai của Visual Studio.Net, lựa frmSQLTest2 trong khung xem thiết kế và nhấp **CTRL+V** để dán tất cả các control vào form mới.

Để sử dụng thành phần COM trực tiếp, chọn **Project|Add Reference** và lựa tab **COM** trong hộp thoại **Add Reference**. Cuộn xuống danh sách của các thành phần COM cho tới khi bạn thấy – **PK VB6 Legacy**, nhấp Select và sau đó nhấp OK. Bạn sẽ nhận cảnh báo được trình bày ở hình 4.

***Hình 4:** Cảnh báo khi chèn tham chiếu thành phần COM.*

Nhấp Yes để cho Visual Studio.NET biết để phát sinh RCW cho thành phần này. Bạn sẽ thấy VB6SQL xuất hiện trong danh sách của References ở Solution Explorer.

Thêm mã để sử dụng thành phần COM

Bây giờ bạn sẵn sàng viết mã sử dụng các thành viên của lớp Products. Nhấp đôi btnTest và sửa đổi thủ tục btnTest_Click như sau:

```
Private Sub btnTest_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnTest.Click
  Dim ovb As New VB6SQL.Products()

  Dim dt As DataTable
  Dim ds As New DataSet()

  ' Use StringReader object to retrieve data.
  ' Get the data from COM, in DataSet format.
  Dim sr As New _
   System.IO.StringReader(ovb.GetDataTableXML)

  ' Send the StringReader object's data into the
dataset,
  ' filling it with data.
  ds.ReadXml(sr)

  grdDemo.PreferredColumnWidth
DataGrid.AutoColumnSize
  grdDemo.DataSource = ds.Tables(0)
End Sub
```

Mẹo:

Với ngoại lệ của tên thư viện, nó giống như mã bạn đã dùng trong phiên bản đầu tiên của project.

Để xem lớp Products hoạt động, sử dụng RCW được tạo bởi VS.NET, thực hiện các bước sau:

1. Ấn **F5** để bắt đầu project.
2. Nhấp Test để lập thể nghiệm đối tượng Products và nạp khung lưới với dữ liệu.
3. Đóng form để dừng project.

Tóm tắt

Mặc dù việc gọi một thành phần COM từ managed code đơn giản, nhưng bạn không nên xem điều này là một giải pháp thường xuyên cho hầu hết các ứng dụng. Chạy một chương trình thời gian lâu, bạn sẽ muốn di trú nhiều thành phần đã sử dụng tới mã .NET gốc. Điều này sẽ cung cấp cho bạn toàn bộ phạm vi các khả năng của .NET cũng như tạo cho mã của bạn nhanh hơn bằng cách loại bỏ tầng RCW.

Khi bạn bắt đầu di chuyển một ứng dụng phức tạp từ Visual Basic sang VB.NET, nhưng khả năng gọi các thành phần COM từ mã .NET là yếu tố cần thiết. Như bạn thấy, nó cũng dễ thực hiện. Hãy xem kĩ thuật này như một thành phần quan trọng của chiến lược di trú của bạn.

Câu hỏi ôn tập

1. Tại sao bạn cần sử dụng một VB6 DLL sẵn có trong một ứng dụng .NET?

2. Tiện ích tlbimp thực hiện điều gì?

3. Thành phần COM có thể gọi một thành phần .NET không?

Bài tập

- Làm bài tập trong chương này.

Trả lời câu hỏi ôn tập

1. Để bạn không phải viết lại bất kỳ mã nào đã được viết và đã chạy thử trong VB6.

2. Tạo ra một hợp ngữ của DLL để nó thể hiện như một thành phần .NET.

3. Có.

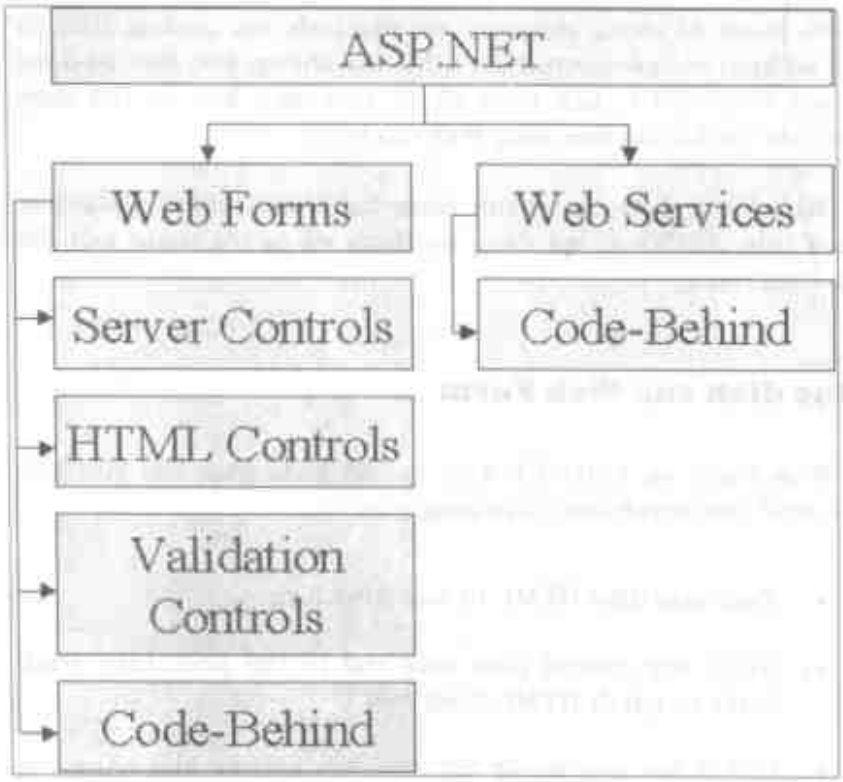
Chương 25

ASP.NET

- Tìm hiểu về các Web Form.
- Tìm hiểu về các Web control được thiết lập vào các Web Form.
- Xây dựng một Web Form.

ASP.NET

ASP.NET là thế hệ kế tiếp của các công cụ phát triển cho các ứng dụng Web. Nó khai thác tất cả ưu điểm của Active Server Pages (ASP), phối hợp nó với kĩ thuật các thành phần COM (Component Object Model – Mô hình đối tượng thành phần) và RAD (Rapid Application Development – Phát triển ứng dụng nhanh) để thiết kế các giao diện người dùng và gói tất cả nó vào một nền. Kết quả là một kinh nghiệm phát triển Web mạnh, phù hợp và nhanh, nó sẽ cho bạn tính năng rất mềm dẻo mà chỉ với một ít mã.



***Hình 1:** ASP.NET có nhiều thành phần.*

ASP.NET chứa các Web Form và các Web Service. Bạn sẽ tìm hiểu một số chúng ở chương này. Mỗi thành phần sẽ được đề cập chi tiết ở các chương kế tiếp.

Web Form

Web Form là trái tim và là linh hồn của ASP.NET. Web Form là phần User Interface (UI) để cho ứng dụng Web của bạn phong cách và cảm giác đặc biệt của chúng. Web Form giống Windows

Form trong đó chúng cung cấp các đặc tính, các phương thức và các sự kiện cho các control được đặt vào chúng. Nếu bạn sử dụng Visual Studio.NET, bạn cũng sẽ có giao diện kéo và thả được dùng để tạo UI của ứng dụng Web của bạn.

Web Form được tạo thành bằng hai thành phần: phần trực quan (file .ASPX) và mã đàng sau form để cư trú trong một file lớp tách riêng.

Mục đích của Web Form

Web Form và ASP.NET được tạo để khắc phục các giới hạn của ASP. Sức mạnh mới của chúng gồm:

- Tách giao diện HTML từ ứng dụng logic.
- Nhiều tập control phía máy chủ có thể phát hiện trình duyệt và gửi đi HTML thích hợp.
- Viết ít mã hơn do dữ liệu liên kết với các khả năng của các control mới của .NET phía máy chủ.
- Mô hình lập trình dựa vào sự kiện giống như lập trình Visual Basic.
- ASP.NET được biên dịch (ASP được diễn dịch) mỗi lần một trang khi chạy.
- Quản lý trạng thái lập sẵn.
- Cho phép các bên thứ ba tạo các control để cung cấp thêm tính năng.

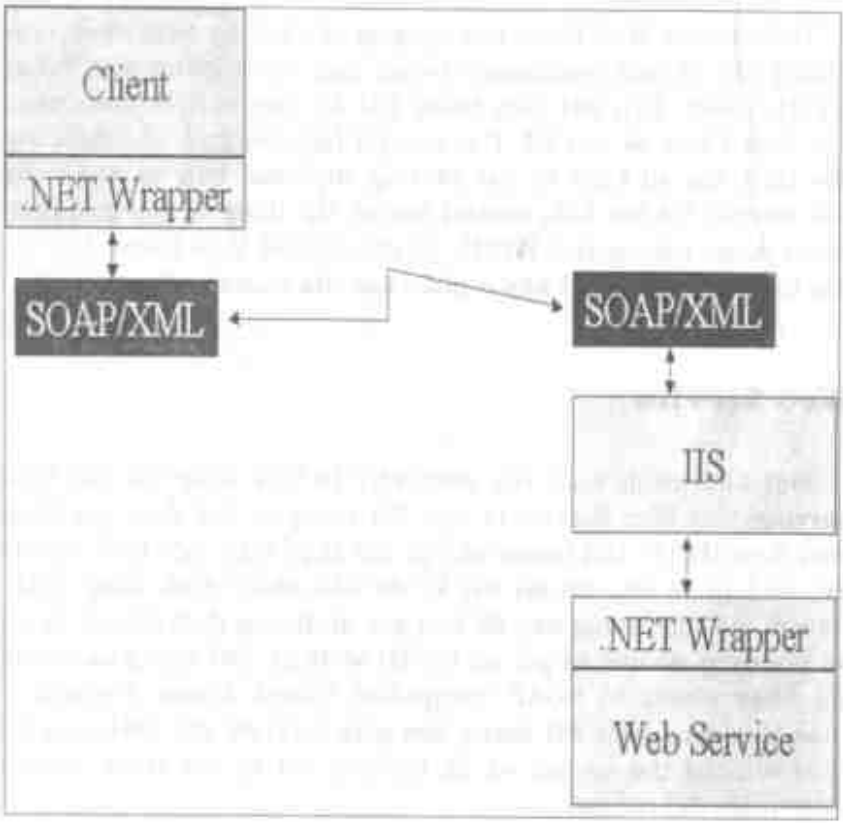
Tự thân các Web Form không cung cấp bất kỳ tính năng nào. Chúng chỉ là một workspace (vùng làm việc) giống một trang HTML trống. Bạn đặt một trong bất kỳ các control khác nhau vào Web Form để tạo UI. Các control bạn dùng sẽ xác định các đặc tính, các sự kiện và các phương thức nào bạn sẽ nhận của mỗi control. Có hai kiểu control bạn có thể dùng để tạo giao diện người dùng: các control HTML và các control Web Form. Bạn sẽ tìm hiểu về các control này ở phần sau của chương này.

Web Service

Một khía cạnh khác của ASP.NET là khả năng tạo các Web Service. Một Web Service là một đối tượng có thể được gọi bằng giao diện HTTP. Đối tượng này có thể thực hiện một dịch vụ bất kỳ cho bạn và cho trở lại bất kỳ dữ liệu nào ở định dạng XML. Bạn sẽ viết đối tượng này để bạn xác định mục đích của nó là gì, nó lấy tham số nào và giá trị trở lại sẽ là gì. Đối tượng này được gọi bằng phong bì SOAP (Simplified Object Access Protocol – Giao thức truy xuất đối tượng đơn giản hóa) để gói XML nào đó chứa tên của thủ tục gọi và dữ liệu của bất kỳ các tham số nào chuyển tới thủ tục này.

Các Web Service có thể được viết bằng một ngôn ngữ .NET bất kỳ hoặc bất kỳ ngôn ngữ nào chạy trên một nền nào đó. Chúng cũng có thể được gọi từ một ngôn ngữ .NET bất kỳ hoặc bất kỳ ngôn ngữ nào có khả năng xử lý các phong bì SOAP và gọi các giao diện HTTP.

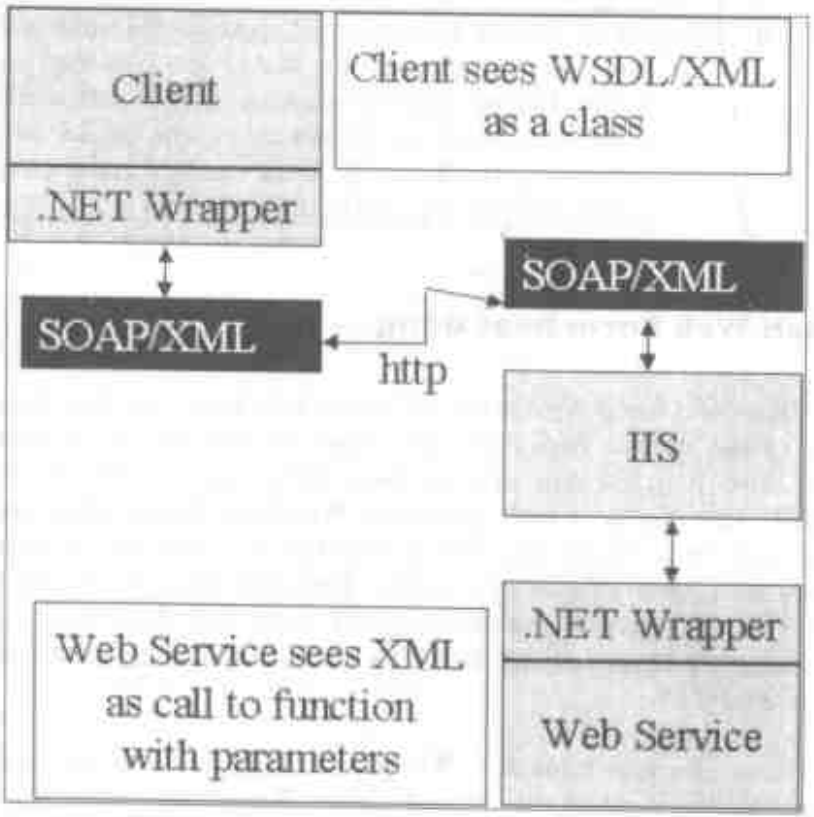
Web Service được tạo trong .NET luôn luôn cho trở lại chuỗi XML. Khi bạn sử dụng .NET Framework, bạn sẽ không cần dùng XML nữa; bất kỳ một giá trị trả lại mà bạn gửi trở lại sẽ tự động được gói vào chuỗi XML cho bạn.



Hình 2: *Cách client nhận và gửi thông tin tới và từ Web Service.*

Các thành phần của Web Service

Khi bạn tạo một Web Service trong .NET, bạn tạo một lớp có các phương thức Public. Mỗi phương thức Public này có thể được trình bày như một Web Service cần được gọi. Trình biên dịch .NET đảm trách việc tạo các file cần thiết để cho phép một tiến trình khác gọi Web Service này bằng giao diện HTTP.



***Hình 3:** .NET ẩn sự thi hành của SOAP.*

Khi bạn biên dịch một Web Service được tạo trong .NET, nó sẽ tự động phát sinh một số file cho bạn. Các file này được dùng để trợ giúp trong việc khám phá các dịch vụ web và mô tả về dịch vụ web.

File	Mô tả
.VSDISCO	File Visual Studio DISCOvery. File này cho một danh sách các Web Service sẵn có trên Web site này. Mỗi Web Service được liệt kê bên trong file SDL.

.WSDL / .SDL	File Service Description Language. File này mô tả (các) tên phương thức sẵn có cho Web Service này. Đối với mỗi phương thức, còn có một danh sách các tham số và mô tả về giá trị trả lại. File này chỉ được xây dựng sau khi triển khai Web Service này vào một Web Server.

Cách Web Form hoạt động

Giống như trong WinForms, có các sự kiện kích hoạt theo một thứ tự nào đó khi Web Form khởi hoạt và nạp. Còn có các sự kiện kích hoạt để đáp ứng các hoạt động xảy ra trên trang HTML. Khi bạn để ý cách một trang Web bình thường được tạo trên một trình duyệt, bạn cho là mọi thứ đều được xử lý theo tuyến tính, theo kiểu từ trên xuống. Trên một trình duyệt, điều này có thể khá chính xác và trong ASP trước đây, điều này hẳn là đúng. Tuy nhiên, đối với một Web Form, không có gì có thể đi xa hơn sự thật.

Giống như một WinForm, Web Form thông qua kiểu sự kiện Load, Draw (Render), và Unload chuẩn. Trong suốt tiến trình này, các thủ tục khác nhau bên trong module lớp đều được gọi. Khi một trang được gọi từ một trình duyệt của client, trang .ASPX được nạp cùng với DLL, nơi module lớp của bạn cư trú. Cả hai đều được biên dịch với nhau và được lưu trữ trong bộ nhớ. Trong giai đoạn này, sự kiện Init và Load kích hoạt. Sự kiện Init được .NET Framework sử dụng bên trong và rất có thể bạn chưa bao giờ thay đổi thủ tục sự kiện này. Sự kiện Load là nơi bạn có thể kiểm tra xem có phải đây là lần đầu trang này được tải hay không hoặc đây là trang người dùng gửi trở lại khi họ nhấp một nút hoặc một control khác trên trang đó. Bạn có thể thực hiện một khởi tạo nào đó trên trang đầu tiên và có thể làm điều gì đó khác nữa nếu nó là post back. Một khi trang đã được biểu hiện đầy đủ, sự kiện Unload được kích hoạt và trang ấy được hủy. Một

số thông tin trạng thái được gửi với trang ấy, vì vậy khi trang được gọi lại bằng một post back hoặc một siêu liên kết thì ASP.NET biết đây là lần thứ hai trang này được gọi.

Các trang Web Form không bị hủy khi chúng gửi xong thông tin của chúng tới trình duyệt, thay vào đó, chúng lập cache trong bộ nhớ. Nếu bạn thay đổi trang .ASPX hoặc nếu bạn thay đổi DLL, trang ấy sẽ bị hủy và phiên bản mới sẽ được tải và được lập cache vào bộ nhớ.

Khi người dùng nhấp một control nút của Web Form để gửi trở lại máy chủ, thì tất cả thông tin của trang ấy được gửi trở lại cùng trang, trang ấy được tạo lại và sự kiện được phát sinh bên trong lớp. Sau đó bạn có thể thực hiện bất kỳ tiến trình nào của bạn và .NET Framework sẽ đảm nhiệm việc gửi trang mới trở lại trình duyệt.

Control Web Form

Có một số kiểu control Web Form khác nhau tạo thành các Web Form. Bạn sẽ tìm hiểu về từng kiểu này ở phần kế tiếp.

Control HTML

Các control HTML đúng là các control HTML thực mà bạn sẽ dùng nếu bạn đang sử dụng Front Page hoặc một trình soạn thảo HTML khác để thiết kế UI. Bạn cũng có thể dùng chúng trong các Web Form. Thí dụ, nếu bạn muốn tạo một hộp văn bản, bạn sẽ viết:

```
<input type="text" id=txtFirstName size=25>
```

Nếu bạn đang dùng Visual Studio.NET, bạn chọn control TextField từ cửa sổ Toolbox và vẽ control nơi bạn muốn nó ở trên trang HTML.

Bạn có thể dùng tất cả các control HTML sẵn có bạn muốn khi sử dụng ASP.NET.

Control	Mô tả	Thí dụ mã HTML
Button	Một control nút bình thường mà bạn có thể dùng để đáp ứng các sự kiện click	<input type=button>
Reset Button	Xác lập lại tất cả các control HTML khác trên form thành một giá trị mặc định	<input type=reset>
Submit Button	Tự động POST dữ liệu form tới trang xác định được liệt kê trong thuộc tính Action= ở tag FORM	<input type=submit>
Text Field	Cho người dùng vùng nhập liệu trên một form HTML	<input type=text>
Text Area	Được dùng cho nhập liệu nhiều dòng trên một form HTML	<input type=textarea>
File Field	Đặt trường văn bản và nút Browse trên form và cho phép người dùng lựa tên file từ máy cục bộ của họ khi nút Browse được nhấp.	<input type=file>
Password Field	Vùng nhập liệu trên form HTML, tuy nhiên, bất kỳ các kí tự nào được gõ vào trường này đều được hiển thị bằng các dấu sao	<input type=password>

Checkbox	Cho người dùng dấu kiểm mà họ có thể lựa hoặc xóa	<input type=checkbox>
Radio Button	Được dùng hai hoặc nhiều nút trên một form và cho phép người dùng chọn một trong các control ấy	<input type=radio>
Table	Cho phép bạn trình bày thông tin ở định dạng cột	<table><tr><td></td></tr></table>
Image	Hiển thị ảnh trên form HTML	<img src ="<FileName">
ListBox	Hiển thị danh sách các mục với người dùng. Bạn có thể xác lập kích thước từ hai trở lên để chỉ định bạn muốn trình bày bao nhiêu mục. Nếu có nhiều mục hơn sẽ không vừa bên trong giới hạn này, một thanh cuộn tự động được thêm vào control này	<select size =2><option></option></select>
Dropdown	Hiển thị danh sách các mục với người dùng, nhưng mỗi lần chỉ một mục duy nhất xuất hiện. Người dùng có thể nhấp mũi tên xuống ở bên cạnh control này và một danh sách các mục sẽ được hiển thị.	<select><option></option></select>
Horizontal Rule	Hiển thị đường kẻ ngang qua trang HTML	<hr>

Bảng 1: *Các control HTML sẵn có trong ASP.NET.*

Tất cả các control này ghi HTML chuẩn vào Web Form. Bạn sẽ gắn thuộc tính ID vào mỗi control để cho phép bạn viết mã JavaScript phía client cho bất kỳ một trong các sự kiện nào thông dụng cho một kiểu control đặc biệt này. Bên dưới là một một phần danh sách của một số các sự kiện phía client thông dụng hơn.

Sự kiện	Mô tả
OnBlur:	Control làm mất tiêu điểm
OnChange:	Các nội dung của control được thay đổi
OnClick:	Control được nhấp
OnFocus:	Control nhận tiêu điểm
OnMouseOver:	Chuột di chuyển vào control này

Các control Web Form

Các control Web Form được tạo và chạy trên Server. Sau khi thực hiện bất kỳ thao tác nào chúng được thiết kế để thực hiện, chúng biểu hiện HTML và gửi HTML đó vào luồng xuất liệu. Luồng xuất liệu này là những gì được gửi như HTML chuẩn tới trình duyệt của người dùng. Các control này không nhất thiết phải ánh xạ tới bất kỳ một control HTML nào. Thay vào đó, chúng có thể là một control lại của một hoặc nhiều control kết hợp với nhau.

Tất cả các control Web Form thừa kế từ một lớp cơ sở chung, có tên là lớp System.Web.UI.WebControls. Lớp cơ sở này thực thi một tập các đặc tính chung mà tất cả các control này sẽ có. Một số các đặc tính chung này gồm:

- BackColor
- Enabled

- Font
- ForeColor
- Modifiers
- TabIndex
- Visible
- Width

Có một số phân loại control khác được .NET Framework cung cấp. Một số control có tương ứng một một với các thành phần tương đương HTML của chúng. Một số control cung cấp thêm thông tin khi được gửi trở lại máy chủ và một số control cho phép bạn hiển thị dữ liệu ở định dạng kiểu danh sách hoặc kiểu bảng. Bảng 2 trình bày danh sách các control phía máy chủ Web Form và các sự kiện phía máy chủ mà bạn có thể đáp ứng với mỗi control.

Control	Mô tả	Sự kiện phía server	Thí dụ mã HTML
Label	Hiển thị văn bản trên trang HTML.	Không có	<asp:Label id="Label1" runat="server">Label</asp:Label>
TextBox	Cho người dùng nhập liệu trên form HTML.	TextChanged	<asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
Button	Control nút nhấn thường được dùng để đáp ứng các sự kiện click trên server. Bạn được phép chuyển thêm thông tin bằng cách các lập đặc tính CommandName và CommandArgum.	Click, Command	<asp:Button id="Button1" runat="server" Text="Button" /></asp:Button>
LinkButton	Giống như một nút trong đó nó gửi trở lại server, nhưng nút lại	Click, Command	<asp:LinkButton id="LinkButton1" runat="server">LinkButton

	trông giống như một siêu liên kết		</asp:LinkButton>
ImageButton	Có thể hiển thị ảnh đồ họa và khi được nhấp, gửi trở lại thông tin lệnh server, chẳng hạn như các tọa độ của chuột bên trong ảnh khi được nhấp.	Click	<asp:ImageButton id=ImageButton1 runat="server"></asp:ImageButton>
Hyperlink	Một control siêu liên kết bình thường đáp ứng sự kiện click	Không có	<asp:HyperLink id=HyperLink1 runat="server">HyperLink </asp:HyperLink>
DropDownList	Một control danh sách thả xuống bình thường như control HTML, nhưng có thể là dữ liệu liên kết với nguồn dữ liệu	SelectedIndexChanged	<asp:DropDownList id=DropDownList1 runat="server"></asp:DropDownList>
ListBox	Control ListBox bình thường giống như control HTML, nhưng có thể là dữ liệu liên kết với nguồn dữ liệu	SelectedIndexChanged	<asp:ListBox id=ListBox1 runat="server"></asp:ListBox>
DataGrid	Giống như <TABLE> trên các steroid. Bạn liên kết nguồn dữ liệu với control này và nó hiển thị tất cả thông tin cột. Bạn cũng có thể thực hiện lập trang, sắp xếp và định dạng rất dễ dàng với control này.	CancelCommand, EditCommand, DeleteCommand, ItemCommand, SelectedIndexChanged, PageIndexChanged, SortCommand, UpdateCommand	<asp:DataGrid id=DataGrid1 runat="server"></asp:DataGrid>
DataList	Cho phép bạn tạo kiểu định dạng khác bảng cho dữ liệu. Bạn có thể liên kết dữ liệu với các mục của template, giống như các bit của HTML được kết hợp ở một định dạng lặp đi lặp lại riêng biệt	CancelCommand, EditCommand, DeleteCommand, ItemCommand, SelectedIndexChanged, UpdateCommand	<asp:DataList id=DataList1 runat="server"></asp:DataList>
Repeater	Cho phép bạn tạo kiểu định dạng khác bảng cho dữ liệu. Bạn có thể liên kết dữ liệu với các mục của	Không có	<asp:Repeater id=Repeater1 runat="server"></asp:Repeater>

	template, giống như các bit của HTML được kết hợp ở định dạng lặp lại riêng biệt		
CheckBox	Rất giống control HTML bình thường hiển thị hộp kiểm để người dùng chọn hoặc thôi chọn	CheckChanged	<asp:CheckBox id=CheckBox1 runat="server"></asp: CheckBox>
CheckBoxList	Hiển thị một nhóm hộp kiểm mà tất cả chúng làm việc với nhau	SelectedIndexChanged	<asp:CheckBoxList id=CheckBoxList1 runat="server"></asp: CheckBoxList>
RadioButton	Rất giống control HTML bình thường hiển thị một nút để người dùng chọn hoặc thôi chọn	CheckChanged	<asp:RadioButton id=RadioButton1 runat="server"></asp: RadioButton>
RadioButtonList	Hiển thị một nhóm nút radio mà tất cả chúng đều làm việc với nhau	SelectedIndexChanged	<asp:RadioButtonList id=RadioButtonList1 runat="server"></asp: RadioButtonList>
Image	Rất giống control HTML, bình thường hiển thị ảnh bên trong trang	Không có	<asp:Image id=Image1 runat="server"></asp: Image>
Panel	Được dùng để tạo nhóm các control khác	Không có	<asp:Panel id=Panel1 runat="server">Panel</asp: Panel>
PlaceHolder	Đóng vai trò như một vị trí, nơi bạn có thể thêm các control phía máy chủ vào thời gian chạy	Không có	<asp:PlaceHolder id="PlaceHolder1" runat="server"></asp: PlaceHolder>
Calendar	Tạo phiên bản HTML của một lịch. Bạn có thể xác lập ngày tháng mặc định, di chuyển tới lui trên lịch, vân vân	SelectionChanged, MonthViewChanged	<asp:Calendar id=Calendar1 runat="server"></asp: Calendar>
AdRotator	Cho phép bạn chỉ định danh sách quảng cáo hiển thị. Mỗi lần người dùng hiển thị lại trang, màn hình xoay qua một loạt các quảng cáo.	Không có	<asp:AdRotator id=AdRotator1 runat="server"></asp: AdRotator>
Table	Rất giống control	Không có	<asp:Table id=Table1

	HTML bình thường.		runat="server"><asp:Table >
Xml	Được dùng để hiển thị các tài liệu XML bên trong HTML. Nó cũng có thể được dùng để thực hiện biến đổi XSLT trước khi hiển thị XML.	Không có	<asp:Xml id="Xml1" runat="server"><asp:Xml>
Literal	Giống như nhãn, nó hiển thị nguyên nghĩa, nhưng cho phép bạn tạo các nguyên nghĩa mới theo thời gian chạy và đặt chúng vào control này	Không có	<asp:Literal id="Literal1" runat="server"><asp: Literal>

Bảng 2: Các control phía máy chủ được dùng trong ASP.NET và các Web Form.

Tất cả các control này thay đổi xuất liệu của chúng dựa trên kiểu trình duyệt được phát hiện cho người dùng. Nếu trình duyệt của người dùng là IE, thì một phong cách phong phú hơn có thể được phát sinh bằng cách dùng một số đuôi mở rộng DHTML. Nếu một trình duyệt mức thấp được phát hiện (trình duyệt khác hơn IE), chuẩn HTML 3.2 bình thường được gửi trở lại trình duyệt của người dùng.

Các control Field Validator

Một kiểu control Web Form khác hữu hiệu hóa dữ liệu trên trình duyệt của client trước khi người dùng đệ trình dữ liệu tới server phía sau (back-end). Nếu người dùng điền một số dữ liệu vào một nhóm control trên trang HTML, thông thường bạn phải gửi tất cả dữ liệu trở lại máy chủ để nó có thể được hữu hiệu hóa bởi mã ASP hoặc ASP.NET. Thay vì phải thực hiện điều này lòng vòng để kiểm tra xem có giá trị nào được điền hay không, bạn có thể sử dụng một trong các control Field Validator để thực hiện

việc phía client kiểm tra (với điều kiện trình duyệt hỗ trợ điều này). Nếu bạn đang sử dụng IE 4.0 hoặc cao hơn, mỗi control này viết mã phía client vào trang HTML để các giá trị ấy có thể được kiểm tra mà không phải đi lòng vòng.

Control	Mô tả	Thí dụ mã HTML hoặc JavaScript
RequiredFieldValidator	Cho phép bạn kiểm tra một control trên form để xem nó được điền với bất kỳ điều gì hay không. Nếu chưa được điền, ErrorMessage mà bạn xác lập sẽ được hiển thị vào control này	<asp:RequiredFieldValidator id="RequiredFieldValidator1" runat="server" ErrorMessage="RequiredField Validator"></asp: RequiredFieldValidator>
CompareValidator	Cho phép bạn kiểm tra nội dung của một control dựa trên nội dung của một control khác trên form để xem chúng có khớp nhau hay không. Nếu chúng không khớp nhau, ErrorMessage bạn xác lập sẽ được hiển thị vào control này	<asp:CompareValidator id="CompareValidator1" runat="server" ErrorMessage="CompareValid ator" /></asp: CompareValidator>
RangeValidator	Cho phép bạn kiểm tra xem giá trị bạn nhập vào một control có ở trong phạm vi chỉ định hay không. Nếu không, ErrorMessage bạn xác lập sẽ được hiển thị vào control này	<asp:RangeValidator id="RangeValidator1" runat="server" ErrorMessage="RangeV alidator">
RegularExpressionValidator	Cho phép bạn kiểm tra xem nội dung của một control có khớp với mẫu nhập liệu (hữu thức hình thường) bạn đã định nghĩa hay không. Nếu không, ErrorMessage bạn xác lập sẽ được hiển thị vào control này	<asp:RegularExpressionValidator id="RegularExpressionValidator1" runat="server" ErrorMessage="RegularExpressionValidator"> </asp:RegularExpressionValidator>
CustomValidator	Cho phép bạn tự định hàm script phía server hoặc phía client để hợp lệ hóa nội dung của một control cụ thể. Bạn phải cho trở lại giá trị True hoặc False từ các hàm này. Nếu giá trị True được cho trở lại, thì tiến trình tiếp tục; nếu giá trị False được cho trở lại, ErrorMessage được chỉ định cho control này được hiển thị	<asp:CustomValidator id="CustomValidator1" runat="server" ErrorMessage="CustomValidator"></asp: CustomValidator>
ValidationSummary	Tự động thu thập tất cả các	<asp:ValidationSummary

	đặc tính ErrorMessage từ mỗi control validator khác trên form này và hiển thị từng control ở định dạng danh sách đánh số, danh sách bullet hoặc đoạn văn bản	id=ValidationSummary1 runat="server"></asp: ValidationSummary>

Bảng 3: *Các control Field Validator.*

Các control Field Validator này là các công cụ rất tốt để cho phép bạn kiểm tra nhập liệu của người dùng mà không phải vòng trở lại server.

Các control tùy biến người dùng

Ngoài các control lập sẵn trong .NET Framework, bạn còn có thể xây dựng các control tùy biến riêng. Thí dụ, có thể bạn muốn tạo một hệ thống menu, nơi mỗi mục menu được xây dựng từ một CSDL.

Global.asax

File Global.asax giống như file Global.asa trong ASP, ngoại trừ nó được biên dịch thay vì diễn dịch như nó ở trong ASP. Bạn vẫn có các thủ tục sự kiện để kích hoạt bên trong file Global.asax khi một số sự kiện xảy ra trên Web site của bạn. Global.asax được dùng khi bạn cần khởi tạo một (số) giá trị hoặc thực hiện một số thao tác khi một người dùng mới vào web site của bạn hoặc thoát khỏi nó. Bạn cũng có thể dùng nó để thực hiện sự khởi tạo nào đó khi người dùng đầu tiên hoặc cuối cùng vào hoặc thoát khỏi web site của bạn.

Trong danh sách sau, bạn sẽ thấy các thủ tục sự kiện bạn có thể sử dụng bên trong file Global.asax.

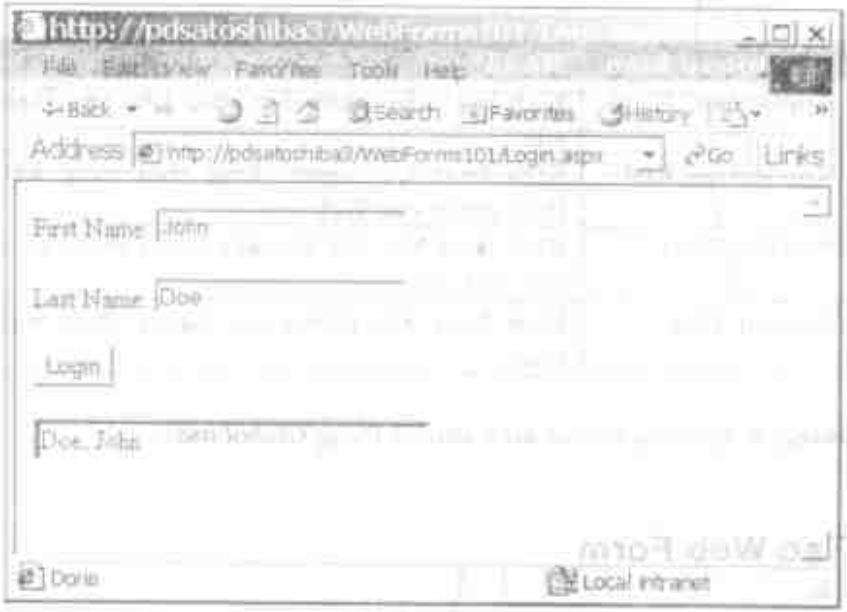
Thủ tục sự kiện	Mô tả
Application_Start	Kích hoạt khi người dùng đầu tiên vào Web site
Application_End	Kích hoạt khi người dùng cuối cùng kết thúc phiên của Web site
Session_Start	Kích hoạt khi bất kỳ một người dùng mới vào Web site
Session_End	Kích hoạt khi phiên của người dùng kết thúc

Bảng 4: *Các thủ tục sự kiện sẵn có trong Global.asax.*

Tạo Web Form

Chúng ta hãy tạo một Web Form để cho phép người dùng nhập liệu họ và tên của họ. Sau khi nhập dữ liệu vào hai trường văn bản này trên trang Web, người dùng nhấp nút Login và Last name, First name của người dùng xuất hiện ở nhãn bên dưới nút Login.

Hình 4 trình bày mẫu đăng nhập của Web Form mà bạn sẽ dùng.



Hình 4: Mẫu này trình bày cách nhập dữ liệu và hiển thị nó trở lại vào control nhãn.

Các bước xây dựng Form đăng nhập

Để bắt đầu, bạn phải tạo một project Web Application mới trong Visual Studio.

1. Khởi động Visual Studio và nhấp Create New Project từ Start Page.
2. Tô sáng Visual Basic Projects từ hiển thị cây ở bên trái.
3. Nhấp Web Application trong cửa sổ Templates.
4. Điền vào Name của project này là **WebForms101**.
5. Chọn vị trí của máy, nơi bạn muốn tạo Web site này.

6. Nhấp OK để bắt đầu tiến trình tạo một project Web Application mới.

7. Bây giờ bạn sẽ có Web Form với tên WebForm1.aspx trong project Visual Studio. Đặt lại tên form này là **Login.aspx**.

8. Hiển thị Toolbox và tạo form ở hình 3 bằng cách thêm các control thích hợp và xác lập các đặc tính của các control này như được trình bày ở bảng 5.

Kiểu control	Đặc tính	Giá trị
Label	Name	Label1
	Text	First Name
TextBox	Name	txtFirst
	Text	
Label	Name	Label2
	Text	Last Name
TextBox	Name	txtLast
	Text	
Button	Name	btnSubmit
	Text	Login
Label	Name	lblName
	BorderStyle	Inset
	Text	

***Bảng 5:** Sử dụng các control này để xây dựng form đăng nhập.*

Vào lúc này, bạn có thể chạy ứng dụng và xem Web Form này xuất hiện trong trình duyệt của bạn. Mặc dù trang này chưa có tính năng nào, nhưng việc thực tập này là một kiểm tra tốt để chắc chắn mọi thứ tiến hành cho tới điểm này.

- Ấn **F5** để chạy ứng dụng mẫu này.

Nếu bạn nhận một thông báo lỗi cho biết bạn cần có trang Start Page, nhấp nút chuột phải vào trang Login.aspx và chọn Set as Start Page từ menu context.

Bây giờ bạn sẽ thấy Web Form được hiển thị trong trình duyệt và bạn có thể nhập dữ liệu vào hai trường văn bản. Nếu bạn nhấp nút Login, không có gì xảy ra vì bạn chưa cho nó thực hiện điều gì. Kế tiếp bạn sẽ tìm hiểu cách làm cho nút Login thực hiện một điều gì đó.

Thêm mã vào nút

Chúng ta hãy thêm một số mã vào nút để nó gửi dữ liệu bạn đã nhập vào các hộp văn bản và điền dữ liệu thích hợp vào nhãn bên dưới control nút.

1. Dừng chương trình chạy bằng cách đóng trình duyệt.

2. Khi xem trang Login ở chế độ thiết kế, nhấp đôi control nút Login. Bây giờ bạn sẽ thấy một cửa sổ mã xuất hiện với thủ tục sự kiện btnSubmit. Nhấp nút chuột phải.

3. Điền thủ tục sự kiện Click vào như ở mã sau.

```
Private Sub btnSubmit_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnSubmit.Click
    lblName.Text = txtLast.Text & ", " & txtFirst.Text
End Sub
```

Những gì bạn vừa thực hiện là để truy tìm đặc tính văn bản từ cả hộp văn bản txtLast lẫn txtFirst và đặt dữ liệu vào control nhãn bên dưới nút Login. Nó giống như bạn lập trình trong VB. Thực ra, điểm chủ yếu nhất của .NET đều quy về việc viết mã, dù

cho các ứng dụng Windows hoặc các ứng dụng Web đều sẽ thể hiện như nhau.

Tóm tắt

Việc sử dụng các trang HTML, các trang ASPX (Web Form), các control HTML và các control Web Form, bây giờ bạn có thể có một môi trường RAD thuyết phục để tạo các ứng dụng Web mềm dẻo, phù hợp và rất mạnh trong ASP.NET. Bạn sẽ thấy là việc di chuyển sự phát triển sang ASP.NET dễ hơn di chuyển từ VB6 sang ASP khi bạn có cùng môi trường phát triển và cùng kiểu lập trình.

Câu hỏi ôn tập

1. Các Web Form được tạo thành từ hai thành phần, chúng là các thành phần nào?

2. Bạn có thể sử dụng hai kiểu control giao diện người dùng nào trên các Web Form?

3. Tất cả các Web Form thừa kế từ lớp cơ sở nào?

4. Ích lợi của việc sử dụng control Field Validator là gì?

5. Bạn làm gì với file Global.asax?

Bài tập

- Tạo form đăng nhập được trình bày ở chương này.

Trả lời câu hỏi ôn tập

1. Visual (ASPX) và mã.
2. HTML và Server-Side
3. System.Web.UI.WebControls.
4. Để kiểm tra dữ liệu ở phía client thay vì phải thực hiện lòng vòng.
5. Sử dụng nó để khởi tạo hoặc thực hiện một thao tác khi người dùng nhập vào hoặc thoát khỏi một web site.

Chương 26

Tạo form đăng nhập Web

- Sử dụng ASP.NET và các Web Form để tạo form đăng nhập.
- Trình bày cách dùng các thành phần trong một Web Form.

Tạo form đăng nhập Web

Có lẽ bạn đã ghé tới một số web site, nơi họ yêu cầu bạn đăng nhập để truy xuất các tính năng của site. Bạn phải đăng nhập vào mạng của mình để truy xuất các tài nguyên mạng. Là một người phát triển, bạn rất có thể sẽ được yêu cầu tạo form đăng nhập cho các ứng dụng của bạn để người dùng có thể nhận được một số tính năng của chương trình. Ở chương này, bạn sẽ tìm hiểu cách tạo form đăng nhập mà nó có thể trở thành cơ sở để một hệ thống đăng nhập vào các ứng dụng web của bạn.

Tại sao cần thực hiện điều này

Khi bạn tạo một ứng dụng, rất có thể bạn muốn kiểm soát những người dùng nào được phép vào ứng dụng của bạn. Ngoài ra, bạn có thể muốn theo dõi người dùng đã làm gì trong ứng dụng của bạn hoặc người dùng được phép truy xuất các lãnh vực nào của ứng dụng. Để thực hiện điều này, bạn phải yêu cầu người dùng cho tên, địa chỉ email, User ID hoặc một định danh duy nhất nào đó của họ để bạn có thể theo dõi họ. Nhiều lúc bạn cũng có thể cần yêu cầu họ cho Password.

Tạo một Web Form để cho phép người dùng nhập họ và tên. Bạn sẽ truy tìm các giá trị được nhập từ web form và gán cho họ các đặc tính của lớp Employee. Lớp Employee này sẽ mở một kết nối với SQL Server và kiểm tra xem họ và tên có ở trong bảng Employee của CSDL Northwind hay không. Nếu tên có trong bảng ấy, người dùng sẽ được phép tiếp tục vào web site, nếu tên đó không có trong bảng, họ sẽ được yêu cầu đăng nhập lại. Hình 1 trình bày mẫu đăng nhập web form mà bạn sẽ dùng ở chương này.

Sử dụng mẫu lập sẵn

Nếu bạn không muốn thực hiện các bước ở chương này, bạn chỉ cần xem một mẫu lập sẵn và thực hiện theo các bước được trình bày bên dưới để tạo mẫu này trên máy của bạn. Vì project lập sẵn đã được phát triển trên máy chủ web, nếu bạn cố mở project này, tên của máy chủ web sẽ không chính xác. Bạn phải xây dựng project Web Form riêng và nạp các file vào project mới để có thể chạy mẫu này.

Hình 1: *Form đăng nhập được dùng để hữu hiệu hóa chỉ các người dùng hợp lệ được phép vào web site.*

1. Khởi động Visual Studio và nhấp Create New Project từ Start Page.
2. Tô sáng Visual Basic Projects từ hiển thị cây ở bên trái.
3. Nhấp Web Application trong cửa sổ Templates.
4. Điền vào Name là **LoginSample**.
5. Chọn vị trí của máy, nơi bạn muốn tạo web site này.
6. Nhấp OK để bắt đầu tiến trình tạo project Web Application mới.

Bây giờ bạn có một web form với tên Web Form1.aspx trong project Visual Studio. Xóa file này bằng cách nhấp nút chuột phải vào nó trong Solution Explorer và chọn Delete từ menu.

Nạp các file sau vào project từ thư mục, nơi bạn thực hiện chương này. Chắc chắn nạp chúng theo thứ tự chính xác được liệt kê trong bảng 1. Khi bạn thêm mỗi file ASPX, nó sẽ hỏi xem bạn có muốn sử dụng .VB form cùng tên là "code-behind class" cho web form hay không. Trả lời Yes cho lời nhắc này.

File Name
Employee.vb
Login.vb
Login.aspx
LoginFailure.htm
LoginSuccess.vb
LoginSuccess.aspx

Bảng 1: *Các file cần nạp để tạo mẫu lập sẵn.*

Bây giờ nhấp nút chuột phải vào Login.aspx và lựa menu Set as Start Page từ menu.

Thay đổi chuỗi kết nối

Nếu bạn sử dụng một SQL Server trên máy cục bộ, bạn sẽ có thể chạy ứng dụng này ngay. Nếu bạn không có, bạn sẽ phải thay đổi chuỗi kết nối trong file Employee.vb để trỏ tới tên của SQL Server của bạn.

Mở file Employee.vb, tìm chuỗi kết nối bằng phương thức EmployeeValid và đổi tham số Data Source thành tên của server. Bạn cũng có thể thay đổi User ID và tham số Password để sử dụng ID đăng nhập hợp lệ cho server của bạn.

Ở điểm này, bạn có thể chạy mẫu này. Ấn **F5** để chạy chương trình. Bạn sẽ thấy một trang đăng nhập xuất hiện như ở hình 1.

Các bước xây dựng form đăng nhập

Để bắt đầu bạn phải tạo một project Web Application mới trong Visual Studio.

1. Khởi động Visual Studio và nhấp Create New Project từ Start Page.
2. Tô sáng Visual Basic Projects từ hiển thị cây ở bên trái.
3. Nhấp ASP.NET Web Application trong cửa sổ Templates.
4. Điền vào Name là **LoginSample**.
5. Chọn vị trí máy, nơi bạn muốn tạo web site này.
6. Nhấp OK để bắt đầu tiến trình tạo project Web Application mới.
7. Bây giờ bạn có một web form với tên Web Form1.aspx trong project Visual Studio. Đặt lại tên form này là **Login.aspx**.
8. Hiển thị Toolbox và tạo form như ở hình 1, bằng cách thêm các control thích hợp và xác lập các đặc tính của các control này như được trình bày ở bảng 2.

Kiểu control	Đặc tính	Giá trị
Label	Name	Label1
	Text	First Name
TextBox	Name	txtFirst
	Text	Andrew
Label	Name	Label2
	Text	Last Name

TextBox	Name	txtLast
	Text	Fuller
Button	Name	btnSubmit
	Text	Login

Bảng 2: *Sử dụng các control này để xây dựng form đăng nhập.*

Vào lúc này bạn có thể chạy ứng dụng này và xem web form này xuất hiện trong trình duyệt của bạn. Dù trang này chưa có tính năng nào, nhưng đây là một kiểm tra tốt để chắc chắn mọi thứ cho tới điểm này đều tiến hành được.

- Ấn **F5** để chạy ứng dụng mẫu này.

Nếu bạn nhận một thông báo lỗi cho bạn biết cần có Start Page, nhấp nút chuột phải vào trang Login.aspx và chọn Set as Start Page từ menu context.

Bây giờ bạn thấy web form của bạn được hiển thị trong trình duyệt và bạn có thể nhập dữ liệu vào hai trường văn bản. Nếu bạn nhấp nút Login, không có gì xảy ra vì bạn chưa cho nó thực hiện điều gì. Bạn sẽ tìm hiểu cách làm cho nút Login này thực hiện điều gì đó.

Tạo lớp Employee

Bạn phải cố tạo mọi chương trình như một đối tượng được định hướng khi có thể. Đối với mẫu này, bạn sẽ tạo một lớp Employee để có hai đặc tính và một phương thức. Bạn sẽ tạo đặc tính FirstName và LastName. Mỗi đặc tính sẽ lưu giữ các giá trị được đọc vào từ web form. Bạn sẽ tạo một phương thức

EmployeeValid để cố truy tìm một nhân viên từ bảng Employee khớp với họ và tên được chuyển vào lớp ấy bằng các đặc tính.

1. Lựa Project | Add New Item từ menu Visual Studio.
2. Lựa Class từ cửa sổ Templates.
3. Đổi tên thành **Employee.vb**.
4. Nhắp Open để tạo Class mới này.
5. Gõ mã sau vào cửa sổ mã.

```
Public Class Employee
    Private mstrFirst As String
    Private mstrLast As String

    Property LastName() As String
        Get
            Return mstrLast
        End Get
        Set(ByVal Value As String)
            mstrLast = Value
        End Set
    End Property

    Property FirstName() As String
        Get
            Return mstrFirst
        End Get
        Set(ByVal Value As String)
            mstrFirst = Value
        End Set
    End Property

    Public Function EmployeeValid() As Boolean
        Dim oCmd As OleDb.OleDbCommand
        Dim oDR As OleDb.OleDbDataReader
        Dim strSQL As String
        Dim strConn As String
        Dim boolFound As Boolean
```

```
            '** Build the connect string
            strConn = "Provider=sqloledb"
            strConn &= ";Data Source=(local)"
            strConn &= ";Initial Catalog=Northwind"
            strConn &= ";User id=sa"
            strConn &= ";Password="

            '** Set the command text
            strSQL = "SELECT EmployeeID FROM Employees "
            strSQL &= "WHERE LastName = '" & mstrLast & "'
"
            strSQL &= "AND FirstName = '" & mstrFirst & "'
"

            '** Build the Command object
            oCmd = New OleDb.OleDbCommand()
            With oCmd
                .CommandText = strSQL
                .CommandType = Data.CommandType.Text
                .Connection =
                 New OleDb.OleDbConnection(strConn)
                .Connection.Open()
            End With

            '** Submit the SQL
            oDR = oCmd.ExecuteReader(
             CommandBehavior.SequentialAccess)
            boolFound = oDR.Read

            '** Close Data Reader and Connection
            oDR.Close()
            oCmd.Connection.Close()

            '** Return True or False
            Return boolFound
        End Function
    End Class
```

Ở phương thức EmployeeValid, bạn sẽ tạo một câu lệnh SQL SELECT để truy tìm bản ghi của một nhân viên so khớp hai đặc tính được chuyển vào lớp này. Bạn sẽ đệ trình SELECT bằng cách dùng phương thức Execute của đối tượng OleDbCommand. Bạn sẽ điền vào đối tượng OleDbDataReader với các kết quả của câu lệnh SELECT. Khi bạn thi hành phương thức Read trên

DataReader, phương thức đó sẽ cho trở lại giá trị của mã đã được đọc hoặc cho giá trị False nếu không có mã nào được đọc. Biến cục bộ boolFound được xác lập là giá trị trả lại này và nó được cho trở lại từ hàm ấy.

Mẹo:

Nhớ là câu lệnh Return sẽ thoát khỏi một hàm tại điểm đó trong mã. Vì vậy nếu bạn cần làm việc thêm, như đóng kết nối, bạn phải lưu giá trị trả lại vào một biến cục bộ và đặt câu lệnh Return ở ngay cuối hàm.

Phương thức này giả định bạn đã cài đặt SQL Server và có thể truy xuất nó. Chuỗi nhà cung cấp ở mã trên có tham số Data Source được xác lập là (local), là máy cục bộ của bạn. Bạn có thể phải sửa đổi tham số này để trỏ tên của máy SQL Server. Bạn cũng có thể phải sửa đổi tham số User ID thành một User ID khác và bạn cũng có thể phải thay đổi tham số Password để có thể đăng nhập chính xác server này.

Nếu bạn không truy xuất được SQL Server, thay vào đó bạn có thể phải sử dụng CSDL Northwind Access. CSDL Northwind được phân phối với Microsoft Access. Nếu bạn muốn kết nối với CSDL Access này, bạn sẽ dùng chuỗi nhà cung cấp sau.

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Northwind\Northwind.mdb"
```

Thử đăng nhập

Bây giờ bạn đã viết lớp Employee, bạn có thể dùng nó từ thủ tục sự kiện Click của nút Login.

1. Mở Login.aspx Web Form ở chế độ thiết kế.

2. Nhấp đôi nút Login.

3. Viết mã được trình bày bên dưới vào thủ tục sự kiện Click.

```
Private Sub btnSubmit_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnSubmit.Click
    Dim oEmp As Employee = New Employee()

    With oEmp
        .LastName = txtLast.Text
        .FirstName = txtFirst.Text

        If .EmployeeValid() Then
            Session("EmpName") = .FirstName & _
                " " & .LastName
            'Response.Write("<B>Welcome " & _
             .FirstName & "</B>")
        Else
            'Response.Write("<B>Invalid            Login
Name</B>")
        End If
    End With
End Sub
```

Ở mã này, bạn tạo một đối tượng Employee mới. Bạn truy tìm các giá trị từ hai hộp văn bản và đặt các giá trị đó vào đặc tính LastName và FirstName của đối tượng Employee. Kế tiếp, bạn gọi phương thức EmployeeValid và nếu giá trị True được cho trở lại, bạn sẽ ghi "Welcome <Name>" tới người được đăng nhập. Nếu giá trị False được cho trở lại từ phương thức EmployeeValid, bạn sẽ ghi "Invalid Login Name" tới trình duyệt.

Bạn sử dụng phương thức ASP.NET Response objects Write để hiển thị các câu này trở lại trình duyệt của người dùng. Cùng với các câu này bạn sẽ dùng tag kí tự đậm (<B>) bao quanh các câu này để làm cho các từ nổi bật ở trên trình duyệt của người dùng.

Bây giờ chúng ta chạy ứng dụng, gõ một tên hợp lệ vào để xem ứng dụng này hoạt động như thế nào.

1. Chạy ứng dụng bằng cách ấn **F5**.

2. Nhấp nút Login để đệ trình tên Andrew Fuller với tiến trình đăng nhập. Bạn sẽ thấy văn bản Welcome Andrew được hiển thị bằng kí tự đậm ở trên đầu form.

3. Bây giờ đổi tên ấy thành một tên khác và nhấp nút Login. Bạn sẽ thấy văn bản Invalid Login Name được hiển thị bằng kí tự đậm ở trên đầu form.

Tạo các form phản hồi

Bạn nên tạo một giao diện tốt hơn một chút để hiển thị một số văn bản ở trên đầu form, thông báo cho người dùng kết quả của nút Login. Bạn sẽ tạo hai Web Form tách riêng, một cho đăng nhập thành công và một cho đăng nhập không thành công.

1. Lựa Project | Add New Item từ menu Visual Studio.

2. Lựa HTML Page từ cửa sổ Templates.

3. Xác lập Name là **LoginFailure.htm**.

4. Nhấp Open để tạo HTML Page mới này.

5. Gõ các từ sau ngay vào trang: The first name and last name that you typed in, do not exist in the employee database, please re-enter the information. (Họ tên bạn đã gõ vào không có trong CSDL nhân viên, làm ơn nhập lại thông tin).

6. Ấn phím Enter hai lần sau khi nhập các từ trên.

7. Gõ: Click here to re-login to the site (Nhấp vào đây để đăng nhập lại site).

8. Bây giờ tô sáng các từ bạn vừa gõ ở bước 7.

9. Lựa Insert | Hyperlink từ menu Visual Studio.

10. Xóa trường URL và gõ vào **Login.aspx**.

11. Nhấp OK.

12. Hiển thị Login.aspx Web Form.

13. Nhấp đôi nút Login.

14. Tìm dòng sau:

```
Response.Write("<B>Invalid Login Name</B>")
```

15. Đổi nó thành:

```
Response.Redirect("LoginFailure.htm")
```

16. Một lần nữa bạn sử dụng đối tượng Response của ASP.NET để thực hiện một tác vụ cho bạn. Trong trường hợp này, bạn sử dụng phương thức Redirect để hiển thị một trang khác trong trình duyệt của người dùng. Phương thức Redirect gửi toàn bộ URL của trang yêu cầu tới trình duyệt client. Trình duyệt sẽ tự động đệ trình lại yêu cầu này trở về server để điều hướng tới trang web LoginFailure. Do đó khi bạn sử dụng phương thức này sẽ phải đi lòng vòng. Dù đi lòng vòng thường không được ưa thích, đây là một cách hướng người dùng tới một trang khác.

Bây giờ chúng ta hãy thử trang web LoginFailure mới này.

1. Chạy ứng dụng.
2. Đặt vào một tên nhân viên không có thật.
3. Nhấp nút Login. Bây giờ bạn sẽ thấy trang Login Failure.
4. Bạn có thể nhấp vào siêu liên kết và nó sẽ đưa bạn về trang đăng nhập.

Tạo trang đăng nhập thành công

Bạn tạo một trang để hiển thị tên người dùng và thông điệp chào đón khi người dùng đăng nhập vào site thành công.

1. Lựa Project | Add New Item từ menu Visual Studio.
2. Lựa Web Form từ cửa sổ Templates.
3. Xác lập Name là **LoginSuccess.aspx**.
4. Nhấp Open để tạo Web Form mới này.
5. Nhấp đôi Web Form.
6. Bạn sẽ được định vị trên thủ tục sự kiện Load.
7. Thêm mã sau vào thủ tục Load này.

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

```
    Response.Write("Welcome to our web site " & _
        Session("EmpName").ToString())
End Sub
```

Trong mã này, bạn sẽ thực hiện một Response.Write để hiển thị một số văn bản trên trang web. Bạn sẽ lấy được tên người dùng từ biến Session mà bạn sẽ tạo được gọi là EmpName.

8. Mở form Login.aspx.

9. Nhấp đôi nút Login.

10. Tìm dòng mã sau:

```
Response.Write("<B>Welcome " & .FirstName & "</B>")
```

11. Đổi nó thành

```
Session("EmpName") = .FirstName & _
  " " & .LastName
Server.Transfer("LoginSuccess.aspx")
```

Bạn sẽ tạo biến phiên (session) để lưu giữ họ và tên của người dùng. Các giá trị này có thể được truy tìm từ các hộp văn bản trên web form hoặc có thể truy tìm từ đối tượng Employee. Đối tượng ASP.NET Server là một công cụ khác mà bạn có thể dùng từ mã ASP.NET của mình. Trong trường hợp này, bạn sẽ dùng phương thức Server.Transfer để hướng trình duyệt của người dùng tới một trang mới trên web site của bạn. Không giống phương thức Response.Redirect, phương thức Server.Transfer không gửi yêu cầu này tới trình duyệt người dùng. Server tự hướng tới trang web mới và chỉ gửi kết quả trở lại trình duyệt người dùng.

Bạn có thể chạy lại ứng dụng và đăng nhập bằng một tên hợp lệ để có thể xem trang mới này.

1. Chạy ứng dụng.

2. Nhấp nút Login khi bạn đã điền vào tên Andrew Fuller.

3. Nếu mọi thứ thành công, bây giờ bạn sẽ thấy văn bản Welcome to our web site Andrew Fuller.

Tóm tắt

Bạn có thể tạo một hệ thống đăng nhập khá dễ dàng với các web form và ASP.NET. Bạn có thể dễ dàng mở rộng hệ thống đăng nhập này để sử dụng các cookie để truy tìm tên người dùng và tự động điền vào các giá trị mỗi lần họ trở lại site. Bạn có thể thêm chứng nhận bảo mật để thông tin người dùng không được truyền bằng văn bản thông thường trên Internet. Bạn cũng có thể thêm văn bản bảo mật. Lưu ý, bạn không thể dùng Server.Transfer với các trang HTML, mà chỉ với các trang ASP.NET.

Câu hỏi ôn tập

1. Để tạo một ứng dụng Web Form bạn chọn template dự án (project) nào trong hộp thoại New Project?

2. Bạn dùng control nào để nhận thông tin nhập liệu của người dùng?

3. Bạn dùng control nào cho người dùng đệ trình form tới máy chủ để xử lý?

4. Bạn dùng phương thức nào để hướng người dùng tới một trang khác mà không đi lòng vòng tới client?

5. Bạn phải dùng phương thức nào để định hướng tới một trang HTML?

Bài tập

- Tạo Login Form ở chương này.

Trả lời câu hỏi ôn tập

1. ASP.NET Web Appllication.

2. TextBox.

3. Button.

4. Server.Transfer.

5. Response.Redirect.

Chương 27

Các control hữu hiệu hóa

- Tìm hiểu các control hữu hiệu hóa trong ASP.NET.
- Tìm hiểu cách tạo các form để hữu hiệu hóa dữ liệu.

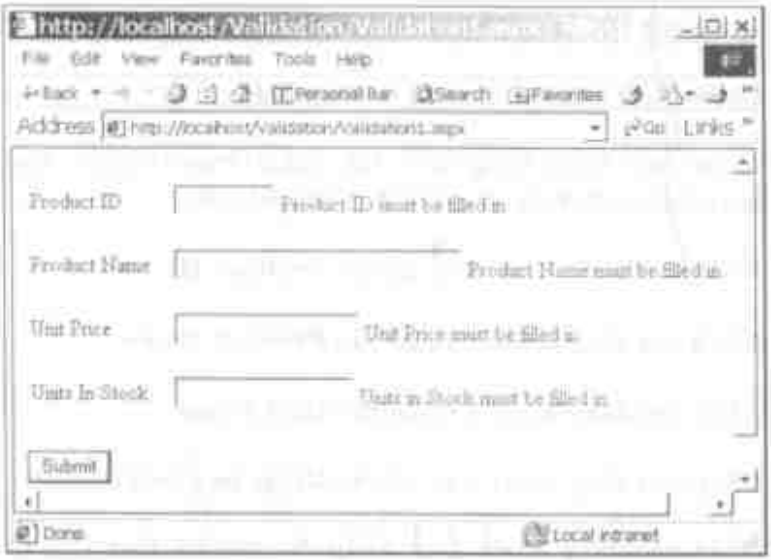
Các control hữu hiệu hóa

Có một loạt các control sẽ giúp bạn hữu hiệu hóa dữ liệu khi nó được nhập vào phía client. Sự hữu hiệu hóa (validation) này có thể xảy ra với điều kiện bạn có đúng loại trình duyệt (thường là IE 4.0 hoặc phiên bản sau). Nếu bạn không có trình duyệt "up-level", như IE, thì bạn vẫn có thể dùng các control này, tuy nhiên, sự hữu hiệu hóa sẽ xảy ra trên server. Nếu trình duyệt hỗ trợ đúng loại scripting phía client sử dụng các control này có thể thực sự giúp tăng tốc độ xử lý của các form bằng cách loại bỏ việc đi lòng vòng tới server. Điều đó cũng có thể đơn giản hóa việc viết mã mà bạn phải thực hiện trên chương trình phụ.

Mỗi control hữu hiệu hóa là một control tách riêng từ control bạn sẽ hữu hiệu hóa. Thí dụ, nếu bạn có control hộp văn bản, nơi bạn muốn người dùng điền vào dữ liệu nào đó, bạn sẽ đặt control hữu hiệu hóa bên control đó và cho control hữu hiệu hóa ấy biết tên của control bạn muốn hữu hiệu hóa dữ liệu nhập. Khi người dùng cố đệ trình control với chương trình phía sau (back end), control hữu hiệu hóa sẽ nhận dữ liệu từ control khác và hữu hiệu hóa nó theo một số nguyên tắc. Các nguyên tắc sẽ khác nhau dựa vào kiểu control hữu hiệu hóa được dùng. Ở chương này, bạn sẽ tìm hiểu về mỗi control hữu hiệu hóa khác nhau có thể dùng trong .NET Framework.

Control hữu hiệu hóa trường đòi hỏi

Control đầu tiên bạn sẽ tìm hiểu cách dùng là control hữu hiệu hóa trường đòi hỏi (Required Field Validation). Control này sẽ kiểm tra để chắc chắn là dữ liệu nào đó được điền vào control phải hữu hiệu hóa. Hình 1 trình bày một thí dụ về control hữu hiệu hóa đòi hỏi để hiển thị văn bản lỗi ở bên phải của các hộp văn bản thông thường. Vì không có hộp văn bản nào đã được điền vào bằng dữ liệu, nên mỗi hộp hiển thị một thông báo lỗi với người dùng. Nếu bạn đang sử dụng IE 4.0 hoặc phiên bản cao hơn, thì tất cả sự hữu hiệu hóa này xảy ra ở phía client và không phải đi lòng vòng. Điều này tiết kiệm được rất nhiều thời gian và giúp làm cho ứng dụng mở rộng hơn vì ít tài nguyên được dùng trên server.



Hình 1: *Control hữu hiệu hóa trường đòi hỏi giúp bạn đừng các trường có mục nhập trống.*

Tạo form hữu hiệu hóa đòi hỏi

Bây giờ bạn sẽ đi qua các bước cần thiết để tạo Web Form được trình bày ở hình 1.

1. Khởi động Visual Studio.NET và nhấp nút New Project.
2. Lựa Visual Basic Project từ hiển thị cây ở bên trái của hộp thoại.
3. Lựa template ASP.NET Web Application.
4. Điền tên là **Validation**.

5. Nhấp nút OK.

6. Đặt lại tên WebForm1.aspx là **Main.aspx**.

7. Thêm một bảng bằng cách lựa Table|Insert|Table. Xác lập số dòng là **4** và số cột là **2**. Nhấp nút OK.

8. Nhấp vào dòng 1, cột 1 và gõ vào **Product ID**.

9. Nhấp vào dòng 2, cột 1 và gõ vào **Product Name**.

10. Nhấp vào dòng 3, cột 1 và gõ vào **Unit Price**.

11. Nhấp vào dòng 4, cột 1 và gõ vào **Units in Stock**.

12. Nhấp vào dòng 1, cột 1 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

13. Xác lập đặc tính Name là **txtProductID**.

14. Nhấp vào dòng 1, cột 2 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

15. Xác lập đặc tính Name là **txtProductName**.

16. Nhấp vào dòng 1, cột 3 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

17. Xác lập đặc tính Name là **txtUnitPrice**.

18. Nhấp vào dòng 1, cột 4 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

19. Xác lập đặc tính Name là **txtUnitsInStock**.

20. Nhấp vào dòng 1, cột 1 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

21. Xác lập đặc tính Name là **txtProductID**.

22. Định vị con trỏ của bạn ngay dưới bảng và nhấp đôi control Button.

23. Xác lập đặc tính Text là Submit.

24. Xác lập đặc tính Name là **btnSubmit**.

Móc nối các Required Field Validator

Bây giờ bạn đã hoàn tất giao diện cơ sở mà bạn sẽ cần móc nối các control Required Field Validator với mỗi hộp văn bản này. Đối với mỗi hộp văn bản trên form, thực hiện các hướng dẫn bên dưới.

1. Đặt con trỏ ngay bên phải của hộp văn bản.

2. Nhấp đôi control RequiredFieldValidator để thêm nó ngay bên hộp văn bản.

3. Xác lập đặc tính ControlToValidate là tên của hộp văn bản vào bên trái control này.

4. Xác lập đặc tính ErrorMessage là một chuỗi, chẳng hạn như "? must be filled in". Điền vào dấu chấm hỏi với dữ liệu người dùng sẽ nhập.

Tại điểm này bạn sẵn sàng chạy thử mẫu của mình để xem nó có hữu hiệu hóa các control mà không có dữ liệu hay không.

1. Ấn **F5** để chạy ứng dụng. Bạn có thể phải xác lập Start Page trước.

2. Đừng nhập bất kỳ dữ liệu nào vào các control hộp văn bản và nhấp nút Submit.

3. Bây giờ bạn sẽ thấy các thông báo lỗi xuất hiện ngay bên các hộp văn bản.

Control hữu hiệu hóa phạm vi

Control RangeValidator cho phép bạn kiểm tra xem một trị số được nhập vào hộp văn bản có ở giữa một phạm vi trị số nào đó hay không. Bạn sẽ đặt control RangeValidator bên hộp văn bản giống như control RequiredFieldValidator. Bạn sẽ xác lập đặc tính ErrorMessage là lỗi bạn muốn hiển thị nếu người dùng không nhập trị số thích hợp. Sau đó bạn thấy một thông điệp giống như thông điệp được hiển thị ở hình 2.

***Hình 2:** Control hữu hiệu hóa phạm vi.*

Các bước sử dụng RangeValidator

Bây giờ bạn sửa đổi form bạn vừa tạo và thêm control RangeValidator để hữu hiệu hóa dữ liệu được nhập vào hộp văn bản Unit Price.

1. Đặt con trỏ ngay sau control RequiredFieldValidator bên hộp văn bản Unit Price.

2. Nhấp đôi control RangeValidator trong hộp công cụ.

3. Đổi đặc tính ErrorMessage thành **Unit Price must be filled in**.

4. Xác lập đặc tính MinimumValue là **0**.

5. Xác lập đặc tính MaximumValue là **99999999**.

6. Bây giờ bạn sẵn sàng chạy thử control mới này.

7. Ấn F5 để chạy ứng dụng này.

8. Nhập dữ liệu vào hộp văn bản Product ID và ProductName.

9. Nhập giá trị e –1 vào hộp văn bản Unit Price.

10. Khi bạn ấn tab ra khỏi control, bạn sẽ thấy thông báo lỗi được hiển thị ở mé phải.

Dù thông báo lỗi hiển thị, nhưng nó không đặt thông báo lỗi ngay sau hộp văn bản. Vấn đề đó là control Validator khác ngăn chặn. Mặc dù control này không hiển thị, nhưng nó vẫn chiếm không gian trên trang HTML.

Để giải quyết vấn đề này, bạn phải trở lại chế độ thiết kế. Nhấp control RequiredFieldValidator bên hộp văn bản Unit Price và xác lập đặc tính Display là **Dynamic**. Điều này sẽ buộc RequiredFieldValidator không sử dụng hết không gian nếu thông báo lỗi sẽ được hiển thị.

Control tóm lược sự hữu hiệu hóa

Trong nhiều tình huống khác, bạn sẽ không có đủ màn hình để có thể hiển thị các thông báo lỗi bên mỗi hộp văn bản. Trong trường hợp này, bạn nên chọn một vùng màn hình khác và để tất cả các thông báo lỗi hiển thị trong vùng màn hình này. Bạn có thể thực hiện điều này bằng cách dùng control ValidationSummary như được trình bày ở hình 3.

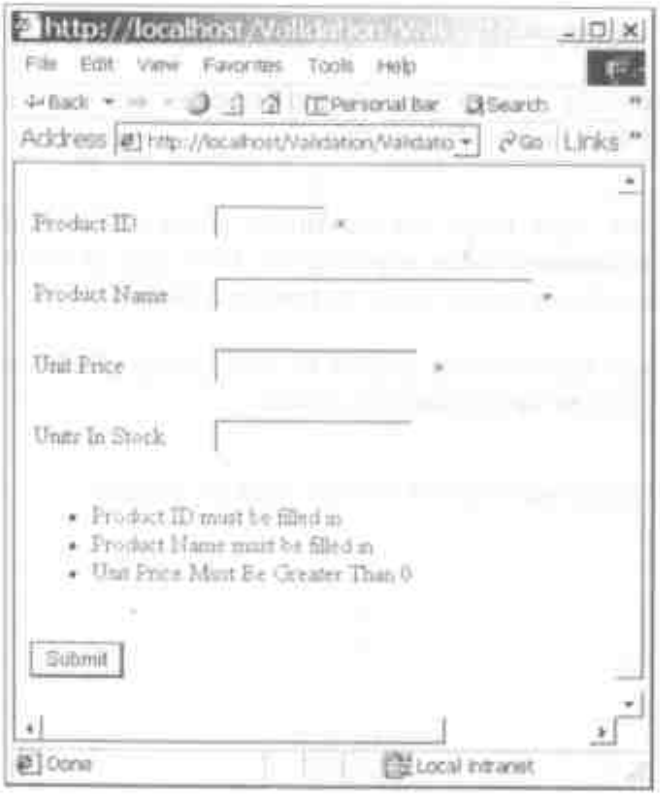
Các bước sử dụng control ValidationSummary

Nhấp lần lượt mỗi control Validation trong form của bạn và xác lập đặc tính Text là một dấu sao (*).

- Đặt con trỏ ngay trước nút Submit và nhấp đôi control ValidationSummary trong hộp công cụ. Điều này sẽ thêm control này vào ngay sau Table và trước nút Submit.

- Bây giờ bạn chỉ cần chạy project và xem những gì xảy ra khi bạn nhập dữ liệu không hợp lệ vào.

Control ValidationSummary rất dễ thực hiện và sử dụng.



***Hình 3:** Control Summary Validation.*

Control hữu hiệu hóa so sánh

Trong một số trường hợp, bạn cần so sánh dữ liệu trong một trường với dữ liệu ở một trường khác. Điển hình nhất của việc này là yêu cầu người dùng thay đổi Password của họ. Họ sẽ phải nhập Password của họ và sau đó gõ lại nó để xác minh họ đã gõ vào chính xác mật khẩu của họ. Một thí dụ được trình bày ở hình 4 bên dưới.

Hình 4: *Control hữu hiệu hóa so sánh kiểm tra hai control xem giá trị của chúng có bằng nhau hay không.*

Các bước kiểm tra control CompareValidator

Để tạo màn hình này bạn sẽ thực hiện các bước được trình bày bên dưới.

1. Thêm một Web Form mới bằng cách lựa Project | menu Add Web Form.

2. Đặt tên là LoginPassword.aspx.

3. Thêm một bảng bằng cách lựa Table | Insert | Table. Xác lập số dòng là **3** và số cột là **2**. Nhấp nút OK.

4. Nhấp vào dòng 1, cột 1 và gõ **Login ID** vào.

5. Nhấp vào dòng 2, cột 1 và gõ **Password** vào.

6. Nhấp vào dòng 3, cột 1 và gõ **Confirm Password** vào.

7. Nhấp vào dòng 1, cột 1 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

8. Xác lập đặc tính Name là **txtLoginID**.

9. Nhấp vào dòng 1, cột 2 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

10. Xác lập đặc tính Name là **txtPassword**.

11. Xác lập đặc tính TextMode là **Password**.

12. Nhấp vào dòng 1, cột 3 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

13. Xác lập đặc tính Name là **txtConfirm**.

14. Xác lập đặc tính TextMode là **Password**.

15. Đặt con trỏ ngay bên phải của hộp văn bản txtConfirm và nhấp đôi control CompareValidator trong hộp công cụ.

16. Xác lập đặc tính ControlToCompare là **txtPassword**.

17. Xác lập đặc tính ControlToValidate là **txtConfirm**.

18. Xác lập đặc tính ErrorMessage là 'Password' and 'Confirm Password' Must Match.

19. Định vị con trỏ ngay dưới bảng.

20. Nhấp đôi control ValidationSummary để thêm control này vào ngay dưới bảng.

21. Nhấp đôi control Button để thêm Button vào ngay sau control Validation Summary.

22. Xác lập đặc tính Text là **Submit**.

23. Xác lập đặc tính Name là **btnSubmit**.

Bây giờ bạn có thể chạy project này để kiểm tra nó. Phải chắc chắn đổi Start Page sang Web Form mới bạn vừa tạo. Nếu bạn nhập hai giá trị khác nhau vào hai hộp văn bản Password, bạn sẽ thấy thông báo lỗi xuất hiện trong control Validation Summary.

Control hữu hiệu hóa biểu thức thông thường

Control RegularExpressionValidator cung cấp các khả năng so khớp mẫu rất mạnh mà bạn có thể dùng để chắc chắn là dữ liệu được nhập chính xác vào một control HTML cá biệt. RegularExpressionValidator sẽ gửi file JavaScript (.JS) xuống phía client với trình phân tách biểu thức thông thường. Trình phân tách này rất có hiệu quả và có thể kiểm tra nhiều kiểu biểu thức chuẩn. Một danh sách các biểu thức chuẩn có thể được kiểm tra được trình bày ở bảng 1.

ValidationExpressions
Custom
French Phone Number
French Postal Code
German Phone Number

German Postal Code
Internet E-Mail Address
Internet URL
Japanese Phone Number
Japanese Postal Code
P.R.C. Phone Number
P.R.C. Postal Code
P.R.C. Social Security Number (ID Number)
U.S. Phone number
U.S. Social Security Number
U.S. ZIP Code

Bảng 1: *Danh sách các biểu thức hữu hiệu hóa chuẩn.*

Như bạn thấy đây là một danh sách khá nhiều các kiểu biểu thức chuẩn mà bạn có thể sử dụng ngay. Bảng các biểu thức này giúp cho bạn không phải tự xây dựng các kiểu biểu thức thông thường này.

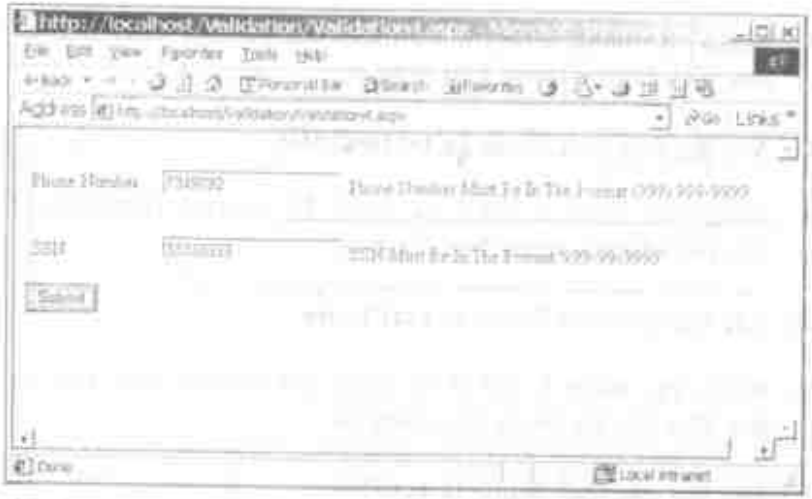
Các biểu thức thông thường có định dạng riêng và một số kí tự bạn phải đặt vào theo thứ tự nào đó để có mask được áp dụng cho dữ liệu nhập. Thí dụ, định dạng số điện thoại của Mỹ như sau:

```
(\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}
```

Số thẻ an sinh xã hội của Mỹ như sau:

```
\d{3}-\d{2}-\d{4}
```

Hình 5 bên dưới trình bày nhập liệu số điện thoại và số thẻ an sinh xã hội (SSN – Social Security Number) của Mỹ bằng cách dùng hai control Regular Expression Validator.



Hình 5: Trình soạn thảo biểu thức thông thường sẽ bảo đảm là nhập liệu theo đúng các định dạng đã định nghĩa sẵn.

Các bước tạo các biểu thức thông thường

1. Để tạo form được trình bày ở hình 5, thực hiện các bước sau.

2. Thêm một Web Form mới bằng cách lựa Project | menu Add Web Form.

3. Đặt tên là **RegExp.aspx**.

4. Thêm một bảng bằng cách lựa Table | Insert | Table. Xác lập số dòng là 2 và số cột là 2. Nhấp nút OK.

5. Nhấp vào dòng 1, cột 1 và gõ **Phone Number** vào.

6. Nhấp vào dòng 2, cột 1 và gõ **SSN** vào.

7. Nhấp vào dòng 1, cột 1 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

8. Xác lập đặc tính Name là **txtLoginID**.

9. Nhấp vào dòng 1, cột 2 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

10. Xác lập đặc tính Name là **txtPhone**.

11. Nhấp vào dòng 1, cột 3 và nhấp đôi control Text Box từ tab Web Forms trong hộp công cụ.

12. Xác lập đặc tính Name là **txtSSN**.

13. Đặt con trỏ của bạn ngay bên phải của hộp văn bản txtPhone và nhấp đôi control RegularExpressionValidator trong hộp công cụ.

14. Xác lập đặc tính ControlToValidate là **txtPhone**.

15. Xác lập đặc tính ErrorMessage là Phone Number Must Be In The Format (999) 999-9999.

16. Nhấp đặc tính ValidationExpression và nhấp nút với dấu ba chấm (...) bên trong cửa sổ đặc tính. Điều này sẽ hiển thị hộp thoại mà bạn có thể chọn **U.S. Phone Number**. Nhấp OK sau khi tô sáng chọn lựa này.

17. Đặt con trỏ ngay bên phải của hộp văn bản txtSSN và nhấp đôi control RegularExpressionValidator trong hộp công cụ.

18. Xác lập đặc tính ControlToValidate là **txtSSN**.

19. Xác lập đặc tính ErrorMessage là SSN Number Must Be In The Format 999-99-9999.

20. Nhấp đặc tính ValidationExpression và nhấp nút có dấu ba chấm (...) bên trong cửa sổ đặc tính. Điều này sẽ hiển thị hộp thoại mà bạn có thể chọn **U.S. Social Security Number**. Nhấp OK sau khi tô sáng chọn lựa này.

21. Đặt con trỏ vào dòng bên dưới Table.

22. Nhấp đôi control Button để thêm Button ngay sau bảng.

23. Xác lập đặc tính Text là **Submit**.

24. Xác lập đặc tính Name là **btnSubmit**.

Bây giờ bạn sẽ có thể chạy project này và kiểm tra nó. Chắc chắn phải đổi Start Page sang Web Form mới bạn vừa tạo. Thử nhập các định dạng số thẻ an sinh xã hội và số điện thoại hợp lệ và không hợp lệ để xem những gì được hiển thị bên trong các control hữu hiệu hóa.

Control hữu hiệu hóa tùy biến

Đôi khi bạn không thể sử dụng một trong các control hữu hiệu hóa chuẩn mà .NET cung cấp. Thí dụ, không có control "Valid Date Validation". Trong các trường hợp này, bạn sẽ phải tạo các thủ tục hữu hiệu hóa tùy biến riêng. Bạn có thể dùng control CustomValidator để viết mã này. Bạn sẽ phải viết hai thủ tục cho mỗi control CustomValidator bạn sử dụng. Một thủ tục sẽ cho phía client nếu trình duyệt hỗ trợ scripting phía client. Một thủ tục sẽ thực hiện cùng tính năng, nhưng sẽ chạy trên server trong trường hợp trình duyệt không hỗ trợ scripting.

***Hình 6:** Control hữu hiệu hóa tùy biến cho phép bạn viết script riêng để xử lý các trường hợp đặc biệt.*

Bây giờ bạn sẽ biết thủ tục tạo các kiểu form này trong ứng dụng web. Đặt tên hộp văn bản là **txtDate** trên form này. Một khi bạn đã tạo form ở trên, bạn sẽ phải xác lập các đặc tính sau trên control CustomValidator.

1. Xác lập đặc tính Name là **cvalDate**.
2. Xác lập đặc tính ControlToValidate là **txtDate**.
3. Xác lập đặc tính ClientValidationFunction là **ValidDate**.
4. Xác lập đặc tính ErrorMessage là **Invalid Date Format**.

5. Viết mã sau vào sự kiện ServerValidate cho control CustomValidator.

```
Private Sub cvalDate_ServerValidate( _
 ByVal source As System.Object, _
 ByVal args As _
 System.Web.UI.WebControls.ServerValidateEventArgs) _
 Handles cvalDate.ServerValidate
    If IsDate(args.Value) Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

Sự kiện này sẽ kích hoạt khi người dùng đệ trình form để hữu hiệu hóa. Giá trị của hộp văn bản cần hữu hiệu hóa được chuyển vào đặc tính Value của đối số *args* của thủ tục sự kiện này.

Sau khi viết mã phía server, bạn phải viết mã phía client cho một trình duyệt bất kỳ hỗ trợ DHTML. Thêm script sau vào nơi bất kỳ bên trong <BODY> của file .ASPX.

```
<SCRIPT language="VBScript">
Sub ValidDate(source, arguments)
   If IsDate(arguments.value) Then
      arguments.IsValid=True
   Else
      arguments.IsValid=False
   End If
End Sub
</SCRIPT>
```

Bây giờ bạn sẵn sàng chạy thử form này. Tiến hành và chạy nó để xem mọi thứ móc nối có chính xác hay không.

Tóm tắt

Các control hữu hiệu hóa mà .NET cung cấp sẽ giúp chúng ta tạo kinh nghiệm phong phú hơn cho người dùng khi sử dụng các ứng dụng web. Số lượng mã bạn phải viết vào các ứng dụng ASP giảm đáng kể khi bạn bắt đầu dùng các Web Form và các control hữu hiệu hóa này.

Câu hỏi ôn tập

1. Trình duyệt nào được coi là trình duyệt "up-level" tiêu biểu?

2. Bạn sẽ dùng control hữu hiệu hóa nào để chắc chắn người dùng nhập liệu bất kỳ dữ liệu nào vào một control?

3. Bạn sẽ dùng control nào nếu bạn cần có một mask nhập liệu?

4. Bạn sẽ dùng control nào nếu bạn cần bảo đảm các giá trị trong hai control khác nhau so khớp?

5. Hai đặc tính nào ở trên mỗi control hữu hiệu hóa?

Bài tập

- Tạo mỗi mẫu ở chương này.

Trả lời câu hỏi ôn tập

1. IE 4.0 và phiên bản cao hơn.
2. RequiredFieldValidator.
3. RegularExpressionValidator.
4. CompareValidator.
5. ControlToValidate và ErrorMessage.

Chương 28

Control liên kết Web Form

- Tìm hiểu cách nạp DataGrid trên Web Form với dữ liệu.
- Tìm hiểu cách nạp ComboBox trên Web Form với dữ liệu.

Liên kết tên các Web Form

Việc liên kết dữ liệu liên quan tới tiến trình tạo nguồn dữ liệu và tự động cung cấp nguồn dữ liệu đó cho (các) control trên một form. Các Web Form sử dụng ADO.NET dưới các vỏ bọc để thực hiện liên kết dữ liệu. Với việc liên kết dữ liệu, bạn không cần viết mã cụ thể để lập thể nghiệm kết nối và tạo dataset (như bạn thực hiện với form ngưng liên kết). .NET framework viết mã ADO.NET cần thiết cho bạn.

Các Web Form cho phép bạn liên kết dễ dàng với hầu hết cấu trúc nào chứa dữ liệu. Điều này có nghĩa là bạn có thể liên kết với các kho lưu trữ dữ liệu truyền thống, chẳng hạn như dữ liệu được lưu trữ trong Microsoft Access hoặc bảng Microsoft SQL

Server hay với kết quả dữ liệu đọc từ một file, được chứa trong các control khác hoặc được lưu trữ trong một mảng. Dữ liệu đi vào cấu trúc như thế nào không quan trọng đối với các mục đích ở chương này.

Sau khi bạn liên kết một form với dữ liệu, bạn có thể liên kết các control trên form với các phần tử dữ liệu riêng biệt. Một số việc liên kết dữ liệu thông dụng nhất bạn sẽ thực hiện trên WebForm sẽ là liên kết với control DataGrid, ListBox và ComboBox. Đó là những gì bạn sẽ tìm hiểu cách thực hiện ở chương này.

DataGrid liên kết dữ liệu

Việc sử dụng liên kết dữ liệu cơ bản nhất sẵn có trong môi trường .NET để hiển thị nội dung bảng CSDL trong một DataGrid. Một điển hình bao gồm một số bước cơ bản sau (không có chi tiết):

- Xây dựng một Web Form.
- Tạo và cấu hình dataset bạn muốn liên kết một control trên form.
- Thêm control DataGrid vào form và liên kết nó với dữ liệu.

Thí dụ về một form có kết quả từ tiến trình này xuất hiện ở hình 1.

***Hình 1:** Kết quả tạo một Web Form liên kết dữ liệu đơn giản.*

Xây dựng form mẫu

Sau là các bước chi tiết cần thiết để xây dựng form mẫu.

1. Lựa File ➢New➢Project. Hộp thoại New Project xuất hiện.
2. Lựa Project Type từ hiển thị cây ở bên trái hộp thoại. Đối với thí dụ này, lựa Visual Basic Projects.
3. Lựa ASP.NET Web Application từ danh sách các template ở bên phải của hộp thoại.
4. Xác lập Name là **DataBoundWeb** và nhấp OK để tạo project và thư mục ảo trong IIS.
5. Đặt lại tên WebForm1.aspx là **DataGridBound**.

Tạo và cấu hình Dataset

Sau khi bạn tạo project và web form, bạn sẵn sàng tạo và cấu hình dataset nằm dưới web form. Dataset là cache trong bộ nhớ bao gồm các bảng, các quan hệ và các ràng buộc. Mỗi bảng trong dataset có một tập hợp các cột và dòng. Vì các dataset nhận biết cả trạng thái ban đầu và hiện hành của chúng, nên chúng có thể truy tìm các thay đổi xảy ra. Dữ liệu trong dataset được coi là có thể cập nhật được.

Sử dụng wizard DataAdapter

Để tạo một kết nối với CSDL bạn sẽ dùng để truy tìm dữ liệu, bạn phải sử dụng thành phần OleDbDataAdapter được định vị dưới tab Data của Toolbox. Sau khi bạn kéo thành phần này vào web form, nó sẽ đưa bạn đi qua một loạt các bước để cấu hình câu lệnh SELECT và so sánh đối tượng sẽ giúp bạn xây dựng lớp DataSet. DataAdapter chỉ thực hiện việc truy tìm từ CSDL. DataSet là những gì bạn sẽ dùng để liên kết với control Data Grid.

1. Lựa OleDbDataAdapter từ tab Data của Toolbox.

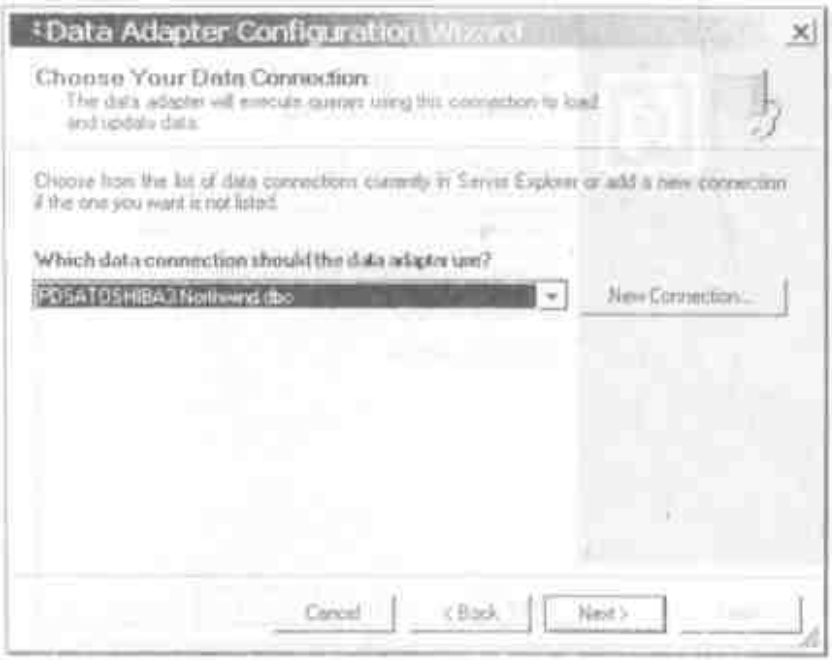
2. Kéo và thả nó vào web form.

3. Bây giờ Data Adapter Configuration Wizard sẽ khởi động.

4. Màn hình đầu tiên được trình bày ở hình 2, không có gì khác hơn là một màn hình thông tin. Chỉ cần nhấp Next để bỏ qua màn hình này.

***Hình 2:** Màn hình Welcome.*

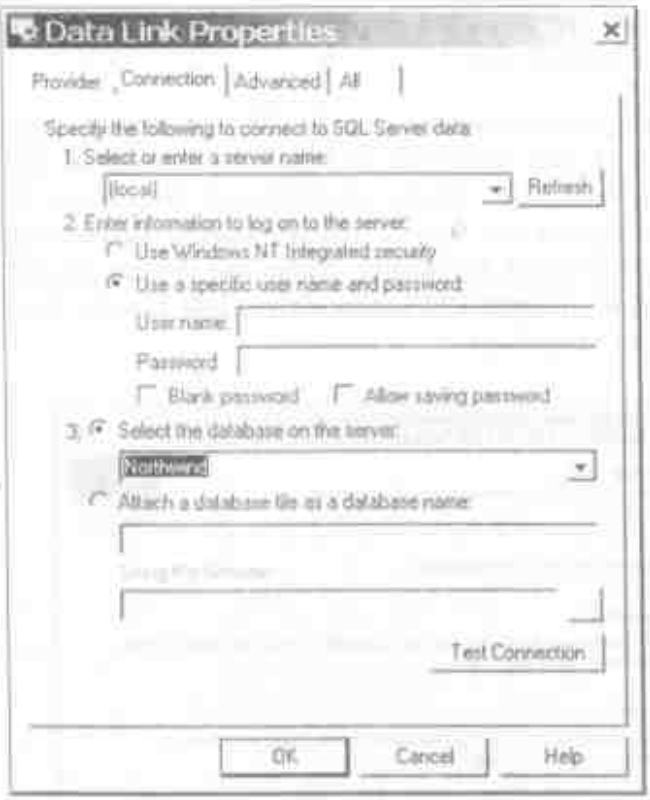
5. Nhấp Next để xem màn hình sẽ nhắc bạn về kết nối dữ liệu sử dụng. Nếu bạn chưa thiết lập kết nối dữ liệu, bạn sẽ phải nhấp nút New Connection.



Hình 3: Chọn kết nối dữ liệu.

Nếu bạn chọn nhấp nút New Connection, bạn sẽ thấy một màn hình như ở hình 4. Rất có thể bạn đã biết màn hình này vì nó thường được dùng trong các ứng dụng ADO. Trên màn hình này là nơi bạn sẽ lựa nhà cung cấp dữ liệu mà bạn muốn sử dụng và sau đó điền vào thông tin thích hợp về cách kết nối với nhà cung cấp dữ liệu đó.

Trong thí dụ được trình bày ở hình 4, bạn đã chọn nhà cung cấp SQL Server và chọn server cục bộ của bạn.



Hình 4: *Hộp thoại Data Link Properties cho phép bạn chỉ định thông tin kết nối cho dữ liệu nằm dưới form liên kết.*

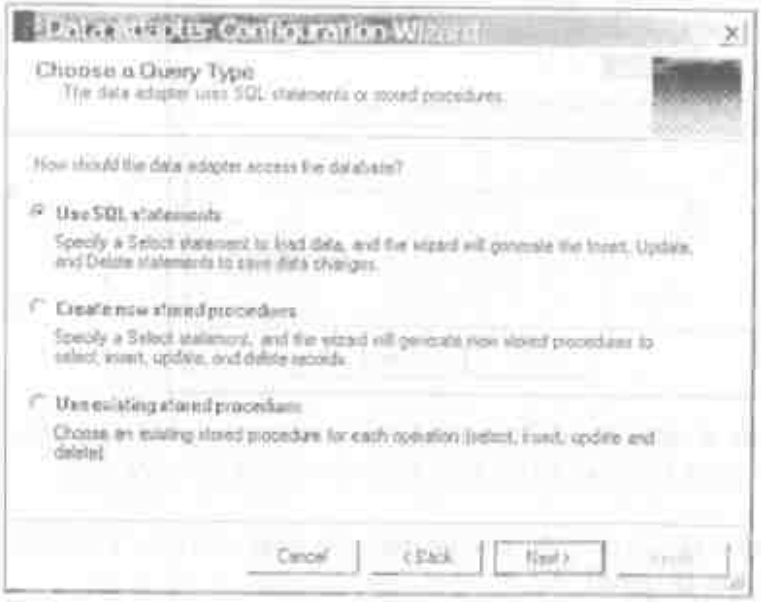
Đối tên server để trỏ tới tên của SQL Server. Trong thí dụ này, nó là (local).

1. Thay đổi User name và Password cần thiết để đăng nhập vào SQL Server này.

2. Lựa CSDL Northwind vì đây là nơi có bảng Products để truy tìm thông tin.

3. Nhấp OK để trở về wizard và nhấp nút Next để di chuyển tới bước wizard kế tiếp.

4. Bước kế tiếp của wizard cho phép bạn chỉ định Query Type (xem hình 5). Bạn có thể lựa Use SQL Statements để xây dựng câu lệnh SQL, Create New Stored Procedures để wizard phát sinh các thủ tục lưu trữ thường xuyên hoặc Use Existing Stored Procedures để lựa các thủ tục lưu trữ đã được định vị trong CSDL của server. Lựa các câu lệnh Use SQL và nhấp Next.



Hình 5: *Chỉ định kiểu truy vấn mà form liên kết sẽ dựa vào.*

5. Gõ câu lệnh SQL hoặc nhấp Query Builder để wizard xây dựng câu lệnh SQL cho bạn. Đối với thí dụ này, gõ câu lệnh SQL sau:

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock
FROM Products
```

6. Nhấp Next để tiến hành tới bước cuối cùng của wizard và nhấp Finish để hoàn tất tiến trình. Lưu ý, đối tượng OleDbConnection và OleDbDataAdapter có tên là OleDbConnection1 và OleDbDataAdapter1 xuất hiện trong Tray của form.

Đối tượng OleDbConnection1 chứa thông tin về cách truy xuất CSDL đã lựa. Đối tượng OleDbDataAdapter1 chứa truy vấn định nghĩa các bảng và các cột trong CSDL mà bạn muốn truy xuất.

Lưu ý:

Trong thí dụ, OleDbDataAdapter được lựa từ hộp công cụ vì bạn có thể sẽ sử dụng CSDL nào đó khác hơn SQL Server. Nếu bạn truy xuất được SQL Server, bạn nên sử dụng các thành phần Sql* vì chúng có hiệu quả hơn. Các bước bạn thực hiện sẽ hoàn toàn giống nhau.

Tạo lớp Dataset

Như đã nói ở trên, tự thân DataAdapter không thể được liên kết với control trên một form. Bạn cần DataSet cho nó. Visual Studio.NET sẽ giúp bạn phát sinh lớp DataSet dựa trên truy vấn bạn đã tạo ở đối tượng DataAdapter. Để phát sinh lớp dataset mới này, thực hiện các bước sau.

1. Lựa Data | Generate Dataset từ menu VS.NET.

2. Bạn sẽ thấy hộp thoại Generate Dataset xuất hiện như ở hình 6.

3. Nhấp nút tùy chọn New.

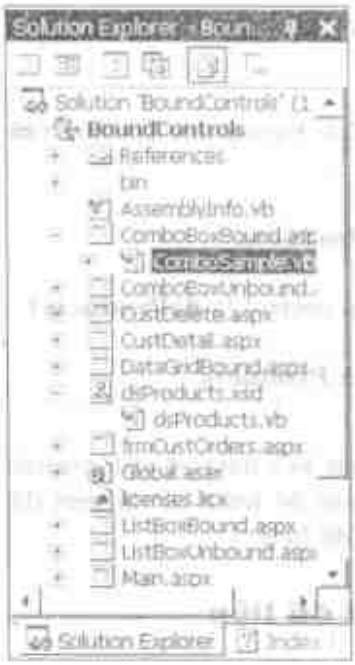
4. Gõ tên dsProducts vào.

5. Nhấp nút OK để phát sinh lớp và file định nghĩa giản đồ DataSet.

***Hình 6:** Hộp thoại Generate Dataset cho phép bạn chỉ định tên và các thuộc tính khác của dataset bạn sẽ phát sinh.*

Sau khi hoàn tất bước này, bạn sẽ thấy một control mới, dsProducts1 trong Tray cho web form này. Control mới này là một tham chiếu với file dsProducts.xsd cũng đã được thêm vào project của bạn. File dsProducts.xsd là một định nghĩa giản đồ XML chứa định nghĩa đầy đủ cho bảng và các cột của câu lệnh

SQL bạn đã gõ vào trước đây. Còn có một lớp đằng sau file XSD (XML schema definition – Định nghĩa giản đồ XML) này mà bạn không thể thấy trừ khi bạn lựa Project>Show All Files từ menu. Sau đó bạn có thể nhấp vào dấu cộng xuất hiện ở đằng sau file dsProducts.xsd để xem file dsProducts.vb, như được trình bày ở hình 7.



***Hình 7:** File XML schema definition (XSD) có một module lớp chứa mã để nạp dataset vào bộ nhớ.*

Lớp dsProducts.vb có các đặc tính tương ứng dataset thực và các đặc tính tương ứng với mỗi cột được chỉ định trong câu lệnh SQL. Mặc dù bạn không phải làm gì với lớp này, nhưng biết nó ở đó rất hữu dụng.

Thêm control Data Grid để hiển thị dữ liệu

Bây giờ bạn sẵn sàng tạo một web form để hiển thị dữ liệu được chứa trong dataset. Trong thí dụ này, bạn sẽ thêm control DataGrid vào web form. Control khung lưới dữ liệu sẽ hiển thị dữ liệu được chứa trong dataset. Thực hiện các bước sau:

1. Nhấp tab ở trên đầu cửa sổ để hiển thị web form DataBoundGrid.aspx.

2. Kéo control DataGrid từ tab Web Forms của Toolbox vào form.

3. Xác lập đặc tính Name là **grdProducts**.

4. Xác lập đặc tính DataSource của control là **dsProducts1**.

5. Xác lập đặc tính DataMember là **Products**.

Các bước bạn vừa thực hiện đã liên kết dataset với control khung lưới dữ liệu. Chỉ còn một bước nữa để hoàn tất là xem dữ liệu xuất hiện trong control khung lưới dữ liệu.

Cư trú control Data Grid với dữ liệu

Mặc dù control khung lưới dữ liệu được liên kết với dataset, nhưng dataset ấy không tự động được cư trú khi form được nạp. Sử dụng thủ tục sự kiện Page_Load của web form để cư trú control khung lưới dữ liệu với dữ liệu khi web form được nạp.

```
Private Sub Page_Load(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    OleDbDataAdapter1.Fill(DsProducts1)
```

```
    grdProducts.DataBind()
End Sub
```

Trong thủ tục sự kiện Load, bạn điền dataset bằng cách dùng phương thức Fill của đối tượng OleDbDataAdapter1 bạn đã tạo trước đây. Bạn gửi tới phương thức này lớp DataSet mới bạn đã tạo. Dữ liệu sẽ được điền vào theo câu lệnh SQL bạn đã xây dựng trước đây. Cuối cùng, bạn cho DataGrid thực hiện phương thức DataBind của nó để điền khung lưới với dữ liệu từ lớp dsProducts1.

Tại điểm này, bạn sẵn sàng để xem dữ liệu từ bảng Products hiển thị trong control khung lưới dữ liệu. Chỉ còn một bước nữa để hoàn tất trước khi bạn có thể chạy form này. Vì bạn đã thay đổi tên của form, bạn phải báo cho project form Startup cho project này là gì.

1. Nhắp vào file DataBoundGrid.aspx trong cửa sổ Solution Explorer.

2. Nhắp nút chuột phải và chọn Set as Start Page từ menu.

3. Ấn **F5** để chạy project này và nếu bạn đã thực hiện mọi thứ chính xác, thì bây giờ bạn sẽ thấy dữ liệu sản phẩm bên trong control khung lưới dữ liệu giống như màn hình được trình bày ở hình 1.

Dữ liệu liên kết Combo Box

Việc sử dụng các thành phần lập sẵn hoạt động tốt, nhưng khó thấy được mọi thứ sẽ tiến hành đàng sau các phông. Thực ra, các wizard sẽ xây dựng một lớp hoàn chỉnh để bổ sung, soạn thảo, xóa và truy tìm dữ liệu. Điều này thường đề cập quá nhiều khi thảo luận việc nạp một hộp combo hoặc hộp danh sách.

Trong các trường hợp này, bạn tự viết mã để nạp các kiểu control sẽ có ý nghĩa hơn.

1. Tạo một web form mới trong project này.
2. Đặt tên nó là CombBoxBound.aspx.
3. Xác lập nó là Start page.
4. Thêm control ComboBox vào web form này bằng cách nhấp đôi control ComboBox trong hộp công cụ.
5. Xác lập đặc tính Name là **cboCust**.
6. Nhấp đôi vào form và thêm mã sau vào thủ tục sự kiện Page_Load.

```
Private Sub Page_Load(ByVal Sender As System.Object, _
 ByVal e As System.EventArgs) Handles MyBase.Load
    If Page.IsPostBack Then
       ComboLoad()
    End If
End Sub
```

Trong thủ tục này bạn sẽ gọi một thủ tục sẽ nạp hộp combo bạn vừa tạo. Bây giờ ngay dưới thủ tục End Sub của Page_Load, thêm mã sau.

```
Private Sub ComboLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User Id=sa"
```

```
        strSQL = "SELECT CustomerID, CompanyName "
        strSQL &= " FROM Customers ORDER BY CompanyName"

        oAdapter = New OleDb.OleDbDataAdapter(strSQL,
    strConn)
        oTable = New DataTable()

        oAdapter.Fill(oTable)

        With cboCust
            .DataSource = oTable
            .DataTextField = "CompanyName"
            .DataValueField = "CustomerID"

            .DataBind()
        End With
    End Sub
```

Bạn tạo đối tượng DataTable một chuỗi kết nối, câu lệnh SELECT để truy tìm hai cột từ bảng Customers và đối tượng DataAdapter. Sau đó bạn sử dụng phương thức Fill của DataAdapter để cư trú DataTable với dữ liệu được truy tìm từ câu lệnh SELECT. Một khi DataTable được cư trú, bạn có thể xác lập một vài đặc tính trên hộp combo để cho nó tự động nạp dữ liệu đó.

Xác lập đặc tính DataSource là đối tượng DataTable. Xác lập đặc tính DataTextField là tên file "CompanyName". Đặc tính này sẽ sử dụng dữ liệu đến từ tập các bản ghi để hiển thị trong phần danh sách của hộp combo. Xác lập DataValueField là "CustomerID". Đặc tính này sẽ lưu giữ dữ liệu ID khách hàng này trong thuộc tính "value" ở câu lệnh <OPTION> để được phát sinh trong HTML. Để khởi tạo việc nạp dữ liệu bạn sẽ gọi ra phương thức DataBind trên hộp combo.

Bây giờ bạn có thể chạy project, bạn sẽ thấy hộp combo được điền với các tên khách hàng. Bạn cũng phải làm một khung xem mã nguồn HTML từ trình duyệt của bạn để xem HTML đã phát sinh.

Nạp List box

Bạn cũng có thể dùng cùng mã bạn đã vừa tạo để nạp control ListBox trên Web Form. Không có sự khác nhau nào khi nạp control này hay control kia.

Điền Combo Box thủ công

Bạn không phải dùng dữ liệu liên kết để nạp hộp combo hoặc hộp danh sách. Thay vào đó, bạn có thể lập trình control web server Combo Box trực tiếp để tạo tất cả các mục trong hộp combo.

1. Tạo một web form mới trong project của bạn.
2. Xác lập Name là ComboBoxUnbound.aspx.
3. Thêm control ComboBox vào form này.
4. Xác lập đặc tính Name là **cboCust**.
5. Nhấp đôi vào WebForm để tới thủ tục sự kiện Page_Load. Thêm mã sau.

```
Private Sub Page_Load(ByVal Sender As System.Object, _
 ByVal e As System.EventArgs) Handles MyBase.Load
    If Page.IsPostBack Then
        ComboLoad()
    End If
End Sub
```

Trong thủ tục ComboLoad bạn sẽ viết để sử dụng lớp ListItem nạp vào ComboBox. Bạn sẽ dùng lớp DataReader để nạp dữ liệu. Khi bạn đọc mỗi mục từ lớp DataReader bạn sẽ tạo một thể

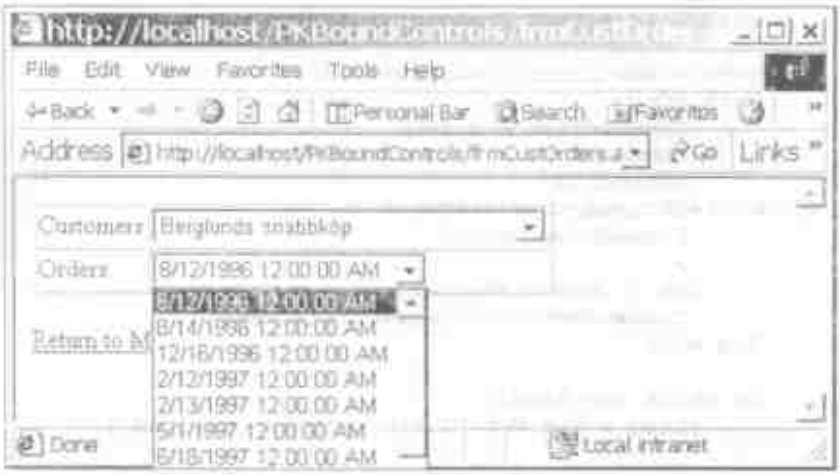
nghiệm mới của lớp ListItem và nạp đặc tính Value và Text của nó với hai cột của bảng Customers. Sau đó bạn sẽ chuyển đối tượng ListItem mới này tới phương thức cboCust.Items.Add.

```
Private Sub ComboLoad()
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim oItem As Web.UI.WebControls.ListItem
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User id=sa"

    strSQL = "SELECT CustomerID, CompanyName "
    strSQL &= " FROM Customers ORDER BY CompanyName"

    oCmd = New OleDb.OleDbCommand()
    With oCmd
        .CommandText = strSQL
        .CommandType = Data.CommandType.Text
        .Connection = New
OleDb.OleDbConnection(strConn)
        .Connection.Open()

        oDR = .ExecuteReader( _
         CommandBehavior.SequentialAccess)
    End With

    Do While oDR.Read()
        oItem = New Web.UI.WebControls.ListItem()

        oItem.Value = oDR.Item("CustomerID").ToString()
        oItem.Text = oDR.Item("CompanyName").ToString()

        cboCust.Items.Add(oItem)
    Loop
    oDR.Close()
    oCmd.Connection.Close()
End Sub
```

Customers và Orders

Ở phần này đầu tiên bạn sẽ nạp hộp combo customer (các khách hàng) giống như bạn đã làm ở phần trước. Sự khác nhau là bây giờ bạn sẽ viết một số mã vào thủ tục sự kiện SelectedIndexChanged của hộp combo customer để nạp các order (đơn đặt hàng) cho khách hàng đó.

Tạo form được trình bày ở hình 8 bằng cách thêm Table và hai hộp combo vào form này. Đặt tên các hộp combo là **cboCust** và **cboOrders** tương ứng.



***Hình 8:** Bạn có thể nạp nội dung của một hộp combo dựa trên giá trị bên trong một hộp combo khác.*

Đầu tiên bạn sẽ nạp các khách hàng vào thủ tục sự kiện Page_Load. Nhấp đôi vào web form để làm cho thủ tục này giống như sau.

```
Private Sub Page_Load(ByVal Sender As System.Object, _
```

```
ByVal e As System.EventArgs) Handles MyBase.Load
    If Not Page.IsPostBack Then
        CustLoad()
    End If
End Sub
```

Bạn phải chắc chắn sử dụng biến IsPostBack để bảo đảm bạn không nạp các khách hàng mỗi lần form được gọi lại. Bạn phải nhớ là mỗi lần người dùng nhấp vào hộp combo customers và sự kiện được kích hoạt trên phía server, điều này sẽ kích hoạt thủ tục sự kiện SelectedIndexChanged, nhưng cũng sẽ khiến cho thủ tục sự kiện Page_Load kích hoạt. Đặc tính Page.IsPostBack tự động được xác lập là True sau khi nạp web form này lần đầu tiên.

Thủ tục CustLoad được liệt kê bên dưới, nhưng bạn sẽ thấy thủ tục này giống như thủ tục bạn đã tạo trước đây ở chương này. Bạn chỉ cần chép và dán thủ tục ấy từ form kia vào thẳng form này.

```
Private Sub CustLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User id=sa"

    strSQL = "SELECT CustomerID, CompanyName "
    strSQL &= " FROM Customers ORDER BY CompanyName"

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oTable = New DataTable()

    oAdapter.Fill(oTable)

    With cboCust
        .DataSource = oTable
```

```
            .DataTextField = "CompanyName"
            .DataValueField = "CustomerID"
            .DataBind()
        End With
    End Sub
```

Kế tiếp bạn sẽ phải nhấp đôi hộp combo cboCust. Sau đó bạn sẽ thấy thủ tục sự kiện SelectedIndexChanged. Viết mã được trình bày bên dưới.

```
Private Sub cboCust_SelectedIndexChanged( _
 ByVal sender As Object, ByVal e As System.EventArgs) _
 Handles cboCust.SelectedIndexChanged
    Call OrdersLoad()
End Sub
```

Bây giờ bạn phải viết thủ tục OrdersLoad để nó lấy giá trị từ đặc tính cboCust.SelectedItem.Value và sử dụng nó như một phần của mệnh đề WHERE trong câu lệnh SELECT để truy tìm các đơn đặt hàng cho ID khách hàng được lựa.

```
Private Sub OrdersLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User id=sa"

    strSQL = "SELECT OrderID, OrderDate "
    strSQL &= "FROM Orders WHERE CustomerID = '"
    strSQL &= cboCust.SelectedItem.Value & "'"

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oTable = New DataTable()

    oAdapter.Fill(oTable)

    With cboOrders
```

```
            .Items.Clear()
            .DataSource = oTable
            .DataTextField = "OrderDate"
            .DataValueField = "OrderID"
            .DataBind()
        End With
    End Sub
```

Tóm tắt

Ở chương này bạn đã tìm hiểu cách điền DataGrid và các hộp Combo với dữ liệu bằng cách dùng đối tượng DataTable. Bạn cũng đã có thể sử dụng đối tượng DataSet như DataSource cho các control liên kết này. Việc viết mã hầu như cũng dễ dàng như sử dụng các wizard để tạo tất cả các thành phần khác nhau.

Câu hỏi ôn tập

1. Bạn dùng thành phần nào từ hộp công cụ để gọi ra DataAdapter Wizard?

2. Hai kiểu thành phần nào được phát sinh khi bạn phát sinh DataSet bằng cách dùng menu Data?

3. Bạn gọi ra phương thức nào trên control DataAdapter để nạp dataset đã phát sinh của mình với dữ liệu?

4. Bạn có thể dùng các đối tượng nào để nạp control liên kết dữ liệu?

5. Bạn dùng đối tượng nào để nạp hộp combo hoặc hộp danh sách thủ công?

Bài tập

- Tạo lại các bài tập ở chương này.

Trả lời câu hỏi ôn tập

1. SqlDataAdapter.
2. XSD và Class.
3. Fill.
4. DataTable, DataView và DataSet.
5. ListItem.

Chương 29

Sử dụng DataGrid của WebForm

- Tìm hiểu cách dùng DataGrid trên WebForm.
- Tìm hiểu cách định dạng dữ liệu trong control DataGrid.
- Tạo các siêu liên kết trong DataGrid.
- Hiển thị form chi tiết sản phẩm từ siêu liên kết.

Sử dụng DataGrid

Một trong các điểm mạnh về nền Microsoft.NET là khả năng tạo các ứng dụng Internet nhanh và dễ dàng. Một trong các công cụ sẽ được dùng để thực hiện tác vụ này là các WebForm và các control mà bạn có thể dùng trên các form này. Có một số control liên kết dữ liệu mà bạn có thể đặt trên các WebForm. Các control này chạy trên server, thế nhưng gửi mã HTML tới trình duyệt của client. Nếu bạn cần hiển thị một bảng thông tin trên

trang HTML, thì một trong các công cụ tốt nhất cho công việc này là DataGrid.

Tại sao cần thực hiện điều này?

Việc sử dụng control DataGrid, bạn sẽ có thể hiển thị một bảng thông tin với người dùng chỉ bằng một vài dòng mã. Bạn cũng có thể thêm định dạng, lập số trang và các siêu liên kết (hyperlink) vào khung lưới này chỉ với một ít mã nữa. Nếu bạn cố viết mã này bằng cách dùng các Active Server Page bình thường, thì bạn sẽ phải viết mã nhiều hơn đáng kể.

Ở chương này, bạn sẽ tìm hiểu cách liên kết DataGrid tới đối tượng DataSet của ADO.NET. Sau đó bạn sẽ thêm định dạng nào đó vào khung lưới này để hiển thị các trị số ở định dạng Currency. Kế tiếp bạn sẽ tìm hiểu cách thêm việc lập số trang vào khung lưới này để người dùng có thể đánh số trang bằng dữ liệu. Cuối cùng, bạn sẽ thêm một siêu liên kết vào một trong các cột trong khung lưới để có thể đưa người dùng tới trang chi tiết trình bày tất cả các thông tin về sản phẩm đã lựa. Hình 1 trình bày màn hình đầu tiên bạn sẽ tìm hiểu cách tạo nó khi đi qua các bước kế tiếp ở chương này.

Hình 1: *Dễ dàng tạo một bảng HTML bằng cách dùng control DataGrid.*

Nạp khung lưới với dữ liệu

Đầu tiên bạn sẽ tìm hiểu cách tạo một DataGrid đơn giản trên trang web và liên kết khung lưới đó với Dataset của ADO.NET. Sau khi thực hiện xong các bước này, trình duyệt của bạn sẽ hiển thị một trang giống như được trình bày ở hình 1. Thực hiện các bước sau để hoàn tất tác vụ này.

1. Mở Visual Studio.NET.
2. Lựa Web Application từ danh sách các Template.
3. Xác lập Name của project này là **DataGridSample**.
4. Nhấp nút OK.
5. Chắc chắn form WebForm1.aspx được hiển thị trong cửa sổ thiết kế.
6. Mở Toolbox và nhấp đôi control DataGrid ở tab WebForm. Điều này sẽ thêm control DataGrid vào file WebForm1.aspx.
7. Nhấp DataGrid để cho nó tiêu điểm, đi tới Properties Windows và xác lập đặc tính ID là **grdProducts**.
8. Mở Code Behind the Form và định vị thủ tục sự kiện Load cho trang này.
9. Thêm một lời gọi vào thủ tục GridLoad được gọi. Thủ tục này sẽ được tạo ở bước sau. Chắc chắn bạn đặt lời gọi bên trong câu lệnh If để bảo đảm thủ tục này chỉ được gọi một lần.

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

    If Not Page.IsPostBack Then
        Call GridLoad()
    End If

End Sub
```

Thủ tục sự kiện Page_Load kích hoạt bất kỳ khi nào trang nạp. Biến IsPostback được xác lập là True nếu đây là lần thứ hai

trang được gọi. Đó là cách form nhận biết nếu đây là lần đầu tiên trang sẽ nạp hoặc dù đây có phải là lần gọi lại hay không để xử lý một sự kiện nào đó trên trang.

Bây giờ tạo thủ tục GridLoad() sẽ được dùng để nạp DataGrid với dữ liệu từ bảng Products. Tới một dòng trống ngay sau End Sub của thủ tục Load và gõ mã sau vào.

```
Private Sub GridLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strSQL = SQLBuild()
    strConn = ConnectStringBuild()

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oTable = New DataTable()

    oAdapter.Fill(oTable)

    grdProducts.DataSource = oTable
    grdProducts.DataBind()
End Sub
```

Trong thủ tục GridLoad bạn sẽ dùng đối tượng OleDbDataAdapter để lưu trữ kết nối và các chuỗi SQL. Sau đó bạn gọi ra phương thức Fill bằng cách chuyển vào đối tượng DataTable. Phương thức Fill sẽ tạo kết nối với server, đệ trình SQL với server, sau đó xây dựng đối tượng DataTable.

Một khi bạn có đối tượng DataTable này trong bộ nhớ, bạn có thể xác lập đặc tính DataSource của grdProducts là đối tượng DataTable này. Điều này cho control DataGrid tất cả dữ liệu mà nó sẽ cần để tự biểu hiện trên trang HTML. Cuối cùng, bạn gọi ra phương thức DataBind trên DataGrid để cho nó tiến hành và biểu hiện trang HTML dựa trên các cột trong câu lệnh SQL.

Control DataGrid có thể liên kết với đối tượng DataTable, DataView hoặc DataSet.

Một số hàm bổ sung

Như bạn thấy ở listing trước, có hai hàm được gọi là SQLBuild và ConnectStringBuild. Bạn có thể tạo các hàm này ngay bên trong WebForm này hoặc bạn có thể tạo một module tách riêng bằng cách lựa Project|menu Add New Item, sau đó chọn template Module. Bạn nên đặt các thủ tục này vào Module nếu bạn định tạo các form khác có thể dùng cùng các hàm này. Trong hầu hết các ứng dụng CSDL, rất có thể bạn sẽ có một vài form cần dùng chuỗi kết nối, vì vậy ít nhất nên đặt hàm ConnectStringBuild vào một Module. Sau khi tạo module này, thêm hai hàm được đề cập sau vào module mới.

Các bước

1. Lựa Project|New Item từ menu Visual Studio.NET. Sau đó lựa template Module.

2. Đặt tên module là **modData.vb** và nhấp OK.

3. Gõ mã sau vào bên trong các câu lệnh Module..End Module ở file Module mới này mà bạn vừa tạo.

```
Public Function SQLBuild() As String
    Dim strSQL As String

    strSQL = "SELECT ProductID, ProductName, "
    strSQL &= " UnitPrice, UnitsInStock "
    strSQL &= " FROM Products ORDER BY ProductName "

    Return strSQL
End Function
```

Trong hàm SQLBuild(), bạn sẽ tạo một chuỗi SQL để lựa bốn cột từ bảng Products và cho trở lại chuỗi SQL. Hàm này cụ thể hơn cho một DataGrid bạn sẽ xây dựng một chút và thực sự không cần đặt nó vào một Module tách riêng, nhưng bạn không biết được khi nào bạn lại có thể cần câu lệnh SQL này.

Bây giờ gõ mã được trình bày bên dưới cho hàm ConnectStringBuild vào Module này.

```
Public Function ConnectStringBuild() As String
    Dim strConn As String

    strConn = "Provider=sqloledb"
    strConn &= ";Data Source=(local)"
    strConn &= ";Initial Catalog=Northwind"
    strConn &= ";User ID=sa"
    strConn &= ";Password="

    Return strConn
End Function
```

Phương thức ConnectStringBuild giả định bạn đã cài đặt SQL Server ở một nơi nào đó mà bạn có thể truy xuất được. Chuỗi nhà cung cấp ở mã trên có tham số Data Source được xác lập là (local), là tên máy cục bộ của bạn. Bạn có thể phải sửa đổi tham số này để trỏ tên của máy SQL Server. Bạn cũng có thể phải sửa đổi tham số User ID thành một User ID khác và có thể phải thay đổi tham số Password để có thể đăng nhập vào server này chính xác.

Nếu bạn không truy xuất được SQL Server, bạn có thể phải dùng CSDL Northwind Access. CSDL Northwind được phân phối với Microsoft Access. Nếu bạn muốn kết nối với CSDL Access này, bạn sẽ dùng chuỗi nhà cung cấp sau.

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Northwind\Northwind.mdb"
```

Đương nhiên, bạn sẽ phải đổi đường dẫn trên thành đường dẫn hợp lệ cho CSDL Northwind này.

Tại điểm này, bạn sẽ có thể chạy ứng dụng này và xem DataGrid thể hiện như thế nào trong một trình duyệt.

- Chạy project bằng cách ấn **F5**.

Nếu bạn đã điền vào tất cả thông tin chuỗi kết nối chính xác, bạn sẽ thấy một bảng HTML xuất hiện trong trình duyệt của bạn giống hình 1. Nếu bạn không có chuỗi kết nối chính xác, bạn có thể thấy một trang lỗi xuất hiện trong trình duyệt với một thông báo lỗi, chẳng hạn như "SQL Server does not exist or access denied". Nếu bạn thấy thông báo này, có nghĩa là bạn phải kiểm tra chuỗi kết nối, vì có lẽ bạn đã gõ điều gì đó sai.

Định dạng các cột trị số

Nếu bạn xem cột UnitPrice và UnitsInStock, thấy cả hai trị số đều vẫn còn được canh giữa. Rõ ràng là bạn sẽ cần canh phải các cột này và trong trường hợp của UnitPrice, nó phải định dạng là giá trị tiền tệ. Để thực hiện điều này, bạn phải thêm một số mã vào HTML và đàng sau DataGrid. Hình 2 trình bày một mẫu hoàn chỉnh thể hiện như thế nào khi bạn hoàn tất các bước sau.

***Hình 2:** Định dạng các cột trị số để bảng thể hiện tốt hơn.*

1. Hiển thị form frmProducts ở chế độ thiết kế và nhấp một lần vào control DataGrid.

2. Xác lập đặc tính AutoGenerateColumns là **False**. Nếu bạn không làm điều này, bạn sẽ có hai tập cột giống nhau mà bây giờ bạn sẽ định nghĩa.

3. Nhấp tab HTML của form WebForm1.aspx.

4. Định vị định nghĩa của DataGrid. Nó sẽ thể hiện giống dòng mã ASP.NET sau.

```
<asp:DataGrid id="grdProducts"
 runat="server"></asp:DataGrid>
```

Định nghĩa này cho DataGrid đã được phát sinh khi bạn vẽ DataGrid trên WebForm này. Giữa tag <asp:DataGrid> và </asp:DataGrid> là nơi bạn phải thêm một số câu lệnh XML định dạng để áp dụng định dạng vào khung lưới.

5. Thay đổi mã như nó được trình bày sau:

```
<asp:DataGrid id="grdProducts" runat="server"
 BorderStyle="None"                    GridLines="Vertical"
BorderWidth="1px"
 BorderColor="#999999"                   BackColor="White"
CellPadding="3"
 AutoGenerateColumns="False" AllowPaging="True">
   <FooterStyle
    ForeColor="Black"
BackColor="#CCCCCC"></FooterStyle>
   <HeaderStyle Font-Bold="True" ForeColor="White"
    BackColor="#000084"></HeaderStyle>
   <PagerStyle                        HorizontalAlign="Center"
ForeColor="Black"
    BackColor="#999999"
Mode="NumericPages"></PagerStyle>
   <SelectedItemStyle Font-Bold="True" ForeColor="White"
    BackColor="#008A8C"></SelectedItemStyle>
   <AlternatingItemStyle
    BackColor="Gainsboro"></AlternatingItemStyle>
   <ItemStyle ForeColor="Black"
    BackColor="#EEEEEE"></ItemStyle>
   <Columns>
      <asp:BoundColumn DataField="ProductID"
       HeaderText="Product ID"></asp:BoundColumn>
      <asp:BoundColumn DataField="ProductName"
       HeaderText="Product
Description"></asp:BoundColumn>
      <asp:BoundColumn DataField="UnitPrice"
```

```
        HeaderText="Unit Price"
DataFormatString="{0:C}">
        <ItemStyle HorizontalAlign="Right">
        </ItemStyle>
      </asp:BoundColumn>
      <asp:BoundColumn DataField="UnitsInStock"
        HeaderText="Units In Stock"
        DataFormatString="{0:N}">
        <ItemStyle HorizontalAlign="Right">
        </ItemStyle>
      </asp:BoundColumn>
    </Columns>
</asp:DataGrid>
```

Ở mã trên bạn khai báo bốn đặc tính BoundColumn. Mỗi đặc tính này tương ứng với các cột bạn đã lựa trong chuỗi SQL. Trong hầu hết các trường hợp, bạn chỉ cần liệt kê thuộc tính DataField và HeaderText cho mỗi cột. Tuy nhiên, khi bạn cần áp dụng định dạng bạn phải dùng thêm một số thuộc tính.

Đối với cột UnitPrice, bạn thêm thuộc tính "ItemStyle HorizontalAlign" và xác lập giá trị là "Right". Điều này sẽ canh phải số bên trong cột. Hơn nữa, bạn thêm thuộc tính DataFormatString và xác lập giá trị là "{0:C}". Giá trị định dạng này cho DataGrid biết lấy giá trị đầu tiên, là giá trị nhập vào từ trường UnitPrice và áp dụng kiểu (C)urrency vào nó. Cũng còn có một số định dạng dữ liệu khác mà bạn có thể áp dụng vào các cột này trong control DataGrid.

Các chuỗi định dạng dữ liệu

Nếu bạn tìm trong trợ giúp trực tuyến "DataFormatString", bạn sẽ thấy có bảy định dạng khác nhau bạn có thể áp dụng. Dựa vào giá trị 12.75, bảng 1 cho bạn mô tả và thí dụ về mỗi định dạng khác nhau bạn có thể áp dụng vào con số này.

Kí tự định dạng	Mô tả	Thí dụ
C	Tiền tệ	$12.75
D	Thập phân	12.75
E	Khoa học (số mũ)	1.275000E+001
F	Cố định	12.75
N	Số	12.75
X	Thập lục phân	0xC

Bảng 1: *Sử dụng các kí tự định dạng ở trên cho dữ liệu số.*

Cho phép lập số trang

Trong bảng Products, có gần 77 dòng dữ liệu. Khi bạn hiển thị bảng ở mã trước đây, tất cả 77 dòng đều được kết xuất vào một bảng, vì vậy người dùng phải cuộn xuống để xem tất cả các dòng. Thay vì kết xuất tất cả dữ liệu, bạn nên trình bày mỗi lần chỉ một vài dòng. Sau đó bạn có thể cho người dùng di chuyển từ trang dữ liệu này tới trang dữ liệu khác bằng cách nhấp kiểu siêu liên kết tới và lui.

Bằng cách dùng DataGrid, kiểu thao tác này khá đơn giản. Bạn có thể chỉ định nơi bạn muốn đặt các siêu liên kết lập số trang trên đỉnh, dưới đáy hoặc cả hai. Bạn có thể chỉ định nếu bạn chỉ có kiểu siêu liên kết Next và Previous hoặc nếu bạn muốn sử dụng các số trang. Hình 3 trình bày một thí dụ về cách sử dụng các số trang, nhưng trong các bước sau bạn sẽ tìm hiểu cách dùng cả kiểu Next và Previous lẫn kiểu số trang.

***Hình 3:** Bạn có thể tạo kiểu lập số trang ở các định dạng khác nhau.*

Tạo kiểu lập số trang Next và Previous

Bằng cách xét các đặc tính của control DataGrid, bạn sẽ thấy một đặc tính được gọi là AllowPaging. Bạn sẽ nghĩ là chỉ bằng cách xác lập đặc tính này là True, cơ chế lập số trang sẽ hoạt động. Tuy nhiên, bạn phải viết một ít mã trong thủ tục sự kiện PageIndexChanged để khiến cho việc lập số trang hoạt động. Lý do là bạn phải báo cho DataGrid tải lại dữ liệu chỉ cho một trang đó.

1. Nhập control DataGrid vào thời gian thiết kế.

2. Xác lập đặc tính AllowPaging là **True**.

3. Xác lập đặc tính PageSize là số dòng hiển thị trên mỗi trang. Kích thước mặc định là 10.

4. Viết mã vào thủ tục sự kiện PageIndexChanged.

```
Private Sub grdProducts_PageIndexChanged( _
 ByVal source As Object, _
 ByVal e As _

System.Web.UI.WebControls.DataGridPageChangedEventArgs)
  Handles grdProducts.PageIndexChanged
    grdProducts.CurrentPageIndex = e.NewPageIndex
    GridLoad()
End Sub
```

Điều duy nhất bạn phải làm trong thủ tục sự kiện PageIndexChanged là xác lập đặc tính CurrentPageIndex là trang được chuyển vào DataGridPageChangedEventArgs, sau đó gọi thủ tục của bạn liên kết dữ liệu. Cần nhớ tên của thủ tục này là GridLoad. Các DataSet có khả năng thực hiện việc lập số trang, vì vậy sau khi gán DataSet vào DataGrid, DataGrid sẽ xác định là số trang nào, đi tới trang đó trong DataSet và tự động làm tươi bảng HTML bắt đầu với dòng đầu tiên trên trang đó trong DataSet.

- Ấn **F5** để chạy project.

Bây giờ bạn sẽ thấy một số siêu liên kết bên dưới bảng với dấu ngoặc góc phải (>) và dấu ngoặc góc trái (<). Bằng cách nhấp vào dấu ngoặc góc phải bạn có thể di chuyển tới qua các trang. Bằng cách nhấp dấu ngoặc góc trái bạn có thể di chuyển trở lại qua các trang.

Mẹo:

Theo mặc định siêu liên kết Pager trình bày ở dưới đáy. Bằng cách xác lập đặc tính Position, bạn có thể xác lập Pager ở trên Bottom, Top hoặc cả Top lẫn Bottom.

Tùy biến Pager

Như đã nói ở trên, bạn có hai phương thức để tùy biến Pager, bạn có thể dùng mũi tên Next và Previous hoặc bạn có thể dùng các trang số. Bạn sử dụng đặc tính Mode của đối tượng PagerStyle của DataGrid để xác lập đặc tính này. Thí dụ, để sử dụng Numeric Pages, bạn sẽ xác lập đặc tính này như sau:

```
grdProducts.PagerStyle.Mode _
Web.UI.WebControls.PagerMode.NumericPages
```

Mã ở trên có thể được viết trong thủ tục sự kiện Load ngay trước lời gọi thủ tục GridLoad(). Sau khi gọi mã này bạn sẽ nhận các số trang nằm ở dưới đáy của khung lưới thay vì các siêu liên kết Next/Prev.

- Gõ mã ở trên vào thủ tục sự kiện Load.
- Ấn **F5** để chạy ứng dụng và xem các kết quả.

Xác lập văn bản Next và Previous

Thay vì sự thể hiện của Pager mặc định bằng cách dùng các dấu ngoặc góc cho siêu liên kết, bạn có thể tùy biến các dấu ngoặc này một chút. Nếu bạn viết mã sau vào thủ tục sự kiện Load, bạn có thể thay đổi các dấu ngoặc góc này để có các từ

"Next" và "Previous". Bạn sẽ phải sử dụng đối tượng PagerStyle bên trong DataGrid.

Thay đổi mã bạn vừa viết vào thủ tục sự kiện Load như mã được trình bày bên dưới.

```
With grdProducts.PagerStyle
     .Mode = Web.UI.WebControls.PagerMode.NextPrev
     .NextPageText = "Next >"
     .PrevPageText = "< Previous"
End With
```

Đầu tiên mã ở trên sẽ xác lập đặc tính Mode là NextPrev. Điều này cho bạn các siêu liên kết tới và lui mặc định. Đặc tính .NextPageText điều khiển văn bản nào được dùng cho siêu liên kết Next và đặc tính .PrevPageText xác lập văn bản cho siêu liên kết Previous.

Chạy project này để xem các kết quả.

Xác lập các ảnh của Next và Previous

Thay vì chỉ sử dụng các từ trong đặc tính NextPageText và PrevPageText, bạn cũng có thể dùng các tag HTML. Thí dụ, bạn có thể dùng các ảnh đồ họa để cho phép người dùng lập số trang bằng dữ liệu. Thay đổi dữ liệu bạn vừa viết và sử dụng mã sau để xem điều này dễ dàng như thế nào.

```
With grdProducts.PagerStyle
     .Mode = Web.UI.WebControls.PagerMode.NextPrev
     .NextPageText = "<img src=NextArrow.ico>"
     .PrevPageText = "<img src=PrevArrow.ico>"
End With
```

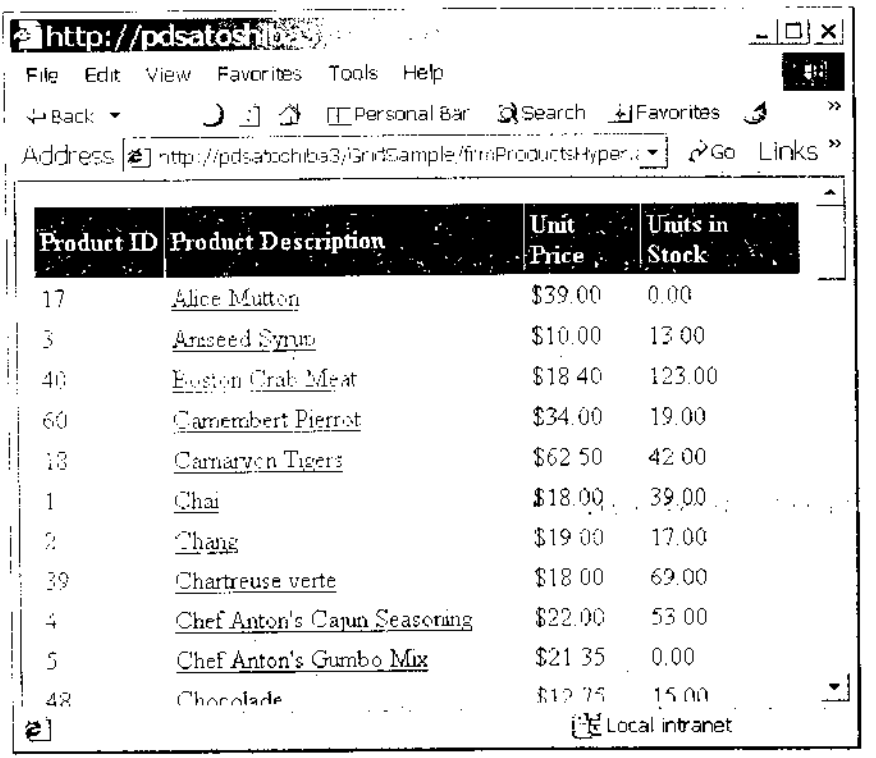
Ở thí dụ này, bạn viết mã HTML thông thường, chẳng hạn như tag <img> để hiển thị biểu tượng mũi tên sang trái và sang phải cho tính năng tới và lui.

Ngoài ra, viết mã và sau đó chạy project để xem các kết quả. Bạn có thể dùng bất kỳ các biểu tượng nào bạn muốn cho các mũi tên. Chúng ta đã dùng NextArrow.ico và PrevArrow.ico với project mẫu đi với chương này.

Một lần nữa, chạy project này để xem các kết quả của sự thay đổi này.

Thêm siêu liên kết

Một bảng chỉ đọc, giống một bảng bạn đã tạo ở chương này, thích hợp cho một danh mục sản phẩm. Tuy nhiên, nếu bạn muốn cho người dùng thêm thông tin hơn là chỉ bên trong bảng này, bạn sẽ phải cho phép người dùng nhấp mục cần hỏi và hiển thị thêm thông tin về sản phẩm đó. Xem hình 4 về thí dụ siêu liên kết trên một trong các cột của bảng.



Product ID	Product Description	Unit Price	Units in Stock
17	Alice Mutton	$39.00	0.00
3	Aniseed Syrup	$10.00	13.00
40	Boston Crab Meat	$18.40	123.00
60	Camembert Pierrot	$34.00	19.00
18	Carnarvon Tigers	$62.50	42.00
1	Chai	$18.00	39.00
2	Chang	$19.00	17.00
39	Chartreuse verte	$18.00	69.00
4	Chef Anton's Cajun Seasoning	$22.00	53.00
5	Chef Anton's Gumbo Mix	$21.35	0.00
48	Chocolade	$12.75	15.00

Hình 4: *Việc thêm siêu liên kết vào một trong các cột thực sự có thể mở rộng tính năng của trang này.*

Để thực hiện điều này, bạn phải thay đổi một trong các tag <asp:BoundColumn> thành tag <asp:HyperLinkColumn>. Khi bạn thay đổi điều này bạn cũng sẽ phải thêm một số thuộc tính để cột ấy biết phải làm gì để tạo siêu liên kết.

Các bước

1. Tìm tag ProductName <asp:BoundColumn> trong cửa sổ HTML đàng sau file WebForm1.aspx.

2. Thay đổi tag đó như sau:

```
<asp:HyperLinkColumn DataNavigateUrlField="ProductID"
 DataNavigateUrlFormatString =
 "frmProductDetail.aspx?ID={0}"
 DataTextField="ProductName"
 HeaderText="Product Description">
</asp:HyperLinkColumn>
```

Có một số thuộc tính mới mà bạn đã thêm vào. Bên dưới là một bảng về mỗi mục này và chức năng mà nó thực hiện.

Mục	Mô tả
DataNavigateUrlField	Trợ giúp trực tuyến cho thuộc tính này không chính xác trong Beta 1. Nó là tên của trường trong bảng bạn muốn chuyển tới một tham số thay thế được chứa trong thuộc tính DataNavigateUrlFormatString. Trong thí dụ của chúng ta, tên của trường là ProductID, vì đây là khóa chính cho bảng. Bạn phải chuyển vào ProductID khác nhau cho mỗi siêu liên kết sẽ được hiển thị trên cột ProductName.
DataNavigateUrlFormatString	Đây là tên của URL bạn sẽ gọi. Bạn có thể tùy chọn đặt vào trình giữ chỗ cho giá trị dữ liệu đến từ tên trường trong thuộc tính DataNavigateUrlField. Bạn tạo trình giữ chỗ này bằng cách đặt vào một "{0}" bên trong trường này, nơi bạn muốn dữ liệu xuất hiện. Dữ liệu cho mỗi dòng ở trường ấy được liệt kê trong DataNavigateUrlField sẽ được thay thế vào trình giữ chỗ này bằng chuỗi. Như bạn thấy ở mã bạn đã viết ở trên, bạn sẽ gọi frmProductDetail.aspx?ID=1 cho sản phẩm, trong đó ProductID bằng 1. Trên dòng, nơi ProductID bằng 40, thì siêu liên kết sẽ là frmProductDetail.aspx?ID=40, vân vân.
HeaderText	Đây là giá trị cho header của bảng này.
DataTextField	Đây là dữ liệu bình thường bạn muốn truy tìm từ dataset liên kết sẽ được dùng như giá trị bên trong siêu liên kết.

Target (không được trình bày ở mã trên)	Thuộc tính này sẽ cho trình duyệt biết nơi hiển thị trang mới này khi siêu liên kết được gọi ra. Các chọn lựa của bạn cho thuộc tính này sẽ xuất hiện trong danh sách cảm ứng thông minh khi bạn soạn thảo mục nhập này. Các chọn lựa là "_blank", "_parent", "_search", "_self", "_top".

Bây giờ bạn đã cho DataGrid biết bạn sẽ có một cột siêu liên kết, bạn có thể chạy nó để xem các kết quả.

1. Chạy project.

2. Đưa con chuột của bạn vào siêu liên kết đầu tiên và để ý ở vùng thanh trạng thái để xem điều gì được liệt kê sau câu lệnh frmProductDetail.aspx?ID=. Nó sẽ là một số tương ứng với sản phẩm đó trong bảng.

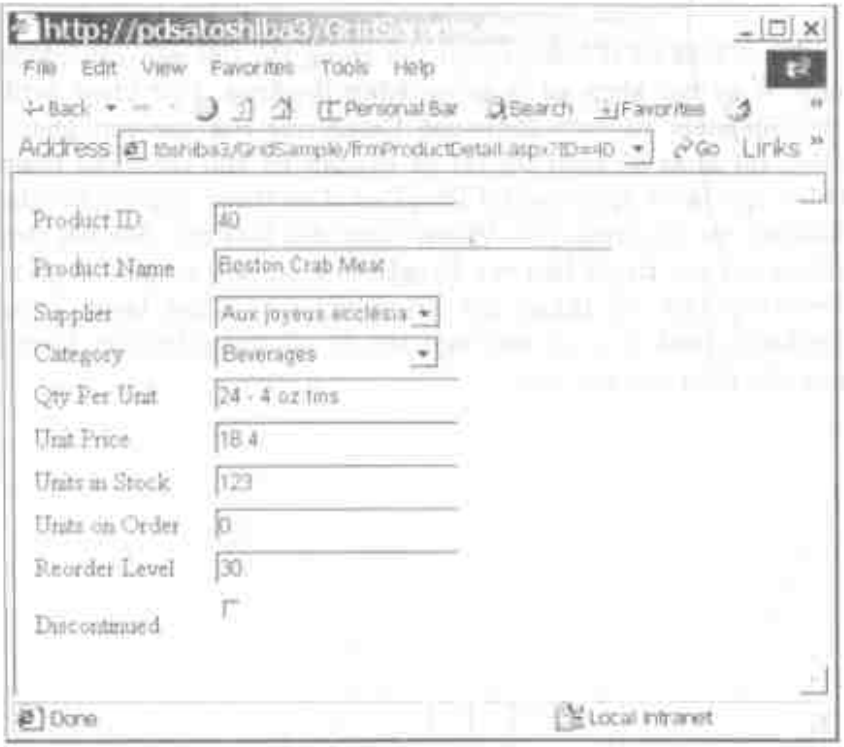
3. Di chuyển chuột xuống danh sách của các siêu liên kết và bạn sẽ thấy một giá trị khác cho mỗi sản phẩm. Số này tương ứng với khóa chính của mỗi sản phẩm.

Đương nhiên, bạn sẽ không thể nhấp vào bất kỳ một trong các siêu liên kết này vì bạn chưa tạo trang frmProductDetail.aspx.

Tạo trang chi tiết

Bây giờ bạn phải tạo trang chi tiết sẽ được điều hướng tới khi siêu liên kết cho một sản phẩm được nhấp. Vì DataGrid chỉ có thể hiển thị nhiều cột mà không tạo cho người dùng cuộn ngang được, bạn sẽ hiển thị tất cả các cột cho một sản phẩm trong frmProductDetail.aspx.

Sẽ có một số thử thách khi tạo trang chi tiết của sản phẩm này, vì có hai khóa xa lạ trong bảng Products. Một khóa dành cho Suppliers và một dành cho Categories của các sản phẩm. Cách tốt nhất để hiển thị tất cả các giá trị cho các bảng tham chiếu này là sử dụng control DropDownList (hoặc hộp combo như hầu hết các lập trình viên Visual Basic đều biết nó). Sau khi nạp tất cả các giá trị dữ liệu vào DropDownList, bạn sẽ phải định vị DropDownList đó thành giá trị tương ứng khóa trong bảng Products. Hình 5 trình bày một thí dụ về trang Product Detail này thể hiện như thế nào.

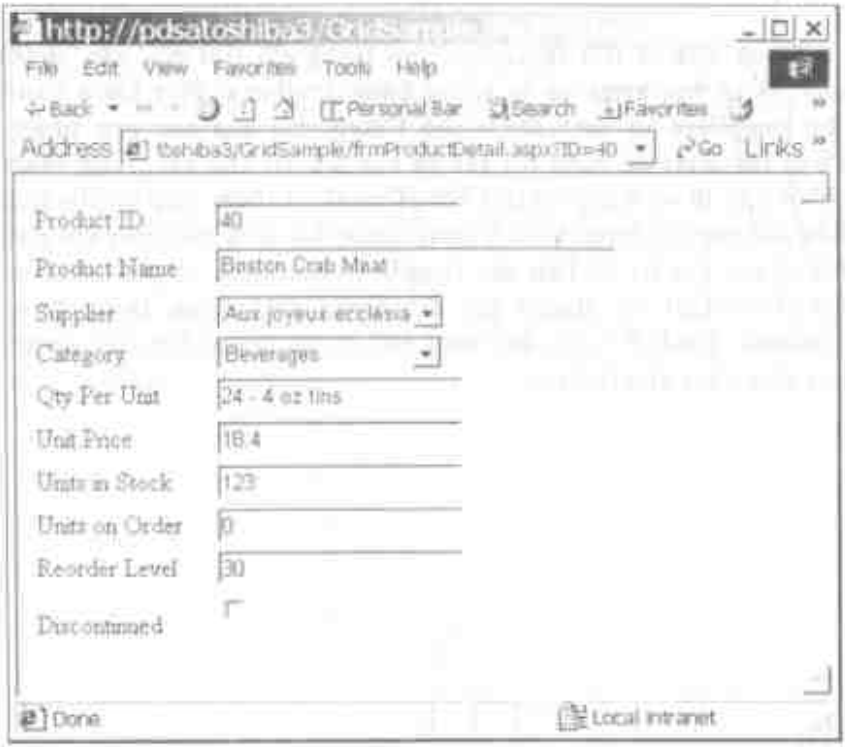


Hình 5: *Trang chi tiết sản phẩm này có thể được xây dựng bằng cách dùng các hộp văn bản, các hộp combo và các hộp kiểm.*

Các bước tạo trang Product Detail

Để có các nhãn và các control nhập liệu trên trang được trình bày ở hình 5, bạn sử dụng control HTML Table với 10 dòng và 2 cột, thuộc tính đường viền được xác lập là zêrô. Vào cột đầu tiên, bạn gõ vào tất cả các nhãn được trình bày ở hình 5, vào cột thứ hai, bạn chèn tất cả các control.

1. Lựa Project | Add WebForm từ menu.



Hình 5: *Trang chi tiết sản phẩm này có thể được xây dựng bằng cách dùng các hộp văn bản, các hộp combo và các hộp kiểm.*

Các bước tạo trang Product Detail

Để có các nhãn và các control nhập liệu trên trang được trình bày ở hình 5, bạn sử dụng control HTML Table với 10 dòng và 2 cột, thuộc tính đường viền được xác lập là zêrô. Vào cột đầu tiên, bạn gõ vào tất cả các nhãn được trình bày ở hình 5, vào cột thứ hai, bạn chèn tất cả các control.

1. Lựa Project | Add WebForm từ menu.

2. Xác lập Name là frmProductDetail.aspx.
3. Nhấp OK để thêm form này vào project của bạn.
4. Lựa Table | Insert | Table từ menu.
5. Điền **10** vào đặc tính Rows.
6. Điền **2** vào đặc tính Columns.
7. Xác lập đặc tính Border Size là **0**.
8. Vào cột đầu tiên của bảng, gõ vào các giá trị được trình bày ở hình 5.

Bây giờ bạn sẽ tạo tất cả các control ở cột thứ hai của bảng này bằng cách nhấp vào mỗi dòng được trình bày ở cột "Row" trong bảng 2. Sau đó bạn sẽ thêm một control từ tab WebForm của Toolbox như được trình bày ở bảng 2. Để thêm control ấy, chỉ cần nhấp đôi nó trong Toolbox sau khi định vị con trỏ của bạn vào dòng được chỉ định.

Dòng	Kiểu control	Đặc tính	Giá trị
1	TextBox	Name	txtProductID
		Text	
2	TextBox	Name	txtProductName
		Text	
3	DropDownList	Name	cboSupplier
4	DropDownList	Name	cboCategory
5	TextBox	Name	txtQtyPerUnit
		Text	
6	TextBox	Name	txtUnitPrice

		Text	
7	TextBox	Name	txtUnitsInStock
		Text	
8	TextBox	Name	txtUnitsOnOrder
		Text	
9	TextBox	Name	txtReorderLevel
		Text	
10	CheckBox	Name	chkDiscontinued

Bảng 2: *Các control để xây dựng trang Product Detail.*

Xây dựng thủ tục sự kiện Load

Mỗi lần bạn gọi trang frmProductDetail.aspx, bạn sẽ phải nạp các hộp combo của các nhà cung cấp và các danh mục, tạo DataSet của dữ liệu sản phẩm mà bạn quan tâm, đưa dữ liệu từ DataSet vào các hộp văn bản thích hợp.

Bạn sẽ gọi ba thủ tục trong sự kiện Load cho trang này: SupplierLoad, CategoryLoad và FormShow. Thủ tục SupplierLoad sẽ đảm trách nạp Suppliers. Thủ tục CategoryLoad sẽ nạp Categories và thủ tục FormShow sẽ nạp sản phẩm và đưa thông tin vào các hộp văn bản.

1. Mở form code-behind cho frmProductDetail.aspx.
2. Định vị thủ tục sự kiện Load.
3. Viết mã sau.

```
Private Sub Page_Load(ByVal Sender As System.Object, _
 ByVal e As System.EventArgs) Handles MyBase.Load
    ' Load Suppliers
    SupplierLoad()
```

```
    ' Load Categories
    CategoryLoad()

    ' Display the Product
    FormShow()
End Sub
```

Nạp Suppliers

Bây giờ bạn sẽ viết thủ tục SupplierLoad. Thủ tục này sẽ nạp trường SupplierID và CompanyName từ bảng Suppliers vào đặc tính DataValueField và DataTextField của control DropDownList tương ứng.

1. Tạo một thủ tục mới sau End Sub của thủ tục sự kiện Load. Gọi nó là SupplierLoad.

2. Gõ mã sau vào thủ tục SupplierLoad này.

```
Private Sub SupplierLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = ConnectStringBuild()

    strSQL = "SELECT SupplierID, CompanyName "
    strSQL &= " FROM Suppliers ORDER BY CompanyName"

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oTable = New DataTable()

    oAdapter.Fill(oTable)

    With cboSupplier
        .DataSource = oTable.DefaultView
        .DataTextField = "CompanyName"
```

```
            .DataValueField = "SupplierID"
            .DataBind()
        End With
    End Sub
```

Sau khi tạo DataSet từ dữ liệu trong bảng, bạn xác lập đặc tính DataSource của control DropDownList là đặc tính DefaultView của đối tượng DataTable. Khung xem này được dùng để nạp dữ liệu vào control DropDownList. Trong phương thức DataBind, các trường được xác lập trong đặc tính DataTextField và DataValueField được dùng để truy tìm dữ liệu từ dòng trong khung xem.

Nạp Categories

Thủ tục được dùng để nạp control Categories DropDownList giống như Suppliers, khác nhau duy nhất là tên của các trường và tên bảng.

1. Tạo một thủ tục mới sau End Sub của thủ tục sự kiện SupplierLoad. Gọi nó là CategoryLoad.

2. Gõ mã sau vào thủ tục CategoryLoad này.

```
Private Sub CategoryLoad()
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oTable As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = ConnectStringBuild()

    strSQL = "SELECT CategoryID, CategoryName "
    strSQL &= " FROM Categories ORDER BY CategoryName"

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
```

```
    oTable = New DataTable()

    oAdapter.Fill(oTable)

    With cboCategory
        .DataSource = oTable.DefaultView
        .DataTextField = "CategoryName"
        .DataValueField = "CategoryID"
        .DataBind()
    End With
End Sub
```

Hiển thị dữ liệu sản phẩm

Bây giờ là lúc nhận dữ liệu sản phẩm sẽ được yêu cầu bằng cách nhấp siêu liên kết trong control DataGrid. Nhớ là ID cho Product được chuyển trên dòng URL, vì vậy bạn có thể dùng phương thức Request.QueryString để truy tìm giá trị đó. Sau đó bạn sẽ dùng giá trị này khi xây dựng chuỗi SQL của bạn để truy tìm sản phẩm cá biệt từ bảng Products.

1. Tạo một thủ tục mới sau End Sub của thủ tục sự kiện CategoryLoad. Gọi nó là FormShow.

2. Gõ mã sau vào thủ tục FormShow này.

```
Private Sub FormShow()
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim oItem As System.Web.UI.WebControls.ListItem
    Dim strSQL As String
    Dim strConn As String

    strConn = ConnectStringBuild()

    strSQL = "SELECT ProductID, ProductName, "
    strSQL &= "SupplierID, CategoryID, "
    strSQL &= "QuantityPerUnit, UnitPrice, "
    strSQL &= "UnitsInStock, UnitsOnOrder, "
```

```
strSQL &= "ReorderLevel, Discontinued "
strSQL &= " FROM Products"
strSQL &= " WHERE ProductID = " & _
 Request.QueryString("ID")

oCmd = New OleDb.OleDbCommand()
With oCmd
    .CommandText = strSQL
    .CommandType = Data.CommandType.Text
    .Connection = New _
        OleDb.OleDbConnection(strConn)
    .Connection.Open()
    oDR = .ExecuteReader( _
        CommandBehavior.SequentialAccess)
End With

If oDR.Read() Then
    txtProductID.Text = oDR("ProductID").ToString()
    txtProductName.Text = _
     oDR("ProductName").ToString()

    ' Position to the correct Supplier
    oItem = _
        cboSupplier.Items.FindByText( _
        oDR("SupplierID").ToString())
    cboSupplier.SelectedIndex = _
        cboSupplier.Items.IndexOf(oItem)

    ' Position to the correct Category
    oItem = _
        cboCategory.Items.FindByText( _
        oDR("CategoryID").ToString())
    cboCategory.SelectedIndex = _
        cboCategory.Items.IndexOf(oItem)
    txtQtyPerUnit.Text = _
     oDR("QuantityPerUnit").ToString()
    txtUnitPrice.Text = oDR("UnitPrice").ToString()
    txtUnitsInStock.Text = _
     oDR("UnitsInStock").ToString()
    txtUnitsOnOrder.Text = _
     oDR("UnitsOnOrder").ToString()
    txtReorderLevel.Text = _
     oDR("ReorderLevel").ToString()
    If CInt(oDR.Item("Discontinued")) = 0 Then
        chkDiscontinued.Checked = False
    Else
        chkDiscontinued.Checked = True
```

```
            End If
        End If
        oDR.Close()
        oCmd.Connection.Close()
    End Sub
```

Trong thủ tục này, bạn sẽ dùng đối tượng OleDbCommand và đối tượng OleDbDataReader để truy tìm dữ liệu. Một trong các thử thách trong thủ tục này là có thể truy tìm SupplierID và CategoryID từ DataReader và định vị các control DropDownList tới chỉ mục thích hợp. Để thực hiện điều này, bạn phải tạo control ListItem mới, đặt giá trị từ DataReader, như Supplier ID vào đặc tính Text của ListItem này, sau đó chuyển đối tượng ListItem này tới phương thức IndexOf của control cboSupplier DropDownList. Phương thức IndexOf cho trở lại vị trí, trong đó giá trị trong đặc tính Text được định vị trong DropDownList. Sau đó bạn có thể chuyển giá trị này tới đặc tính SelectedIndex của DropDownList và điều này sẽ buộc danh sách ấy được định vị ở mục nhập này trong danh sách.

Đương nhiên, bạn đã không phải dùng control DropDownList để hiển thị tất cả các nhà cung cấp và các danh mục. Bạn đã có thể truy tìm ngay các giá trị bằng cách tạo INNER JOIN với bảng Suppliers và Categories, sau đó chỉ cần hiển thị CompanyName và CategoryName vào các hộp văn bản. Tuy nhiên, tại điểm này bạn nên cho phép người dùng soạn thảo màn hình chi tiết sản phẩm này, vì vậy cần có tất cả các giá trị này trong các control DropDownList này.

Tóm tắt

DataGrid là một control phía server rất mạnh. Nhớ là tất cả việc lập số trang, định dạng và liên kết dữ liệu đều xảy ra trên server, chứ không phải trên client. Có nhiều tính năng của DataGrid mà bạn có thể khai thác. Thí dụ, bạn có khả năng sắp

xếp trên các cột, bạn cũng có thể tạo các Pager tùy biến với phong cách riêng biệt khác hơn các tính năng chuẩn được lập sẵn. DataGrid còn có khả năng soạn thảo nữa. Tất cả điều này dẫn tới một control mà bạn chắc chắn có thể khai thác ưu điểm trong các ứng dụng WebForm.

Câu hỏi ôn tập

1. DataGrid có thể dùng ba kiểu đối tượng nào như nguồn Data của nó?

2. Bạn cần các tag nào thêm vào bên trong các tag <asp:datagrid> để liên kết các cột thủ công?

3. Bạn xác lập thuộc tính nào để định dạng các cột số?

4. Bạn phải xác lập đặc tính gì để cho khung lưới biết đi tới trang nào khi sử dụng đối tượng Pager?

5. Bạn sử dụng tag nào để thêm cột siêu liên kết vào DataGrid?

Bài tập

- Tạo một form mới và nạp DataGrid vào trang đó với CustomerID, CompanyName, ContactName và ContactTitle từ bảng Customers trong CSDL Northwind.

- Tạo một cột siêu liên kết trên cột CompanyName. Khi bạn nhấp vào siêu liên kết này để nó gọi một form khác được gọi là CustomerDetail.aspx.

Trả lời câu hỏi ôn tập

1. DataTable, DataView, DataSet.
2. <asp:boundcolumn>.
3. DataFormatString.
4. CurrentPageIndex.
5. <asp:hyperlinkcolumn>.

Chương 30

Hiển thị dữ liệu bằng cách dùng control DataRepeater

- Tìm hiểu cách các control liên kết danh sách hoạt động.
- Thêm các template để điều khiển ứng xử của control Repeater.
- Liên kết dữ liệu với các template bên trong control Repeater.
- Tác động trở lại các sự kiện của các mục trong control Repeater.

Các control liên kết danh sách

ASP.NET cung cấp ba control để hiển thị dữ liệu từ ADO.NET hoặc từ bất kỳ nguồn dữ liệu nào hỗ trợ giao diện IEnumerable: Control Repeater, DataList và DataGrid. Mỗi control này cung

cấp hiển thị dữ liệu dựa vào template, nghĩa là bạn tách riêng dữ liệu liên kết với hiển thị của nó. Mỗi control này biểu hiện nội dung của nguồn dữ liệu theo các template HTML mà bạn cung cấp cho header, các mục, các mục thay thế, footer, vân vân.

Ngoài ra, control DataGrid và DataList còn cho phép bạn lựa, soạn thảo và xóa dữ liệu, cung cấp hỗ trợ cho đặc tính kiểu và hình thức. Tuy nhiên, control Repeater có một mục đích căn bản: nó có thể hiển thị dữ liệu sống động bằng cách dùng các template HTML theo bố trí luồng. Bạn có thể nhúng các control server bên trong Repeater để bạn có thể tác động trở lại các yêu cầu điều hướng từ bên trong dữ liệu của control Repeater. Bảng 1 liệt kê các tính năng quan trọng của ba control liên kết danh sách khác nhau để bạn có thể so sánh các yêu cầu của bạn dựa trên các khả năng sẵn có của các control và quyết định sử dụng khả năng nào.

Tính năng	Repeater	DataList	DataGrid
Templates	Có (đòi hỏi)	Có (đòi hỏi)	Bên trong các cột (tùy chọn)
Tabular Layout	Không	Không	Có
Flow Layout	Có	Có	Không
Columnar/Newspaper-style layout	Không	Có	Không
Style/Appearance Properties	Không	Có	Có
Selection	Không	Có	Có
Editing	Không	Có	Có
Deleting	Không	Có	Có
Paging	Không	Không	Có
Sorting	Không	Không	Có

Bảng 1: *Mỗi control server liên kết danh sách trong ASP.NET hoạt động hơi khác nhau.*

Ở chương này, bạn sẽ thấy cách tạo một control Repeater đơn giản chứa các siêu liên kết và cách tạo Repeater bằng để tác động trở lại các sự kiện được kích hoạt khi bạn nhắp các liên kết bên trong control ấy.

Control Repeater hoạt động như thế nào?

Để cho control Repeater hiển thị dữ liệu, bạn phải cung cấp hai thứ: Giá trị đặc tính DataSource (chỉ định nguồn dữ liệu) và ít nhất một template HTML (chỉ định control sẽ hiển thị dữ liệu từ nguồn dữ liệu của nó như thế nào).

Xác lập đặc tính DataSource

Để Repeater có thể hiển thị dữ liệu (lý do duy nhất để tồn tại), bạn phải xác lập đặc tính DataSource của control. Bạn có thể dùng DataTable (bạn sẽ thấy ở thí dụ) hoặc bất kỳ một đối tượng khác hỗ trợ IEnumerable• interface, including Array, ArrayList, DataView, HashTable, Queue và hơn nữa.

Hiển thị dữ liệu

Control Repeater không cung cấp hỗ trợ hiển thị dữ liệu lập sẵn. Bạn phải cho control biết bạn muốn làm gì với dữ liệu của mình bằng cách cung cấp một hoặc nhiều template cho tập hợp Items của control. Mỗi mục trong tập hợp Items biểu hiện dữ liệu của nó khi bạn yêu cầu và Repeater lặp lại qua toàn bộ nguồn dữ liệu của nó bằng cách dùng các template bạn cung cấp để hiển thị dữ liệu mỗi lần một dòng. Bảng 2 liệt kê các kiểu template kết hợp dữ liệu mà control Repeater hỗ trợ. Bảng 3 liệt kê các kiểu template không được kết hợp với dữ liệu.

Kiểu Item	Mô tả
Item	Được tạo cho tất cả các dòng trong nguồn dữ liệu, trừ khi bạn đã chỉ định một template AlternatingItem — trong trường hợp đó, nó được tạo cho các mục có số chẵn từ 0.
AlternatingItem	Được tạo cho các mục có số lẻ từ 0 bên trong nguồn dữ liệu.

Bảng 2: *Các kiểu Item kết hợp dữ liệu được control Repeater hỗ trợ.*

Kiểu Item	Mô tả
Header	Được tạo cho header của control.
Footer	Được tạo cho footer của control.
Separator	Được tạo để tách các dòng dữ liệu.

Bảng 3: *Các kiểu Item được control Repeater hỗ trợ không được kết hợp với dữ liệu.*

Khi ASP.NET biểu hiện control Repeater, nó thực hiện các bước sau:

1. Đầu tiên control tìm phần tử HeaderTemplate bên trong thân của nó. Nếu nó tìm thấy một phần tử, nó xuất liệu HTML mà bạn đã cung cấp.

2. Control tìm phần tử ItemTemplate và biểu hiện dòng đầu tiên của dữ liệu bằng cách dùng template nó tìm thấy.

3. Nếu control tìm thấy phần tử SeparatorTemplate, nó sử dụng nội dung của phần tử đó để hiển thị dấu tách giữa các dòng.

4. Nếu control tìm thấy phần tử AlternatingItemTemplate, nó sử dụng template đó để hiển thị dòng dữ liệu thứ hai.

5. Control lặp lại ba bước trước cho tất cả các dòng dữ liệu trong nguồn dữ liệu.

6. Nếu control tìm thấy phần tử FooterTemplate, nó sử dụng template ấy để định dạng footer cho control.

7. Control Repeater đòi hỏi ít nhất một ItemTemplate. Nó là template đòi hỏi duy nhất cho control Repeater, nhưng control sẽ không làm việc nếu không có template này.

Liên kết dữ liệu trong các template

Việc sử dụng cú pháp liên kết dữ liệu của ASP.NET (tức là bao quanh các mục dữ liệu với biểu tượng <%# ... %>), bạn có thể chỉ định bạn muốn hiển thị dữ liệu từ nguồn dữ liệu của Repeater bên trong một template. Khi ASP.NET biểu hiện mỗi template, nó cung cấp đối tượng logic, Container để tham chiếu đối tượng RepeaterItem để biểu thị mục template đang được hiển thị. Đối tượng Container cung cấp đặc tính DataItem, tham chiếu dòng dữ liệu đang được control biểu hiện.

Đối tượng **DataBinder** cho phép ASP.NET cung cấp việc liên kết dữ liệu khi nó biểu hiện trang và bạn sẽ dùng phương thức **Eval** của đối tượng để thực hiện tìm kiếm dữ liệu bên trong dòng hiện hành, khi ASP.NET biểu hiện mỗi dòng dữ liệu của Repeater.

Kết hợp tất cả điều này với nhau, bạn sẽ thấy các biểu thức như bên dưới ở khắp HTML của Repeater, yêu cầu ASP.NET tìm dòng và cột chính xác khi nó biểu hiện trang:

```
<%# DataBinder.Eval(Container.DataItem, "CategoryName") %>
```

Bên trong thân trang, biểu thức này cho ASP.NET biết để truy tìm trường CategoryName từ dữ liệu được cung cấp ở dòng hiện hành và xuất liệu giá trị của nó vào dòng truyền HTML mà ASP.NET gửi tới trình duyệt của client.

Móc nối dữ liệu

Khi ASP.NET nạp file ASPX, nó sẽ không tự động liên kết dữ liệu với control Repeater của bạn. Vì có thể bạn phải kiểm soát chính xác việc liên kết dữ liệu xảy ra như thế nào và khi nào, ASP.NET không đảm trách điều này khi bạn muốn truy tìm dữ liệu.

Để điền Repeater, bạn phải thực hiện hai bước:

1. Xác lập đặc tính DataSource của control, như đã được mô tả ở trên.

2. Gọi phương thức DataBind của control. Điều này thực sự khiến cho ASP.NET biểu hiện control và dữ liệu của nó.

Lưu ý:

Bạn có thể quyết định gọi phương thức DataBind chỉ một lần: trong thủ tục sự kiện được gọi khi trang nạp. Bạn cũng nên gọi DataBind khi chính dữ liệu đã thay đổi hoặc bạn sẽ gọi nó mỗi lần bạn gửi trở lại trang ấy.

Mẹo:

Nếu bạn cho phép quản lý trạng thái trang, control Repeater lưu tất cả thông tin cần thiết để tạo lại các mục của nó trong khi gửi trở lại. Bạn sẽ không phải xác lập lại nguồn dữ liệu của nó trong trường hợp này và trang minh họa sẽ khai thác điểm này, chỉ phải liên kết khi biểu hiện trang lần đầu.

Các bước sau hướng dẫn bạn tạo một minh họa đơn giản về việc sử dụng control Repeater. Để minh họa, bạn sẽ tạo một trang đọc thông tin phân loại sản phẩm từ CSDL mẫu của Northwind SQL Server và tạo một danh sách như được trình bày ở hình 1.

Mẹo:

Phiên bản hoàn chỉnh của project này có sẵn trong thư mục gốc ảo PKDataRepeater. Duyệt định vị của nó như được trình bày ở hình 1 để xem các kết quả cuối cùng.

Thực hiện các bước sau để tạo trang mẫu:

1. Tạo một ASP.NET Web Application của Visual Basic.NET.
2. Đặt tên project là DRTest.
3. Lựa WebForm1.aspx trong trình thiết kế và xác lập đặc tính **pageLayout** của nó là **FlowLayout**.

Hình 1: *Tạo danh sách các mục dữ liệu bằng cách dùng control Repeater.*

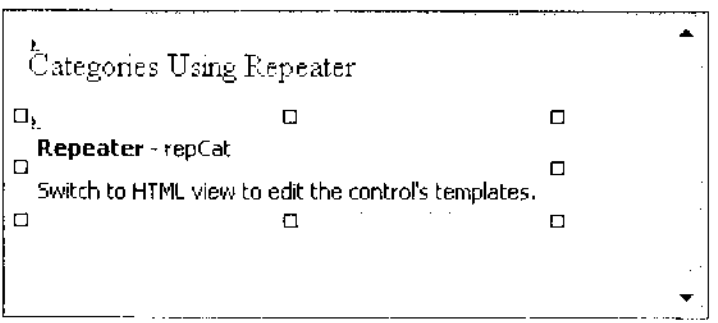
Lưu ý:

Mặc dù bạn có thể sử dụng GridLayout cho đặc tính pageLayout dễ dàng, nhưng không có control nào trong thí dụ này đòi hỏi nó và nó chỉ gây phức tạp thêm. Chúng ta tìm cách sử dụng FlowLayout thích hợp cho hầu hết các mục đích minh họa và nó sẽ làm cho mã phát sinh đơn giản hơn nhiều.

4. Thêm control Label vào trang và xác lập đặc tính **Text** của nó là **Categories Using Repeater**. Xác lập Font và các đặc tính khác cho control này, nếu bạn muốn.

5. Thêm control Repeater vào trang và xác lập **ID** của nó là **repCat**. Như bạn thấy ở hình 2, bạn sẽ phải chuyển sang khung xem HTML để soạn thảo các template.

6. Nhấp tab HTML ở dưới đáy cửa sổ trình thiết kế để chuyển sang khung xem HTML.



Hình 2: *Bạn không thể soạn thảo các template của control Repeater mà không sử dụng khung xem HTML.*

Thêm template Header và Footer

Vì các danh sách có bullet trong HTML bắt đầu với <ul> và dừng với , bạn có thể đặt các tag đó vào phần tử HeaderTemplate và FooterTemplate. Ngoài ra, bạn phải chèn tag và vào để điều khiển font của danh sách.

Tìm phần tử sau bên trong HTML của trang:

```
<asp:Repeater id="repCat" runat="server"></asp:Repeater>
```

Giữa phần tử bắt đầu và dừng, chèn HTML sau để định nghĩa template header và footer:

```
<HeaderTemplate>
  <font face="Verdana" size="2">
```

```
    <ul>
</HeaderTemplate>
<FooterTemplate>
  </ul> </font>
</FooterTemplate>
```

Mẹo:

Khi bạn gõ, bạn sẽ thấy trình soạn thảo của Visual Studio trợ giúp: nó cung cấp một danh sách các template sẵn có và bạn chỉ cần chọn từ danh sách này.

Thêm mã truy tìm dữ liệu

Thực hiện các bước sau để cung cấp mã đòi hỏi cho việc truy tìm dữ liệu từ SQL Server:

1. Ấn **F7** (hoặc sử dụng mục menu **View|Code**) để hiển thị file code-behind cho trang.

2. Ở trên đầu file mã, thêm câu lệnh Imports, để bạn không phải tham chiếu các thành viên của lớp OleDb rõ ràng:

```
Imports System.Data.OleDb
```

3. Thêm thủ tục sau bên trong lớp của trang để truy tìm dữ liệu, xác lập đặc tính DataSource của control Repeater và liên kết dữ liệu:

```
Private Sub RepeaterBind()
    Dim da As OleDbDataAdapter
    Dim dt As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = ConnectStringBuild()

    strSQL = "SELECT 'CategoryDetail.aspx?ID=' + "
```

```
        strSQL &= " Cast(CategoryID As Char(2)) "
        strSQL &= "As CategoryURL, CategoryName "
        strSQL &= " FROM Categories ORDER BY CategoryName"

        da = New OleDbDataAdapter(strSQL, strConn)
        dt = New DataTable()

        da.Fill(dt)

        repCat.DataSource = dt
        repCat.DataBind()
    End Sub
```

Lưu ý:

Câu lệnh SQL bạn đã xây dựng trong phân đoạn mã trước đây truy tìm một URL, liên kết với trang có tên CategoryDetail.aspx, chuyển giá trị CategoryID như tham số của nó. Ngoài ra, SQL truy tìm trường CategoryName. Bạn sẽ thêm trang CategoryDetail.aspx vào project ở phần sau.

4. Sửa đổi thủ tục **Page_Load** để nó gọi thủ tục RepeaterBind bạn vừa thêm vào:

```
Private Sub Page_Load( _
ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    If Not Page.IsPostBack Then
        RepeaterBind()
    End If
End Sub
```

5. Sử dụng mục menu **Project|Add New Item** để hiển thị hộp thoại Add New Item và lựa Module từ danh sách các chọn lựa. Đổi tên từ module1.vb thành tên bạn chọn, nếu cần.

6. Vào module mới, thêm mã sau để cung cấp chuỗi kết nối được đòi hỏi ở bước trước đây:

```
Public Function ConnectStringBuild() As String
    Dim strConn As String

    strConn = "Provider=sqloledb"
    strConn &= ";Data Source=(local)"
    strConn &= ";Initial Catalog=Northwind"
    strConn &= ";User ID=sa"
    strConn &= ";Password="

    Return strConn
End Function
```

Thêm ItemTemplate

Mục tiêu kế tiếp là thêm ItemTemplate vào control Repeater. Template này khiến cho control hiển thị dữ liệu mỗi lần một dòng từ nguồn dữ liệu của control ấy.

Thực hiện các bước sau để thêm template Item và kiểm tra trang:

1. Trong Solution Explorer, lựa trang bạn đã tạo cho project này. Nhắp đôi để nạp trang.

2. Ấn **SHIFT+F7** (hoặc chọn mục menu **View|Designer**) để xem trình thiết kế của trang.

3. Tìm phần tử HeaderTemplate và FooterTemplate bạn đã chèn trước đây. Giữa hai template, chèn phần tử sau (lưu ý hai biểu thức liên kết được đánh dấu bằng nét đậm):

```
<ItemTemplate>
  <li>
    <asp:HyperLink Runat="server"
```

```
        Text='<%# Databinder.Eval(
        Container.DataItem, "CategoryName") %>'
        NavigateUrl='<%# Databinder.Eval(
        Container.DataItem, "CategoryURL") %>'
        ID="Hyperlink1" NAME="Hyperlink1" />
    </li>
</ItemTemplate>
```

Mẹo:

Phần đoạn mã trước, bạn đã thêm control Hyperlink Web để hiển thị dữ liệu từ mỗi dòng trong nguồn dữ liệu của Repeater. Văn bản sẽ là trường CategoryName và URL mà siêu liên kết nhảy tới sẽ là CategoryURL mà bạn đã tạo trong mã khi chèn vào trước đây.

4. Trong cửa sổ Solution Explorer, nhắp nút chuột phải vào WebForm1.aspx và lựa **Set as Start Page** từ menu context.

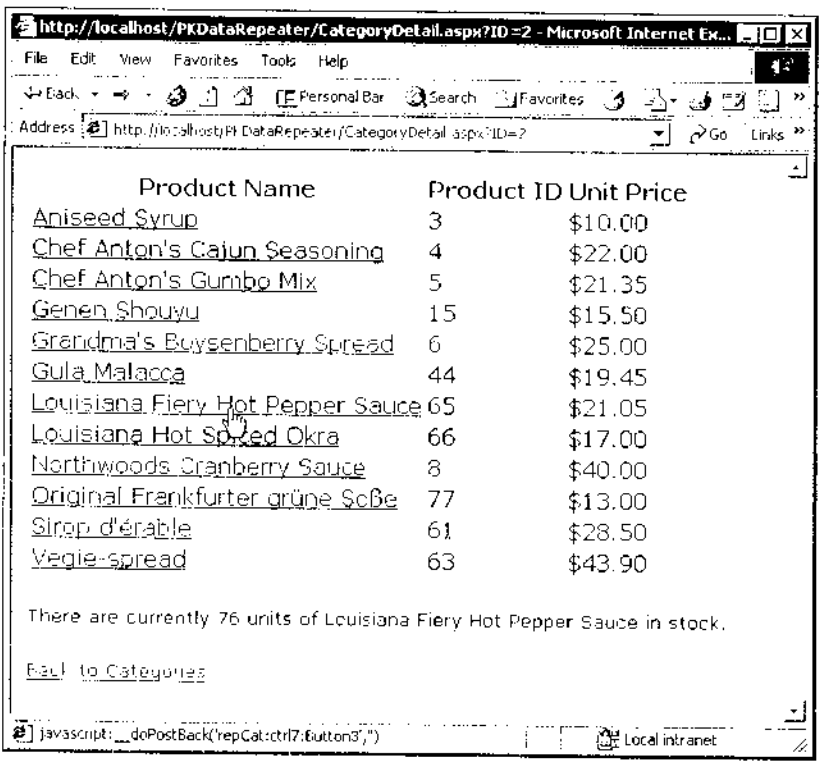
5. Ấn **F5** để chạy project và xác định bạn thấy danh sách các tên phân loại trong cửa sổ trình duyệt. Bạn có thể lựa các liên kết nhưng ở điểm này chúng không có tác dụng. (Bạn chưa cung cấp trang CategoryDetail.aspx, vì vậy các liên kết không thể hoạt động).

6. Đóng trình duyệt và lưu project.

Cho tới đây bạn đã tạo một control Repeater đơn giản và đã cho nó biểu hiện các template Header, Footer và Item. Template Item chứa thông tin được lấy ra từ CSDL SQL Server, bao gồm trường CategoryName và CategoryID.

Các tính năng Repeater cao cấp hơn

Ở phần trước bạn đã tạo một trang đơn giản có các liên kết với trang hiển thị các chi tiết phân loại. Ở phần này, bạn sẽ tìm hiểu cách cung cấp template AlternatingItem, cách thêm các control đã nhúng và cách tác động trở lại các sự kiện của các control này. Theo cách này, bạn sẽ tạo trang chi tiết phân loại được trình bày ở hình 3.



Hình 3: *Việc nhấp vào liên kết trên trang này truy tìm các đơn vị hàng kho cho mục đã lựa.*

Tạo trang chi tiết

Để bắt đầu khảo sát các tính năng cao cấp hơn, bạn sẽ phải tạo trang chi tiết. Thực hiện các bước sau để tạo trang cơ sở bạn sẽ cần:

1. Lựa mục menu **Project|Add Web Form** và đặt tên trang mới là **CategoryDetail.aspx**.

2. Xác lập đặc tính **pageLayout** của trang là **FlowLayout**.

3. Chèn control Repeater vào trang và xác lập đặc tính **ID** của nó là **repCat**.

4. Ấn **ENTER** để chèn một đoạn văn bản mới và sau đó chèn control Label vào trang. Xác lập **ID** của control nhãn là **lblInventory**. Xóa giá trị trong đặc tính Text.

5. Ấn **ENTER** để chèn một đoạn văn bản mới và sau đó chèn control Hyperlink vào trang. Xác lập đặc tính **Text** của control là **Back to Categories** và đặc tính **NavigateURL** là trang bạn đã tạo ở phần trước đây.

6. Ấn **F7** để nạp file code-behind của trang.

7. Ở trên đầu của file mã, thêm câu lệnh Imports này để bạn không phải tham chiếu các thành viên của lớp OleDb rõ ràng:

```
Imports System.Data.OleDb
```

8. Thêm thủ tục sau vào lớp của trang:

```
Private Sub RepeaterBind()
    Dim da As OleDbDataAdapter
```

```
    Dim dt As DataTable
    Dim strSQL As String
    Dim strConn As String

    strConn = ConnectStringBuild()

    ' Get the ID value passed in.
    strSQL = _
     String.Format("SELECT ProductID, ProductName, " &
_
     "UnitPrice, UnitsInStock " & _
     "FROM Products WHERE CategoryID = {0} " & _
     "ORDER BY ProductName", Request.Item("ID"))

    da = New SqlDataAdapter(strSQL, strConn)
    dt = New DataTable()

    da.Fill(dt)

    repCat.DataSource = dt
    repCat.DataBind()
End Sub
```

Trong trường hợp này, thủ tục RepeaterBind truy tìm chuỗi kết nối, sau đó xây dựng chuỗi SQL để chỉ định các dòng truy tìm từ SQL Server. Vì trang này nhận giá trị ID trong chuỗi truy vấn được chuyển từ trang trước, mã sử dụng Request.Item để truy tìm giá trị ID và chèn nó vào chuỗi SQL. Mã thiết lập DataAdapter, điền vào DataTable và xác lập đặc tính DataSource của control Repeater là bảng này. Cuối cùng, mã gọi phương thức DataBind để điền Repeater.

9. Sửa đổi thủ tục **Page_Load** để nó gọi thủ tục RepeaterBind bạn vừa thêm:

```
Private Sub Page_Load( _
ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    If Not Page.IsPostBack Then
        RepeaterBind()
```

```
    End If
End Sub
```

10. Ấn **SHIFT+F7** để nạp trình thiết kế trang và nhấp tag HTML ở dưới đáy của cửa sổ để chuyển sang khung xem HTML.
11. Tìm tag </asp:Repeater> để kết thúc định nghĩa control Repeater và chèn template HeaderItem và FooterItem bên dưới vào trước tag dừng của control Repeater:

```
<HeaderTemplate>
  <table border="0">
    <tr>
      <th>
        Product Name</th>
      <th>
        Product ID</th>
      <th>
        Unit Price</th>
    </tr>
/HeaderTemplate>
<FooterTemplate>
  </table>
</FooterTemplate>
```

Lưu ý:

Template header và footer bạn vừa thêm tạo tag bắt đầu và dừng cho một bảng. Chỉ còn phải thêm nội dung cho mỗi dòng.

12. Giữa phần tử HeaderItem và FooterItem bạn vừa thêm, chèn phần tử ItemTemplate sau:

```
<ItemTemplate>
<tr style bgcolor="PapayaWhip">
  <td>
    <asp:LinkButton
      CommandName='<%# Databinder.Eval(
       Container.DataItem, "ProductName") %>'
      Text='<%# Databinder.Eval(
```

```
        Container.DataItem, "ProductName") %>'
      Runat="server" ID="Button3"
      CommandArgument='<%#Databinder.Eval(
        Container.DataItem, "UnitsInStock")%>' />
  </td>
  <td>
      <%# Databinder.Eval(
       Container.DataItem, "ProductID") %>
  </td>
  <td>
     <%# FormatCurrency(Databinder.Eval(
      Container.DataItem, "UnitPrice")) %>
  </td>
</tr>
</ItemTemplate>
```

13. Ấn **F5** để chạy project và xác định là bạn có thể lựa các phân loại trên trang đầu tiên và thấy các sản phẩm trong phân loại đó trên trang thứ hai.

14. Đóng trình duyệt và lưu project của bạn.

15. Điều gì xảy ra với thuộc tính **CommandName** và **CommandArgument** của phần tử LinkButton bên trong phần tử ItemTemplate? Rõ ràng là thuộc tính Text chỉ định văn bản sẽ được hiển thị trên liên kết. Thuộc tính CommandName và CommandArgument cho phép bạn chỉ định thông tin cần gửi tới thủ tục sự kiện khi bạn nhấp liên kết. Ở phần kế tiếp, bạn sẽ tìm hiểu cách khai thác các tham số này khi bạn hiển thị bảng kiểm kê (inventory) sẵn có cho mục đã lựa.

Thêm template AlternateItem và tác động trở lại sự kiện Command

Cho tới đây, bạn đã tạo một bảng chứa các mục bên trong một phân loại, nhưng có thể bạn muốn có các mục thay thế được hiển

thị bằng các màu khác nhau và nó cũng sẽ rất tốt nếu bạn có thể nhấp một mục và khả năng hữu hiệu hiện hành của nó (các đơn vị hàng kho) được hiển thị. Ở phần kết thúc này, bạn sẽ thêm hai tính năng cuối cùng này vào trang mẫu.

Thêm template AlternatingItem

Thực hiện các bước sau để thêm template mục thay thế:

1. Với trình thiết kế CategoryDetail.aspx mở trong khung xem HTML, lựa toàn bộ phần tử **ItemDetail**, gồm cả các tag của nó. Ấn **CTRL+C** để chép phần tử ấy vào Clipboard.

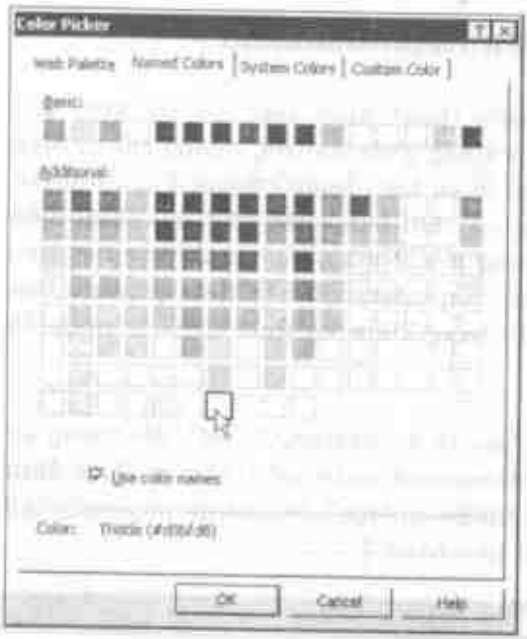
2. Di chuyển điểm chèn ngay sau tag đóng </ItemTemplate> và ấn **CTRL+V** để dán phần tử đã chép trở lại tài liệu.

3. Sửa đổi tag bắt đầu và dừng của phần tử mới, đổi tên từ **ItemTemplate** thành **AlternatingItemTemplate**.

4. Đổi thuộc tính **bgColor** của thuộc tính **style** của dòng thành một màu khác. Bạn có thể thử **Lavender** hoặc chọn bất kỳ màu khác.

Mẹo:

Để giúp lựa các màu, bạn có thể dùng các công cụ được Visual Studio cung cấp. Xóa thuộc tính bgColor và giá trị của nó, gõ lại "bgColor =". Visual Studio sẽ cung cấp lời mách "Pick Color" và sau đó bạn có thể lựa các màu trong hộp thoại được trình bày ở hình 4.



Hình 4: Chọn các màu bằng cách dùng hộp thoại trợ giúp này.

Lưu ý:

Mặc dù bạn chỉ thay đổi màu nền cho các mục thay thế, nhưng bạn cũng có thể thay đổi các thuộc tính khác.

5. Chạy project như trước đây và xác định là bây giờ bạn thấy các màu thay thế trên trang CategoryDetail.aspx.

6. Khi thực hiện xong, đóng trình duyệt và lưu project của bạn.

Thêm thủ tục sự kiện ItemCommand

Mặc dù control Repeater phát sinh một số sự kiện khi ASP.NET biểu hiện và hủy trang chứa control, nhưng chỉ sự kiện bạn có thể tác động trở lại là sự kiện ItemCommand của control. Control phát sinh sự kiện này khi bạn nhấp một nút (hoặc liên kết) bên trong control. Trong minh họa này, bạn đã tạo một siêu liên kết bên trong control Repeater. Để kết thúc project, bạn phải thêm mã để hiển thị bảng kiểm kê thực chất cho mục đã lựa.

Ngoài tham số chuẩn "source As System.Object", đối tượng ủy nhiệm xử lý sự kiện ItemCommand nhận một tham số được định nghĩa "As RepeaterCommandEventArgs". Đối tượng này cung cấp bốn đặc tính như được mô tả ở bảng 4.

Đặc tính	Mô tả
CommandArgument	Truy tìm giá trị được đặt vào thuộc tính CommandArgument trong control Repeater.
CommandName	Truy tìm giá trị được đặt vào thuộc tính CommandName trong control Repeater.
CommandSource	Truy tìm nguồn lệnh.
Item	Truy tìm tham chiếu với RepeaterItem kết hợp với sự kiện.

Bảng 4: *Đối tượng RepeaterCommandEventArgs cung cấp các đặc tính hữu dụng này.*

Mặc dù thông thường bạn có thể xác lập thuộc tính CommandName trong control Repeater để biểu thị tên của một lệnh (như "Sort") và CommandArgument chứa một lệnh phụ (như "Descending"); trong thí dụ này, thuộc tính CommandName chứa trường ProductName và thuộc tính CommandArgument chứa

trường UnitsInStock. Vì bạn đã điền các thuộc tính này vào (xem listing trên) trong control Repeater, bạn có thể truy tìm chúng từ bên trong file code-behind trong handler sự kiện ItemCommand.

Thực hiện các bước sau để thêm sự kiện xử lý trang mẫu:

1. Chắc chắn CategoryDetail.aspx mở trong cửa sổ trình thiết kế.

2. Ấn **F7** để xem file code-behind cho trang.

3. Trong hộp combo Class Name ở đỉnh bên trái của cửa sổ, lựa **repCat**.

4. Trong hộp combo Method Name ở đỉnh bên phải của cửa sổ, lựa **repCat_ItemCommand**. (Tại điểm này, trình soạn thảo chèn cuống của handler sự kiện).

5. Sửa đổi thủ tục repCat_ItemCommand như sau:

```
Private Sub repCat_ItemCommand( _
 ByVal source As System.Object, _
 ByVal e As _
 System.Web.UI.WebControls.RepeaterCommandEventArgs) _
 Handles repCat.ItemCommand
    lblInventory.Text = _
        String.Format("There are currently {0} units "
& _
        "of {1} in stock.", _
        e.CommandArgument, e.CommandName)
End Sub
```

6. Chạy project một lần nữa, lựa một phân loại và sau đó lựa một mục bằng cách nhấp vào liên kết của nó. Bạn sẽ thấy bảng kiểm kê sẵn có trong control nhãn trên trang chi tiết phân loại.

7. Đóng trình duyệt và lưu project của bạn.

Tóm tắt

- Control Repeater là một trong ba control liên kết danh sách được ASP.NET cung cấp: Repeater, DataList và DataGrid.

- Mỗi control trong ba control liên kết danh sách này cung cấp các tính năng khác nhau với các cách dùng khác nhau.

- Bạn phải xác lập đặc tính DataSource của Repeater để liên kết nó với một đối tượng bất kỳ hỗ trợ giao diện IEnumerable. Trong thí dụ này, bạn đã liên kết control với đối tượng DataTable.

- Để cho control Repeater hiển thị dữ liệu của nó, bạn phải cung cấp ít nhất một template HTML: ItemTemplate. Bạn cũng có thể cung cấp phần tử HeaderTemplate, FooterTemplate, SeparatorTemplate và AlternatingTemplate.

- Bạn phải gọi phương thức DataBind của control Repeater để nạp dữ liệu vào control.

- Bạn có thể tác động trở lại sự kiện ItemCommand trong file code-behind của bạn được phát sinh khi người dùng nhắp vào một nút hoặc một liên kết bên trong control Repeater.

- Bạn có thể chuyển thông tin từ Repeater tới handler sự kiện bằng cách dùng thuộc tính CommandArgument và CommandName.

Câu hỏi ôn tập

1. Bạn có thể hiệu chỉnh dữ liệu trong control Repeater không?

2. Bạn phải cung cấp template nào để hiển thị dữ liệu trong control Repeater?

3. Bằng cách nào bạn có thể cung cấp lược đồ màu thay thế trong control Repeater?

4. Bạn phải xác định đặc tính nào và bạn phải gọi phương thức nào trong mã của mình để liên kết dữ liệu từ một nguồn dữ liệu nào đó với control Repeater?

5. Bằng cách nào bạn có thể tác động trở lại sự kiện người dùng nhấp một liên kết hoặc một nút bên trong control Repeater?

Trả lời câu hỏi ôn tập

1. Không, nhưng bạn có thể hiệu chỉnh dữ liệu trong control DataList và DataGrid.

2. Bạn phải cung cấp ít nhất phần tử ItemTemplate.

3. Ngoài phần tử ItemTemplate chuẩn, bạn phải cung cấp phần tử AlternatingItemTemplate.

4. Bạn phải xác lập đặc tính DataSource của control và phải gọi phương thức DataBind.

5. Bạn phải thêm một thủ tục để xử lý sự kiện ItemComment của control Repeater.

Chương 31

Quản lý trạng thái trong ASP.NET

- Tìm hiểu trạng thái là gì và cách sử dụng nó.
- Tìm hiểu các phương thức quản lý trạng thái khác nhau.
- Tìm hiểu các ưu điểm và khuyết điểm của mỗi phương thức.

Quản lý trạng thái

Khi bạn bắt đầu phát triển các ứng dụng Web trong ASP.NET, bạn phải quản lý trạng thái (state). Trạng thái được coi là khả năng lưu giữ dữ liệu bạn sẽ cần cho một người dùng cá biệt trong phiên làm việc của họ. Các trang Web không có trạng thái (stateless), nghĩa là khi bạn di chuyển từ trang này tới trang khác, thì dữ liệu của mỗi trang tự động được loại bỏ. Vì có thể bạn cần dữ liệu của một trong các trang trước đây, bạn phải lưu trữ dữ liệu (hoặc trạng thái) khi di chuyển từ trang này tới trang

khác. Có nhiều phương thức bạn có thể dùng để quản lý các trạng thái này.

Ở chương này, bạn sẽ tìm hiểu cách dùng các phương thức khác nhau để quản lý trạng thái trong ứng dụng Web của .NET. Bạn sẽ tìm hiểu cách dùng đối tượng phiên làm việc, StateBags và .NET Framework để giúp bạn quản lý trạng thái trên một Web farm. Bạn cũng sẽ tìm hiểu các ưu điểm và các khuyết điểm của mỗi kĩ thuật này.

Các phương thức quản lý trạng thái

Có nhiều cách quản lý trạng thái trên Web site của bạn, bao gồm:

- Đối tượng Session và Application.
- Cookie bộ nhớ và đĩa.
- Các trường nhập liệu ẩn và dòng URL.
- StateBag.
- SQL Server.

Đối tượng Session và Application

Bạn sử dụng đối tượng Session để lưu trữ các giá trị giữa các lời gọi các trang Web khác nhau của mỗi người dùng. Bạn sử dụng đối tượng Application để lưu trữ dữ liệu cần thiết trên toàn bộ site và tất cả các người dùng. Một thí dụ điển hình về đối tượng này là chuỗi kết nối cần thiết để thu thập dữ liệu từ một

CSDL. Cả đối tượng Session lẫn Application đều lưu trữ trạng thái trên Web server.

Cookie

Các cookie bộ nhớ là một phương thức mà một số lập trình viên sử dụng để giảm khối lượng tài nguyên được lưu trữ trên server. Cookie bộ nhớ là dữ liệu được lưu trữ trong trình duyệt của người dùng. Dữ liệu này được chuyển tới lui từ trình duyệt tới server khi người dùng di chuyển từ trang này tới trang kia trên site. Khi trình duyệt được đóng, cookie được giải phóng khỏi bộ nhớ.

Nếu bạn biết người dùng sẽ ghé đến site của bạn ngày này tới ngày khác, có thể bạn muốn lưu trữ một cookie thường xuyên trên đĩa cứng của người dùng, nếu họ cho bạn thực hiện. Một cookie thường xuyên được gửi tới từ một Web site tới trình duyệt của người dùng và trình duyệt lưu trữ dữ liệu này trên đĩa cứng của người dùng. Sau đó dữ liệu này được truy tìm từ đĩa bằng trình duyệt khi người dùng ghé lại Web site ấy. Cookie lại được chuyển từ trình duyệt tới server trên mỗi trang mà người dùng điều hướng trên site.

Các trường nhập liệu ẩn

Các trường nhập liệu ẩn có thể được dùng để chuyển dữ liệu từ trang này tới trang khác. Khi người dùng nhấp nút Submit, form cũng gửi dữ liệu người dùng mà họ đã điền vào và bất kỳ các trường nhập liệu ẩn nào. Trường nhập liệu ẩn được tạo bằng cách dùng tag HTML bình thường.

```
<input type="hidden" value="10" name="txtRate">
```

Bạn có thể lưu trữ dữ liệu trong các trường nhập liệu ẩn để giúp duy trì trạng thái từ trang này với trang khác. Ở thí dụ trên, giá trị 10 được lưu trữ trong trường ẩn có tên *txtRate*.

Dòng URL

Bạn có thể chuyển các giá trị trên dòng URL bằng cách dùng các cặp dữ liệu có khóa. Thí dụ, bạn có thể gọi một trang ASP.NET như sau: Main.aspx?CMD=1&ID=29398. Khi bạn gọi trang Main.aspx, bạn chuyển hai biến trên dòng URL: CMD và ID. Bạn có thể chuyển khá ít thông tin trên dòng URL này, vì vậy đây là một phương thức duy trì trạng thái khá hiệu quả.

StateBag

StateBag là một thực thể mới trong .NET để cho phép bạn duy trì trạng thái khi bạn đang làm việc bên trong một trang. Nếu người dùng gửi dữ liệu trở lại server khi vẫn còn trên cùng trang, StateBag có thể được dùng để lưu giữ nhiều giá trị trung gian cho trang này. Một thí dụ điển hình về phương thức này là khi người dùng phải chọn một số giá trị từ một hộp combo, sau đó dựa vào các giá trị được chọn, một hộp combo khác được điền vào với dữ liệu bằng cách dùng dữ liệu từ hộp combo đầu tiên.

SQL Server

Nếu bạn cần duy trì nhiều dữ liệu giữa các trang, như trong trường hợp thu thập một khối lượng lớn dữ liệu về người dùng trên một số trang Web, bạn có thể xem xét việc lưu trữ dữ liệu này vào một CSDL như SQL Server. Có hai phương thức để sử dụng SQL Server: một để xây dựng dữ liệu phiên làm việc và chèn dữ liệu vào chính server của bạn và một để cho .NET Framework xử lý điều đó tự động cho bạn.

Web Forms quản lý ViewState

ViewState được coi là khả năng của các control trên trang Web để lưu giữ dữ liệu của chúng giữa các lần đi dạo từ server trở lại cùng trang. Thí dụ, nếu bạn gõ một ID sản phẩm như "1" vào hộp văn bản và nhấp nút Submit để truy tìm tên sản phẩm hiển thị trên một hộp văn bản khác trên cùng trang, khi trang được vẽ lại với tên sản phẩm ấy, bạn muốn giá trị "1" vẫn còn được dùng bên trong hộp văn bản ID sản phẩm. Vì vậy dù bạn đã dạo tới server và gửi trở lại một trang hoàn toàn mới tới người dùng, nhưng nó trông giống như trang bạn vừa điền vào cùng với một số dữ liệu mới. Web Forms đảm trách việc quản lý ViewState này cho bạn mà về phần bạn không phải viết thêm mã.

Cách Web Forms theo dõi dữ liệu này là bằng cách dùng control nhập liệu ẩn trên trang để theo dõi trang nào và các control nào có các giá trị nào ở trong chúng vào thời gian thích hợp. Thực ra, bạn có thể thấy một biến nhập liệu ẩn này nếu bạn xem nguồn cho một trang Web từ trình duyệt của bạn. Đương nhiên, văn bản trong trường sẽ trông không giống bất kỳ thứ gì khi nó được duy trì ở định dạng mã hóa. Định dạng mã hóa này được phát sinh bởi .NET Framework, vì vậy bạn không cần phải tự mã hóa.

Ngay khi bạn bắt đầu phát triển các ứng dụng Web trong .NET, bạn sẽ thấy có một nhu cầu nhất định phải theo dõi dữ liệu từ một trang này tới trang khác, chứ không phải chỉ trên một trang. Đó là lúc bạn cần dùng đối tượng Session.

Sử dụng đối tượng Session

Mỗi lần một người dùng mới đến site của bạn, một đối tượng Session mới được tạo trên IIS. IIS tự động gán một số duy nhất

vào phiên làm việc này và đặt nó vào đặc tính SessionID của đối tượng Session. Bạn có thể dùng SessionID này để nhận dạng duy nhất một người dùng cá biệt và tạo các biến phiên của riêng bạn để lưu giữ trạng thái với điều kiện phiên làm việc của người dùng hoạt động. SessionID này được gửi tới trình duyệt như một cookie bộ nhớ; mỗi lần trình duyệt điều hướng tới một trang bất kỳ trên site, thì cookie này được gửi tới server bằng các header của HTTP.

Điều này giả định người dùng sẽ chấp nhận các cookie bộ nhớ. Mặc dù hầu hết người dùng chấp nhận các cookie, nhưng vẫn có một số người dùng sẽ không chấp nhận chúng. Ở phần sau trong tài liệu này, bạn sẽ tìm hiểu cách có thể loại bỏ các cookie khi sử dụng các biến Session, nhưng bây giờ chúng ta tiếp tục xét việc dùng các biến Session.

SessionID Longevity

Đối tượng Session cho một người dùng cá biệt không tồn tại vô hạn định. ID chỉ tồn tại cho tới khi một trong các điều kiện sau trở thành sự thực:

- Phương thức Session.Abandon được gọi trong mã của bạn.
- Đối tượng Session hết thời hạn (timeout). Giá trị timeout mặc định là 20 phút. Điều này có nghĩa là nếu người dùng không đệ trình một yêu cầu trở lại site trong thời gian đó, thì đối tượng Session của họ sẽ được giải phóng. Bạn có thể thay đổi giá trị timeout này.
- IIS Service đóng.

Sử dụng biến Session riêng

Đối tượng Session có một tập các đặc tính lập sẵn riêng, nhưng bạn được phép tạo các biến phiên riêng và gán các giá trị vào các biến này. Để tạo một biến đối tượng Session mới, sử dụng cú pháp sau:

```
Session("Email") = "JohnDoe@yahoo.com"
Session("Password") = "password"
```

Ở mã trên tạo một biến mới được gọi là Email, là duy nhất của người dùng này. Nó cũng tạo một biến mới được gọi là Password cũng là duy nhất của người dùng này. Các chuỗi ở bên phải của các dấu bằng là các giá trị bạn gán vào mỗi biến này. Một khi bạn đã tạo các biến này, các giá trị sẽ vẫn còn cho tới khi bạn dứt khoát xác lập chúng bằng Nothing hoặc cho tới khi phiên được hủy như được giải thích ở phần trước.

Các vấn đề với đối tượng Session

Có một số vấn đề bạn cần lưu ý khi dùng đối tượng Session để lưu trữ trạng thái.

Bộ nhớ trên Server

Mỗi biến bạn tạo sử dụng bộ nhớ trên server. Mặc dù bộ nhớ cần cho một biến dường như không nhiều, nhưng khi bạn nhân tất cả dữ liệu trong mỗi biến với số người dùng có thể sử dụng site của cùng một lần, nó có thể là một khối lượng bộ nhớ khá lớn.

Các vấn đề bảo mật

Khi một phiên bắt đầu, người dùng làm việc với cùng SessionID cho tới khi phiên của họ hết giờ. Nếu một tin tặc có thể phát hiện số này, chúng có thể chiếm dụng phiên của người dùng. Nếu bạn lưu trữ thông tin thẻ tín dụng vào một biến phiên, đây là một kẽ hở bảo mật. Mặc dù điều này hầu như không có, nhưng không phải là không có thể xảy ra. Để giải quyết vấn đề này, bạn nên dùng Secure Sockets Layer (SSL) khi truyền thông tin nhạy cảm.

Các vấn đề Web farm

Một vấn đề khác khi sử dụng đối tượng Session để theo dõi trạng thái của người dùng là nếu bạn đang dùng một Web farm. Web farm là một nhóm các Web server cùng làm việc với nhau để phục vụ một Web site cá biệt. Mỗi lần người dùng tới một trang trên Web site của bạn, cầu dẫn (router) IP xác định máy nào không được dùng quá nhiều ngay bây giờ và hướng yêu cầu ấy cho trang của máy đó. Nếu người dùng được hướng tới một máy khác hơn máy mà đối tượng Session đã được tạo, thì server mới không có cách nào để truy tìm trạng thái đó và tất cả dữ liệu của người dùng bị mất.

Web.Config

Để làm giảm bớt vấn đề về các máy khác nhau phục vụ các trang khác nhau, bây giờ .NET có thể xác lập các đặc tính trong file Web.Config chỉ định tên của máy lưu trữ tất cả các biến phiên. Bằng cách này bạn có thể chuyên dùng một máy sẽ không làm gì ngoài quản lý các biến phiên cho site của bạn.

Mặc dù điều này giải quyết vấn đề của một Web farm, nhưng nó lại đưa ra các vấn đề của tự thân nó. Thí dụ, bạn cần thêm một máy này để duy trì và chạy 24/24. Bạn cũng có thể phải gặp tình trạng rầy rà của máy này để đáp ứng được liên tục trong khi bạn thực hiện các tác vụ bảo trì. Ngoài ra, một máy chính có thể

gây ra các vấn đề về hiệu suất, vì một máy có thể bị tắc nghẽn khi nhiều máy phải hoạt động trên mạng để nhận dữ liệu. Bạn sẽ tìm hiểu cách thay đổi xác lập này ở phần sau của chương này.

Không phải tất cả mọi người dùng đều chấp nhận các cookie bộ nhớ

Nếu bạn gặp các người dùng không chấp nhận các cookie bộ nhớ trong các trình duyệt của họ, bạn phải xác lập tùy chọn một cấu hình khác trong .NET để làm cho các biến phiên làm việc chính xác. Trước đây, các cookie bộ nhớ được đòi hỏi trên trình duyệt của client để làm cho Active Server Pages lưu giữ trạng thái. Bây giờ .NET Framework có thể làm việc mà không cần các cookie bộ nhớ mà bạn sẽ tìm hiểu cách thực hiện điều này ở phần "Các phiên không có cookie." sau của chương này.

Tắt các cookie từng trang

Bất kỳ khi nào gặp một trang ASPX, theo mặc định .NET thời gian chạy sẽ cố phát sinh một cookie tới trình duyệt. Nếu bạn biết một trang sẽ sử dụng một trạng thái phiên bất kỳ, bạn có thể xác lập EnableSessionState Page Directive là False để tắt sự phát sinh tự động này trên cơ sở từng trang. Ở trên đầu mỗi trang ASPX, bạn sẽ thấy Page Directive trông như sau:

```
<%@ Page Language="vb" AutoEventWireup="false" _
 Codebehind="SessionTestError.vb" _
 Inherits="DotNetStateMgmt.SessionTestError" _
 EnableSessionState="False" _
%>
```

Thêm dẫn hướng EnableSessionState, như được trình bày ở mã trên và không có cookie nào sẽ được phát sinh cho trang này.

Cảnh báo:

Nếu bạn cố sử dụng đối tượng Session trên trang có EnableSessionState Page Directive được xác lập là False, bạn sẽ nhận lỗi thời gian chạy.

Sử dụng Cookie

Nếu bạn không muốn lưu trữ trạng thái của chương trình trên server, bạn có thể gửi trạng thái tới trình duyệt của client. Bạn thực hiện điều này bằng cách sử dụng các cookie. Rất có thể bạn phải dùng cookie bộ nhớ, vì các cookie này bị hủy khi người dùng đóng trình duyệt của chúng. Để tạo cookie bộ nhớ, bạn sử dụng đối tượng Response như được trình bày ở mã bên dưới.

```
Response.Cookies("Email").Value = "JohnDoe@yahoo.com"
```

Một phương thức khác tạo một cookie mới là sử dụng phương thức Add của tập hợp Cookies trên đối tượng Response. Bạn phải tạo một đối tượng System.Web.HttpCookie và gửi tên và giá trị tới cấu tử cho đối tượng đó.

```
Response.Cookies.Add(New _
 System.Web.HttpCookie("Email", "JohnDoe@yahoo.com"))
```

Cả hai phương thức được trình bày ở trên sẽ xác lập cookie bộ nhớ vào trình duyệt của người dùng cho bạn. Điều này giả định là người dùng cho phép các cookie bộ nhớ vào trình duyệt của họ.

Để truy tìm cookie bộ nhớ trên bất kỳ trang kế tiếp này hoặc thậm chí trên cùng trang, sử dụng đối tượng Request. Đầu tiên bạn sẽ kiểm tra xem cookie bộ nhớ đó đã được tạo hay chưa. Nếu bạn cố truy xuất tập hợp Cookies và chuyển tên biến chưa được tạo vào, bạn sẽ nhận lỗi thời gian chạy.

```
If Request.Cookies("Email") Is Nothing Then
Else
   txtEmail.Text = Request.Cookies("Email").Value
End If
```

Ở mã trên, bạn kiểm tra xem đối tượng Request.Cookies("Email") có phải là Nothing hay không. Nếu phải, bạn sẽ không thể làm gì với cookie đó. Nếu đối tượng có thứ gì đó bên trong nó, bạn có thể truy tìm giá trị và gán nó vào hộp văn bản trên form.

Các cookie thường xuyên

Nếu bạn muốn lưu trữ một cookie vào đĩa cứng của người dùng, bạn phải xác lập đặc tính Expires trên cookie đó. Mã bên dưới trình bày một thí dụ về cách bạn tạo một cookie được gọi là EmailPerm và xác lập thời hạn trong 30 ngày.

```
Response.Cookies("EmailPerm").Value = txtEmail.Text
Response.Cookies("EmailPerm").Expires                    =
Today.AddDays(30)
```

Bằng cách xác lập đặc tính Expires là ngày tháng trong tương lai, trình duyệt lưu trữ cookie này vào một folder trên đĩa cứng của người dùng. Người dùng có thể xác lập trình duyệt của họ để chỉ cho phép các cookie bộ nhớ và không cho phép các cookie thường xuyên. Nếu người dùng xác lập trình duyệt của họ để không cho phép các cookie thường xuyên, thì dữ liệu sẽ không được lưu trữ. Bạn sẽ không nhận lỗi cookie không thể được lưu trữ, nhưng bạn sẽ không nhận được dữ liệu trở lại khi yêu cầu cookie ấy. Nếu bạn muốn xóa một cookie đĩa, xác lập đặc tính Expires thành một ngày trong quá khứ.

Các vấn đề với các cookie

Việc sử dụng các cookie cho bạn các khả năng quản lý trạng thái rất tốt, vì thực hiện chúng đơn giản và chúng giúp bạn di chuyển các tài nguyên khỏi server. Giống như bất kỳ kĩ thuật nào, các cookie có một số giới hạn.

Người dùng không cho phép các cookie

Một số người dùng tin là các virut có thể được gửi vào cookie Some và sẽ không cho phép chúng vào các máy tính của họ. Mặc dù chưa có tài liệu nào chứng minh các trường hợp này xảy ra và thực tế không ai có thể gửi virut bằng cookie, nhưng nhiều người dùng vẫn tắt khả năng chấp nhận các cookie. Khi điều này xảy ra, người dùng sẽ không thể dùng site của bạn nếu bạn sử dụng các cookie để quản lý trạng thái.

Hiệu suất

Hãy hình dung một người dùng lướt trên Web site của bạn bằng wizard, khi bạn thu thập được 100 đoạn dữ liệu từ người dùng đó trên một số trang. Mỗi trang cần phải gửi dữ liệu đã thu thập tới server. Nếu bạn chờ cho tới khi 100 đoạn dữ liệu được thu thập, bạn phải lưu trữ dữ liệu đó ở một nơi trong khi chờ đợi. Nếu bạn tiếp tục đưa dữ liệu vào cookie, có nhiều dữ liệu sẽ được gửi tới lui giữa trình duyệt và server. Điều này sẽ ngốn nhiều băng thông và có thể làm chậm toàn bộ site của bạn. Nhớ là dữ liệu phải đi cả hai chiều cho mỗi trang mà các người dùng gặp trên site.

Bộ nhớ

Một số trình duyệt đặt ra một giới hạn về kích thước của dữ liệu cookie mà chúng có thể chấp nhận hoặc số cookie chúng có

thể chấp nhận cùng một lúc. Ngoài ra, khối lượng bộ nhớ mà bạn có thể dùng trên máy của người dùng có thể khiến cho hệ điều hành của họ trao đổi bộ nhớ với đĩa. Khi đó cookie làm chậm máy của người dùng cũng như server.

Sử dụng StateBag

Trong một số trường hợp, bạn không cần duy trì trạng thái trên các trang, mà chỉ giữa các lời gọi cùng trang. Nếu bạn cần làm điều này, bạn có thể dùng đối tượng StateBag. Ở trang Web được trình bày ở hình 1, bạn nhập liệu ba giá trị. Bạn có thể đệ trình chúng với server và hiển thị dữ liệu nào đó vào nhãn Result. Cùng lúc này, bạn tạo ba biến trong StateBag bằng cách dùng đối tượng State.

Trong thủ tục sự kiện Click của nút Create State, viết mã sau để hiển thị dữ liệu và tạo đối tượng ViewState:

```
Public Sub btnSubmit_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnSubmit.Click
    ViewState("First") = txtFirst.Text
    ViewState("Last") = txtLast.Text
    ViewState("Password") = txtPassword.Text

    lblResult.Text = txtLast.Text & ", " & _
     txtFirst.Text & " (" & txtPassword.Text & ")"
    lblStateResult.Text = ViewState.Count.ToString()
End Sub
```

Ở mã được liệt kê ở trên, bạn sử dụng đối tượng ViewState giống như đối tượng phiên. Tạo một biến mới bằng cách chỉ cần liệt kê tên biến trong các dấu ngoặc, các dấu trích dẫn và gán giá trị mới cho nó. Mỗi giá trị đến từ các hộp văn bản trên trang.

Hình 1: *Sử dụng màn hình này để kiểm tra đối tượng StateBag.*

Trong khi bạn còn ở trên cùng trang, bạn có thể kiểm tra xem các giá trị trong StateBag thực sự được duy trì trong cuộc dạo đó. Bạn có thể nhấp nút Check State để hiển thị lại các giá trị từ StateBag.

```
Public Sub btnStateCheck_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles
btnStateCheck.Click
    lblResult.Text = ViewState("Last").ToString() & _
     ", " & ViewState("First").ToString()
    lblStateResult.Text = ViewState.Count.ToString()
End Sub
```

Các StateBag chỉ hợp lệ khi trang hoạt động. Ngay khi bạn điều hướng tới một trang khác, StateBag được loại bỏ. Nếu bạn

thực sự muốn cho StateBag này kiểm tra, bạn có thể nhấp nút Create State và sau đó khởi động lại dịch vụ IIS. Sau khi khởi động lại dịch vụ IIS, nhấp nút Check State và thấy tất cả các giá trị của bạn được duy trì. StateBag với tất cả dữ liệu đã được duy trì trong một trường ẩn trên form. Đây là một tính năng tốt vì điều đó có nghĩa là việc khởi động lại Web server sẽ không ảnh hưởng tới dữ liệu bạn đang lưu trữ cho một trang đó.

Các vấn đề với StateBag

StateBag là một cơ chế khá tốt. Nó cho bạn khả năng lưu trữ trạng thái cho một trang cá biệt. Nhưng giống như hầu hết các kĩ thuật khác, nó cũng kèm theo một số vấn đề.

Chỉ một trang đơn giản

StateBag chỉ hợp lệ đối với một trang đơn giản. Đối với nhiều trang, bạn phải sử dụng một cấu trúc khác.

Hiệu suất

Bạn càng lưu trữ nhiều trong StateBag, càng nhiều dữ liệu phải trao đổi với trình duyệt của client. Điều này có thể gây ra các vấn đề về hiệu suất trên toàn bộ site của bạn. Nhớ là dữ liệu phải di chuyển tới lui với lần gửi form trở lại server.

Bộ nhớ

Bộ nhớ có thể trở thành có vấn đề vì các trang StateBag có thể tăng lên nhanh và chiếm một khối lượng lớn bộ nhớ trên máy của người dùng. Điều này có thể khiến hệ điều hành của người dùng trao đổi bộ nhớ với đĩa và làm chậm máy của người dùng thậm chí nhiều hơn.

Phiên không có Cookie

Nếu biết bạn sẽ chạy trong một Web farm (được mô tả ở phần kế tiếp) hoặc bạn không muốn tận dụng cơ hội sử dụng các cookie, bạn có thể cấu hình ASP.NET để chạy không sử dụng các cookie. Bạn vẫn có thể dùng đối tượng Session, nhưng cookie bộ nhớ sẽ không được phát sinh. Bạn có thể thực hiện điều này bằng cách thay đổi thuộc tính Cookieless ở xác lập SessionState trong file Web.Config. Khi bạn thay đổi xác lập này, SessionID được gửi tới lui trên dòng URL hoặc trên mỗi siêu liên kết hoặc câu lệnh Action trong form của bạn. .NET Framework tự động thêm ID phiên và bạn chỉ phải thay đổi một xác lập! Nó hoạt động cho tất cả các trang trong Web site của bạn bất kể chúng là trang HTML hay ASPX.

Để cho phép các phiên không có cookie, bạn phải tạo một thay đổi ở file Web.Config trong ứng dụng ASP.NET. Thực hiện các bước sau để cho phép một phiên không có cookie.

1. Mở file Web.Config trong trình soạn thảo Visual Studio.NET.

2. Định vị chuỗi <sessionstate> XML.

3. Thay đổi thuộc tính Cookieless từ False thành giá trị True.

```
<sessionState
    mode="InProc"
    stateConnectionString="tcpip=127.0.0.1:42424"
    sqlConnectionString="data      source=127.0.0.1,user
id=sa;password="
    cookieless="True"
    timeout="20"
/>
```

Sau khi bạn thực hiện xong điều này, chạy ứng dụng lại và xem dòng URL. Bạn sẽ thấy một ID phiên xuất hiện với mỗi trang được hiển thị. ID này được gắn với mỗi và mọi lời gọi tới một trang bất kỳ trong Web site này. Engine của ASP.NET xử lý tất cả các chi tiết để gửi ID này trở lại đối tượng Session.

Quản lý trạng thái của Web Farm

Một trong các vấn đề bạn đã tìm hiểu trước là quản lý trạng thái trên một Web farm. Khi người dùng vào một Web site có một số server có thể phục vụ người dùng vào bất kỳ lúc nào, thì trạng thái phiên không tự động chuyển từ máy này sang máy kia. Một vấn đề khác xảy ra với trạng thái phiên là nó thường chạy trong cùng không gian xử lý như dịch vụ IIS. Do đó, nếu dịch vụ IIS không còn đáp ứng được nữa, thì bạn cũng mất tất cả các trạng thái phiên.

Lưu ý:

Quản lý trạng thái Web Farm không hoạt động với Public Beta 2. Nó sẽ có thể sử dụng ở phiên bản cuối cùng và sẽ làm việc như được mô tả ở chương này.

Nếu bạn muốn giải quyết cả hai vấn đề này, bạn phải di chuyển quản lý trạng thái phiên tới một thành phần bên ngoài tiến trình được tách khỏi dịch vụ IIS. Khi .NET Framework được cài đặt, một dịch vụ mới được gọi là ASP.NET State được cài đặt trên server Windows 2000 hoặc Windows NT. Dịch vụ này được dùng để quản lý đối tượng Session trong một tiến trình tách riêng. Tiến trình tách riêng này có thể được định vị trên cùng một máy hoặc một máy tách riêng.

Lưu ý:

Dịch vụ ASP.NET không được hỗ trợ trên Windows 95/98 hoặc Windows 2000 Professional.

Nếu bạn chọn sử dụng một server tách riêng làm máy quản lý trạng thái, thì tất cả các server trong Web farm của bạn sử dụng máy này để lưu trữ và truy tìm trạng thái cho người dùng. Bất kể máy nào phục vụ người dùng, trạng thái của chúng vẫn được duy trì trên một máy tách riêng, một máy mà mỗi Web server này có thể truy tìm dữ liệu.

Có lẽ bạn sẽ nghĩ là điều này sẽ rất khó khi thiết lập và duy trì. Không có gì quá phức tạp! Thực ra, chỉ cần thay đổi một xác lập trong mỗi file Web.Config của Web server.

Thực hiện các bước sau để cho phép trình quản lý trạng thái phiên ngoài tiến trình.

1. Khởi động ASP.NET State Service.
2. Mở file Web.Config trong trình soạn thảo Visual Studio.NET.
3. Định vị chuỗi <sessionstate> XML.
4. Đổi thuộc tính Mode từ InProc thành giá trị StateServer.
5. Chắc chắn thuộc tính Cookieless được xác lập là True.

```
<sessionState
    mode="StateServer"
    stateConnectionString="tcpip=127.0.0.1:42424"
    sqlConnectionString="data      source=127.0.0.1;user
id=sa;password="
    cookieless="True"
```

```
    timeout="20"
/>
```

Sau khi xác lập thuộc tính này, chạy một số mã để tạo biến phiên. Kế tiếp, dừng và khởi động lại IIS Admin và Web Publishing Services. Bây giờ trở lại trang truy tìm biến phiên mà bạn đã xác lập trước đây và bạn sẽ thấy biến đó vẫn tồn tại. Nếu bạn sẽ sử dụng một máy tách riêng để quản lý trạng thái, chắc chắn là dịch vụ ASP.NET State đang chạy trên máy này. Kế tiếp, xác lập thuộc tính stateConnectionString trong chuỗi <sessionstate> XML là tên hoặc địa chỉ IP của máy sẽ quản lý trạng thái.

Các vấn đề với dịch vụ ASP.NET State

Còn có một số vấn đề khi sử dụng dịch vụ ASP.NET State.

Hiệu suất

Hiệu suất truy tìm trạng thái từ một dịch vụ bên ngoài tiến trình sẽ chậm hơn từ một dịch vụ bên trong tiến trình. Nếu bạn đang truy tìm dữ liệu trên mạng tới một server trạng thái, bạn cũng gặp phải việc tranh chấp khả năng tải. Điều này có thể làm chậm việc truy tìm trạng thái đáng kể.

Tình trạng phức tạp

Nếu bạn sử dụng một máy khác để quản lý trạng thái, bạn sẽ tạo ra tình trạng phức tạp trong trường hợp máy này bị hu. Đương nhiên, tình trạng này sẽ không giúp gì cho bạn nếu máy gốc bị chết, vì tất cả dữ liệu phiên được lưu trữ trong bộ nhớ.

Quản lý trạng thái SQL Server tự động

Nếu bạn sử dụng một CSDL SQL Server sẵn có, bạn nên xét việc di chuyển quản lý trạng thái phiên vào nó, nhất là nếu trạng thái phiên có tầm quan trọng lớn đối với ứng dụng của bạn và bạn không thể đài thọ khi làm mất trạng thái phiên của người dùng. Khi bạn gửi dữ liệu phiên liên tục tới CSDL SQL Server, dữ liệu sẽ không những tiếp tục tồn tại khi khởi động lại các dịch vụ Web, mà còn cả khi khởi động toàn bộ máy.

Thực hiện các bước sau để sử dụng SQL Server để quản lý trạng thái.

1. Mở file Web.Config trong trình soạn thảo Visual Studio.NET.

2. Định vị chuỗi <sessionstate> XML.

3. Đổi thuộc tính Mode tới SQLServer.

4. Chắc chắn thuộc tính Cookieless được xác lập là giá trị True.

5. Đổi thuộc tính sqlConnectionString để nó trỏ vào server của bạn (Data Source) và sử dụng User ID và Password hợp lệ. Bạn không cần chỉ định tên của CSDL, vì các bảng quản lý trạng thái được định vị ở Tempdb.

```
<sessionState
    mode="SQLServer"
    stateConnectionString="tcpip=127.0.0.1:42424"
    sqlConnectionString="data        source=(local);user
id=sa;password="
    cookieless="True"
    timeout="20"
/>
```

Sau khi xác lập các thuộc tính này, bạn phải tạo CSDL ASPState với một số thủ tục lưu trữ mà .NET Framework sẽ dùng để quản lý trạng thái. Còn có một file có tên InstallSqlState.sql được định vị trong folder <systemdrive>\Winnt\Microsoft.NET\Framework\<Version>. Nạp file InstallSqlState.sql này vào SQL Query Analyzer và thi hành các câu lệnh. Điều này tạo CSDL ASPState và tất cả các thủ tục lưu trữ thích hợp.

Sau khi bạn thực hiện xong các điều này, bạn có thể chạy thử mẫu tạo biến phiên. Bạn có thể dừng và khởi động dịch vụ IIS và Web Publishing, một lần nữa trạng thái của bạn vẫn tồn tại. Nếu bạn mở SQL Server Enterprise Manager và điều hướng tới CSDL Tempdb, bạn sẽ thấy một bảng có tên ASPStateTempSessions. Mở bảng này và bạn sẽ thấy một bản ghi với ID phiên của bạn, thời gian phiên này đã được tạo và khi phiên này hết hạn. Bạn cũng sẽ thấy một số trường nhị phân. Đây là nơi dữ liệu của phiên này được lưu. Bạn sẽ không thể xem dữ liệu này, nhưng thực ra sau đó bạn không cần nữa. Vì .NET Framework tự động đảm nhiệm tất cả điều này cho bạn.

Các vấn đề với quản lý trạng thái SQL Server tự động

Khi bạn sử dụng kĩ thuật này, có một số vấn đề.

Chỉ dành cho SQL Server

Kĩ thuật này chỉ có thể sử dụng SQL Server vì các thủ tục lưu trữ được viết cho SQL Server. Nếu bạn không dùng CSDL SQL Server có sẵn, bạn sẽ không thể sử dụng giải pháp này.

Hiệu suất

Giống như bất kỳ một kĩ thuật quản lý trạng thái nào, hiệu suất của bạn có thể giảm xuống còn rất ít. Nó chỉ còn một ít thời gian để tạo kết nối và đọc/ghi vào CSDL.

Sử dụng một engine CSDL khác

Nếu bạn không thể dùng CSDL SQL Server sẵn có (có lẽ bạn đang sử dụng Oracle, Microsoft Access, Sybase hoặc một engine CSDL khác), bạn sẽ phải ghi quản lý trạng thái phiên riêng. Thực ra thực hiện tiến trình này đơn giản; bạn chỉ cần tạo một lớp riêng để lưu giữ dữ liệu.

Ở phiên này, bạn tạo lớp riêng bằng cách:

- Tạo một bảng trong CSDL của bạn để lưu giữ thông tin trạng thái.
- Tạo lớp SQLState để lưu trữ dữ liệu vào bảng CSDL.

Khi bạn thực hiện các tác vụ này, có thể bạn phải thay đổi chuỗi nhà cung cấp ADO.NET trong các listing mã để so khớp với chuỗi kết nối bạn cần kết nối với engine CSDL riêng biệt.

Bảng SessionState

Table bạn tạo phải có thể nhận dạng mỗi người dùng. Session.SessionID có thể được dùng như các khóa chính cho bảng này để nó được bảo đảm là duy nhất cho IIS server. Bên dưới là một mẫu về bảng này có thể trông như thế nào:

```
CREATE TABLE SessionState
(
   szSession_id Char(30) NOT NULL
        PRIMARY KEY,
   szState_tx Text NULL,
   dtLastUpdate_dt DateTime NULL,
)
```

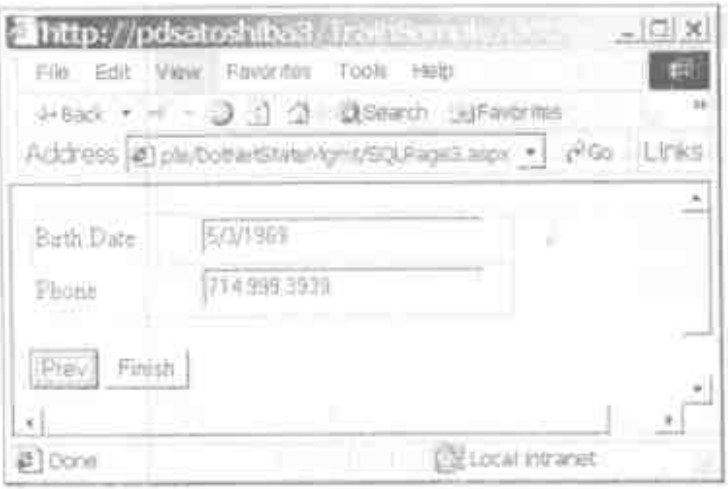
Bảng này có ba trường sẽ lưu giữ ID phiên, trạng thái và giờ và ngày tháng cuối cùng phiên này đã được dùng. Trường szSession_id được dùng để lưu trữ Session.SessionID từ IIS. Trường Text, szState_tx được dùng để lưu trữ một hoặc nhiều giá trị. Các giá trị này được lưu trữ như các cặp khóa ở định dạng *Key=Value*. Mỗi cặp khóa này được tách bằng một gạch dọc (|). Bạn phải sử dụng một kí tự nào đó mà người dùng không thể gõ vào khi làm việc bình thường với Web site của bạn: gạch dọc thường ít được dùng nên nó có thể là một kí tự đáp ứng điều này tốt.

Thí dụ về lưu trữ trạng thái

Thay vì viết các câu lệnh SQL trên mọi trang cần lưu trạng thái, tốt hơn là nếu bạn tạo một lớp thực hiện công việc đó cho bạn. Xét các form sau. Mỗi form yêu cầu người dùng về một số đoạn thông tin. Nếu bạn cần theo dõi thông tin đó từ trang này tới trang kế tiếp, bạn sẽ cần lưu trữ thông tin ở một nơi nào đó.

Hình 2: Form Name.

Hình 3: Form email và Social Security.



***Hình 4:** Form số điện thoại và ngày tháng sinh.*

Khi người dùng nhấp nút Next hoặc Prev ở màn hình trên, trạng thái được lưu và người dùng được thể hiện với trang thích hợp. Mã ở một trong các nút Next thể hiện như sau:

```
Public Sub btnNext_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs)
    If StateSave() Then
        Response.Redirect("SQLPage2.aspx")
    End If
End Sub
```

Ở mã trên, hàm StateSave được gọi. Nó là một hàm Private mà bạn sẽ viết cho mỗi một Web Forms để lưu dữ liệu vào bảng SessionState. Mã của StateSave được thể hiện bên dưới.

```
Private Function StateSave() As Boolean
    Dim oState As SQLState

    Try
        oState = New SQLState(Session.SessionID)
        With oState
            .Add("Page1.txtFirst", txtFirst.Text)
            .Add("Page1.txtLast", txtLast.Text)
```

```
            .Write()
        End With
        Return True

    Catch oException As Exception
        lblError.Text = oException.Message & _
         " - " & oException.StackTrace
        Return False

    End Try
End Function
```

Xét và thực hiện mã ở trên rất đơn giản. Bạn khai báo một đối tượng kiểu SQLState. Nó là lớp đảm trách việc thêm các khóa và các giá trị vào tập hợp bên trong và sau đó thu thập tập hợp đó vào một chuỗi để đối tượng SQLState có thể dệ trình chuỗi ấy tới chương trình phía sau (back end).

Sau khi tạo đối tượng SQLState mới này, gọi ra phương thức Add cho mỗi giá trị dữ liệu bạn muốn lưu. Bạn sẽ cho một khóa phải là duy nhất trên tất cả các trang. Tốt nhất có thể dùng tên trang, một dấu chấm (.) và tên của control trên trang đó tương ứng với dữ liệu bạn muốn lưu. Sau khi thêm mỗi mục này, gọi ra phương thức Write và dữ liệu sẽ được ghi vào bảng SessionState.

Truy tìm dữ liệu

Nếu bạn lưu trữ dữ liệu vào bảng, rất có thể bạn sẽ muốn truy tìm dữ liệu trở lại. Mã mẫu sau trình bày cách truy tìm dữ liệu bạn vừa lưu trữ.

```
Private Sub FormShow()
    Dim oState As SQLState

    Try
        oState = New SQLState(Session.SessionID)
        If oState.Found Then
```

```
            txtFirst.Text = oState("Page1.txtFirst")
            txtLast.Text = oState("Page1.txtLast")
        End If

    Catch oException As Exception
        lblError.Text = oException.Message & " - " & _
         oException.StackTrace
    End Try
End Sub
```

Sự khác nhau duy nhất giữa lớp SQLState và đối tượng Session là bạn phải lập thể nghiệm nó và gửi SessionID vào.

Lớp SQLState

Chúng ta hãy xây dựng lớp SQLState. Bạn phải tạo một số đặc tính và phương thức để đảm trách tất cả tính năng bạn sẽ cần. Các bảng bên dưới cho bạn một cái nhìn tổng quát về mỗi đặc tính và các phương thức bạn sẽ cần tạo.

Ở bên trong, lớp SQLState sử dụng lớp System.Web.UI.StateBag để lưu trữ tất cả các cặp khóa dữ liệu mà bạn sẽ truy tìm. Khi một thể nhiệm mới của lớp SQLState được tạo, bạn chuyển đặc tính vào Session.SessionID của cấu tử New. Cấu tử New đi tới CSDL để xem có trạng thái nào của phiên này có sẵn trong bảng hay không. Nếu có, nó nạp các giá trị đó mà bạn có thể dùng chúng để ghi đè nếu cần.

Các đặc tính của lớp SQLState

Đặc tính	Mô tả
Bag	Cho trở lại tham chiếu với đối tượng StateBag cơ sở.
Found	Đặc tính chỉ đọc để cho trở lại bất kỳ các giá trị nào được tìm thấy khi thể nghiệm mới của đối tượng StateBag được tạo.

Item	Đặc tính mặc định của lớp này cho trở lại giá trị của một Key chỉ định được chuyển vào đặc tính này. Thí dụ: Console.Write(moStateBag("MyKey")).
SessionID	Nhận dạng duy nhất bản ghi trong SQL Server lưu trữ trạng thái của người dùng này. Giá trị này phải được chuyển khi tạo một StateBag mới. Thông thường bạn sẽ nhận được giá trị này từ đặc tính Session.SessionID.
State	Lưu trữ trạng thái ở định dạng chuỗi cặp khóa.

Các phần cứng Public của SQLStateClass

Phương thức	Mô tả
Add	Chuyển vào một giá trị khóa và một giá trị chuỗi cần được lưu trữ vào gói trạng thái.
Clear	Xóa đặc tính State, đối tượng StateBag và xác lập trường szState_tx trong bản ghi cho ID phiên này là Null
Delete	Xóa bản ghi trong bảng SessionState, nơi SessionID khớp với szSession_id trong bảng. Thông thường nó được gọi từ thủ tục sự kiện Session_End trong file Global.asax.
New	Cấu tử của lớp này được chuyển vào đặc tính Session.SessionID và điền toàn bộ StateBag với bất kỳ nội dung nào từ bảng.
Read	Đọc vào dữ liệu từ bảng SessionState và điền toàn bộ StateBag.
Write	Ghi thông tin từ StateBag vào bảng SessionState.

Các phương thức Private của SQLStateClass

Phương thức	Mô tả
Bag2String	Lấy StateBag và chuyển đổi nó thành chuỗi cặp khóa thích hợp cho việc ghi vào bảng SessionState.
ConnectStringBuild	Xây dựng chuỗi kết nối mà các thủ tục khác sử dụng để kết nối với SQL Server
String2Bag	Lấy chuỗi cặp khóa, thường được truy tìm từ bảng SessionState và chuyển đổi nó thành StateBag.

Mã nguồn của lớp SQLState

Chúng ta hãy xét một số thủ tục của lớp SQLState để hiểu về cách lớp này hoạt động. Ở mã mẫu trên, bạn đã tạo một thể nghiệm mới của lớp SQLState. Nó gọi ra cấu tử New. Bên dưới là mã cho phương thức này.

```
Public Sub New(ByVal strSessionID As String)
    moStateBag = New System.Web.UI.StateBag()

    If strSessionID.Length = 0 Then
        Throw New Exception("Must Pass in SessionID
         to Constructor in clsSQLState() Class")
    Else
        mstrSessionID = strSessionID
        Try
            Call Read()
        Catch oException As Exception
            Throw oException
        End Try
    End If
End Sub
```

Bạn chuyển vào đặc tính Session.SessionID khi tạo một thể nghiệm mới của lớp này. Thủ tục này kiểm tra chắc chắn giá trị

hợp lệ được chuyển vào hay không và nếu hợp lệ, gán nó vào biến mức module mstrSessionID. Kế tiếp, nó gọi phương thức Read của lớp này để xem có sẵn bản ghi với một số thông tin trạng thái trong bảng SessionState cho ID phiên cá biệt này hay chưa. Sau là mã của phương thức Read.

```
Public Function Read() As Boolean
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim strSQL As String

    ' Initialize Variables
    mboolFound = False
    mstrState = ""
    moStateBag.Clear()

    strSQL = "SELECT szState_tx FROM SessionState "
    strSQL &= "WHERE szSession_id='" & mstrSessionID &
"'"

    Try
        oCmd = New OleDb.OleDbCommand()
        With oCmd
            .Connection = New _

OleDb.OleDbConnection(ConnectStringBuild())
            .Connection.Open()
            .CommandText = strSQL
            .CommandType = System.Data.CommandType.Text

            oDR = .ExecuteReader( _
             CommandBehavior.SequentialAccess)
        End With
        If oDR.Read() Then
            mboolFound = True
            If IsDBNull(oDR.Item("szState_tx")) Then
                mstrState = ""
            Else
                mstrState = _
                 oDR.Item("szState_tx").ToString()
            End If
        Else
            mboolFound = False
        End If
        oDR.Close()
```

```
            oCmd.Connection.Close()

            If mstrState.Length > 0 Then
                Call String2Bag()
            End If

        Catch oException As Exception
            Throw oException
        End Try

        Return mboolFound
    End Function
```

Mặc dù có nhiều mã trong thủ tục này, nhưng hầu hết nó liên quan tới việc mở kết nối và đọc vào dữ liệu từ bảng. Chuỗi mà nó đọc có thể giống như sau:

```
Page1.txtFirst=John|Page1.txtLast=Doe|Page2.txtSSN=555.
55.5555|Page2.txtEmail=JohnDoe@yahoo.com
```

Sau khi đọc chuỗi vào, phương thức Read gọi phương thức String2Bag để tách nó và đặt mỗi phần tử chuỗi vào StateBag.

```
Private Sub String2Bag()
    Dim intLoop As Integer
    Dim astrValues() As String
    Dim strKey As String
    Dim strValue As String

    astrValues = mstrState.Split("|".ToCharArray())
    For intLoop = 0 To astrValues.Length - 1
        strValue = astrValues(intLoop).Trim()
        If strValue.Length > 0 Then
            strKey = strValue.Substring(0, _
             strValue.IndexOf("="))
            strValue = _
             strValue.Substring(strValue.IndexOf("=") +
1)
            Call Add(strKey, strValue)
        End If
    Next
End Sub
```

Thủ tục String2Bag chuyển đổi chuỗi này thành một mảng bằng cách dùng phương thức Split của lớp String. Sau đó nó lặp qua mảng mới này và đưa ra cặp khóa. Sau đó nó sử dụng phương thức Substring và IndexOf methods của lớp String để trích phần khóa và phần giá trị từ mỗi chuỗi. Cuối cùng nó gọi phương thức Add để thêm khóa mới và giá trị vào StateBag.

Sau là phương thức Add:

```
Public Sub Add(ByVal strKey As String, _
 ByVal strValue As String)
    moStateBag.Add(strKey, strValue)
End Sub
```

Phương thức Add sử dụng phương thức Add của lớp StateBag để thêm mục mới vào StateBag.

Một phương thức khác rất thường được gọi trong lớp SQLState là phương thức Write. Bên dưới là mã cho thủ tục này.

```
Public Sub Write()
    Dim oCmd As OleDb.OleDbCommand
    Dim strSQL As String

    mstrState = Bag2String()

    If mboolFound Then
        strSQL = "UPDATE SessionState SET "
        strSQL &= "szState_tx = '" & mstrState & "', "
        strSQL &= "dtLastUpdate_dt = '" & Now() & "' "
        strSQL &= "WHERE szSession_id = '" & _
         mstrSessionID & "' "
    Else
        strSQL = "INSERT INTO
SessionState(szSession_id, _
         szState_tx, dtLastUpdate_dt)"
        strSQL &= "VALUES('" & mstrSessionID & "',"
        strSQL &= "'" & mstrState & "', "
        strSQL &= "'" & Now() & "')"
    End If
```

```
    Try
        oCmd = New OleDb.OleDbCommand()
        With oCmd
            .Connection = New _

OleDb.OleDbConnection(ConnectStringBuild())
            .Connection.Open()
            .CommandText = strSQL
            .CommandType = System.Data.CommandType.Text

            .ExecuteNonQuery()
        End With
        oCmd.Connection.Close()
    Catch oException As Exception
        Throw oException
    End Try
End Sub
```

Thủ tục này xây dựng câu lệnh INSERT hoặc UPDATE dù tìm thấy hay không tìm thấy dữ liệu trong bảng SessionState. Sau đó nó sử dụng ADO.NET để đệ trình các truy vấn hành động này tới CSDL chương trình phía sau (back-end).

Các ưu điểm của lớp SQLState

Việc sử dụng CSDL để lưu trữ trạng thái của bạn có thể trợ giúp trong một số phương diện. Quan trọng nhất là cắt giảm khối lượng dữ liệu được gửi tới lui client. Thực ra, một thứ duy nhất phải được gửi tới lui là SessionID. Kĩ thuật này cũng rất thích hợp. Lý do mà nó thích hợp là máy chủ CSDL như SQL Server, Oracle hoặc Sybase được dùng để xử lý dữ liệu ở tốc độ cao và cho số lớn các người dùng. CSDL Microsoft Access sẽ không là giải pháp tốt lắm, vì nó khó thích ứng được với nhiều người dùng.

Kĩ thuật này làm việc trên các Web farm, vì tất cả các máy trong farm có thể dùng cùng CSDL. CSDL có thể được mở rộng hoặc thu hẹp trên một hoặc nhiều máy khi tranh chấp tải trên riêng một site gia tăng nhanh. Nếu bạn có nhiều người dùng và

cần phải quản lý trạng thái của các người dùng này nhiều hơn, bạn có thể cắm vào nhiều máy IIS và thêm các bộ vi xử lý, bộ nhớ và không gian đĩa cho máy chủ CSDL để mở rộng hầu như vô hạn. Một lý do khác để sử dụng lớp SQLState là nó dễ sử dụng. Thay vì phải viết mã SQL và mã StateBag ở mọi trang, bạn tạo một đối tượng để gọi ra một vài phương thức và tất cả việc quản lý trạng thái này được thực hiện cho bạn.

Các vấn đề với lớp SQLState

Lớp SQLState cũng không thoát khỏi có các vấn đề.

Hiệu suất

Giống như bất kỳ các kĩ thuật quản lý trạng thái nào, hiệu suất có thể giảm xuống nhiều. Nó còn ít thời gian để tạo đối tượng SQLState và StateBag. Nó cũng phải mất thời gian để tạo kết nối và đọc/ghi CSDL.

Tính phức tạp

Bạn phải viết, gỡ rối và duy trì khá ít mã. Đương nhiên, phần lớn mã được đề cập ở chương này, nhưng còn nhiều mã bạn phải viết để chịu được độ sai sót.

Khác với ASP như thế nào?

Các khả năng quản lý trạng thái trong ASP còn rất thô sơ so với ASP.NET. Đối tượng Session chỉ có thể chạy trong tiến trình với IIS server và không có bất kỳ các khả năng Web farm nào. Nếu bạn muốn dùng một Web farm, bạn buộc phải sắp xếp thành phần COM để lưu trữ trạng thái vào trong CSDL của SQL Server,

giống như lớp SQLState được trình bày ở chương này. Bạn có thể sử dụng các trường nhập liệu ẩn, nhưng bạn phải quản lý ViewState — ASP không làm điều này cho bạn như ASP.NET thực hiện. Không có StateBags nào, vì vậy bất kỳ trạng thái nào bạn cần duy trì khi ở trên một trang phải được quản lý trong các trường nhập liệu ẩn và bạn phải tự mã hóa.

Tóm tắt

Ở chương này bạn đã tìm hiểu về nhiều cách có thể duy trì trạng thái trên một Web site. Thậm chí trong một ít ứng dụng Web bạn cũng có thể phải sử dụng một hoặc nhiều kĩ thuật này. Một máy chủ CSDL như SQL Server cung ứng khả năng mềm dẻo và hiệu suất tốt, tuy nhiên, đôi khi bạn có thể phải dùng sự kết hợp của các kĩ thuật đã được trình bày ở chương này.

Câu hỏi ôn tập

1. Bạn dùng phương thức nào để hủy dứt khoát một phiên của người dùng?

2. Bạn tắt các cookie của một trang trong site của mình bằng cách nào?

3. Bạn tạo một cookie thường xuyên bằng cách nào?

4. Đúng hay sai: Một gói State vẫn tồn tại trong bộ nhớ cho tới khi người dùng thoát khỏi Web site của bạn?

5. Bạn xác lập thuộc tính nào khi không muốn dùng các cookie cho lưu trữ trạng thái phiên?

6. Bạn đổi thuộc tính Mode thành giá trị gì trong phần tử SessionState để sử dụng dịch vụ ASP.NET cho trạng thái phiên?

Bài tập

- Tắt việc sử dụng cookie cho một Web site.
- Tắt việc sử dụng cookie cho một Web site và sử dụng dịch vụ ASP.NET State.

Trả lời câu hỏi ôn tập

1. Session.Abandon.
2. Xác lập thuộc tính EnableSessionState trong phần tử Page là False
3. Xác lập đặc tính Expires thành một ngày tháng trong tương lai trên cookie của bạn.
4. Sai.
5. Cookiesless=True.
6. StateServer.

Chương 32

Web Service

- Web Service là gì?
- SOAP là gì và nó thích hợp với .NET như thế nào?

Web Service

Web service là một lớp và các phương thức có thể được gọi trên giao diện HTTP. Một ứng dụng client hoặc ứng dụng Web bất kỳ đều có thể gọi bất kỳ dịch vụ Web nào bất kể nó chạy ở đâu. Với điều kiện nó có thể trình bày thông qua giao diện HTTP chuẩn, bất kỳ chương trình nào cũng có thể sử dụng dịch vụ này.

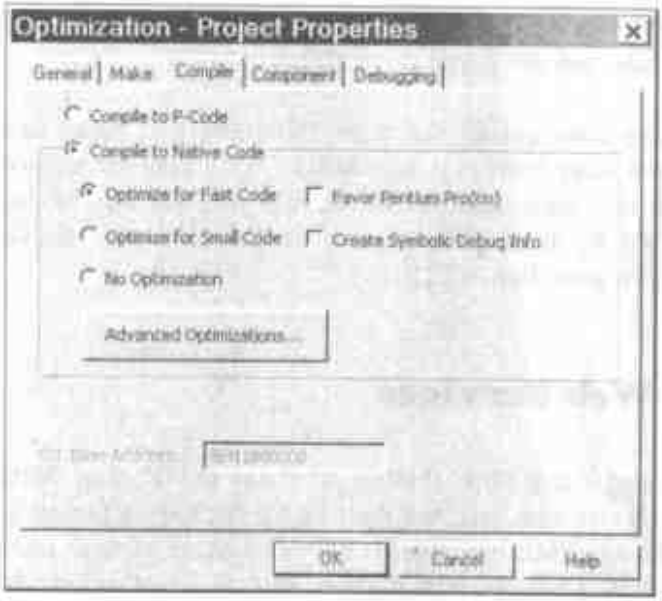
Cơ chế chuẩn để gọi dịch vụ Web là SOAP (Simplified Object Access Protocol – Giao thức truy xuất đối tượng đơn giản hóa). Nó không là gì khác hơn một phong bì XML để gói XML nào đó chứa tên đối tượng, tên phương thức và bất kỳ các tham số nào tùy chọn để gọi dịch vụ Web. Nó tùy thuộc vào ứng dụng client

mở socket TCP/IP và đệ trình yêu cầu này bằng giao diện HTTP tới URL chính xác, nơi đối tượng được định vị.

Một đối tượng được gọi có thể được viết bằng một ngôn ngữ bất kỳ và có thể chạy trên một nền bất kỳ. Các dịch vụ này có thể được gọi từ một ngôn ngữ bất kỳ và một nền bất kỳ. Tất cả truyền thông tới lui được thực hiện qua XML, vì vậy rất dễ truyền tải nó trên giao diện HTTP.

SOAP và Web Services

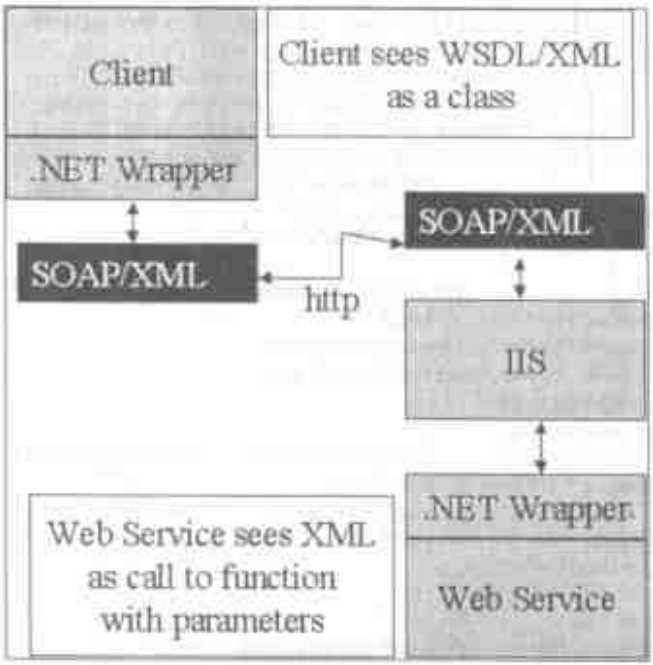
Trong một ứng dụng bình thường sử dụng SOAP, ứng dụng client đảm trách việc tạo XML mà dịch vụ ấy đòi hỏi và phong bì SOAP để truyền tải XML này trên HTTP. Ở bên nhận, bạn phải có listener SOAP. Listener (trình theo dõi) sẽ chọn phong bì SOAP, đọc nội dung và gọi ra đối tượng thích hợp trên server. Nó sẽ chờ đối tượng kết thúc tác vụ của nó, sau đó gửi kết quả XML trở lại máy client.



Hình 1: *SOAP đảm trách việc gọi dịch vụ Web từ một ứng dụng phía client.*

.NET và SOAP

Trước khi .NET xảy ra, việc tạo và sử dụng các phong bì SOAP đòi hỏi một ít mã và phải hiểu về cả định dạng XML lẫn SOAP. .NET đặt một wrapper bao quanh cả việc ghi các đoạn mã phía client và server mà không đòi hỏi bất kỳ kiến thức nào về XML hoặc SOAP! Đây là một sự cải tiến lớn và làm cho việc tạo Web Services nhanh chóng và dễ dàng hơn trước đây rất nhiều.



Hình 2: *SOAP vẫn được dùng trong .NET, nhưng .NET gói nó như một đối tượng tốt cho bạn.*

Ngôn ngữ mô tả Web Service

File XML mô tả dịch vụ Web sử dụng một định dạng đặc biệt và được gọi là WSDL (Web Service Description Language – Ngôn ngữ mô tả dịch vụ Web). Nếu bạn chỉ tạo một dịch vụ Web đơn giản có tên AddTwoNumbers, một lớp được gọi là Adding và một phương thức được gọi là AddTwoNumbers chấp nhận hai giá trị số nguyên, thì định dạng sẽ thể hiện như sau:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding
/"                          xmlns:s0="http://tempuri.org/"
targetNamespace="http://tempuri.org/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <s:schema           attributeFormDefault="qualified"
elementFormDefault="qualified"
targetNamespace="http://tempuri.org/">
      <s:element name="AddTwoNumbers">
        <s:complexType>
          <s:sequence>
            <s:element    minOccurs="1"    maxOccurs="1"
name="intNumber1" type="s:int" />
            <s:element    minOccurs="1"    maxOccurs="1"
name="intNumber2" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="AddTwoNumbersResponse">
        <s:complexType>
          <s:sequence>
            <s:element    minOccurs="1"    maxOccurs="1"
name="AddTwoNumbersResult" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="int" type="s:int" />
    </s:schema>
  </types>
  <message name="AddTwoNumbersSoapIn">
    <part  name="parameters"  element="s0:AddTwoNumbers"
/>
  </message>
  <message name="AddTwoNumbersSoapOut">
    <part                                  name="parameters"
element="s0:AddTwoNumbersResponse" />
  </message>
  <message name="AddTwoNumbersHttpGetIn">
    <part name="intNumber1" type="s:string" />
    <part name="intNumber2" type="s:string" />
  </message>
  <message name="AddTwoNumbersHttpGetOut">
```

```
    <part name="Body" element="s0:int" />
  </message>
  <message name="AddTwoNumbersHttpPostIn">
    <part name="intNumber1" type="s:string" />
    <part name="intNumber2" type="s:string" />
  </message>
  <message name="AddTwoNumbersHttpPostOut">
    <part name="Body" element="s0:int" />
  </message>
  <portType name="AddingSoap">
    <operation name="AddTwoNumbers">
      <input message="s0:AddTwoNumbersSoapIn" />
      <output message="s0:AddTwoNumbersSoapOut" />
    </operation>
  </portType>
  <portType name="AddingHttpGet">
    <operation name="AddTwoNumbers">
      <input message="s0:AddTwoNumbersHttpGetIn" />
      <output message="s0:AddTwoNumbersHttpGetOut" />
    </operation>
  </portType>
  <portType name="AddingHttpPost">
    <operation name="AddTwoNumbers">
      <input message="s0:AddTwoNumbersHttpPostIn" />
      <output message="s0:AddTwoNumbersHttpPostOut" />
    </operation>
  </portType>
  <binding name="AddingSoap" type="s0:AddingSoap">
    <soap:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <operation name="AddTwoNumbers">
      <soap:operation
soapAction="http://tempuri.org/AddTwoNumbers"
style="document" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <binding                              name="AddingHttpGet"
type="s0:AddingHttpGet">
    <http:binding verb="GET" />
    <operation name="AddTwoNumbers">
```

```
      <http:operation location="/AddTwoNumbers" />
      <input>
        <http:urlEncoded />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
  <binding                              name="AddingHttpPost"
type="s0:AddingHttpPost">
    <http:binding verb="POST" />
    <operation name="AddTwoNumbers">
      <http:operation location="/AddTwoNumbers" />
      <input>
        <mime:content      type="application/x-www-form-
urlencoded" />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
  <service name="Adding">
    <port name="AddingSoap" binding="s0:AddingSoap">
      <soap:address
location="http://localhost/AddTwoNumbers/Adding.asmx"
/>
    </port>
    <port                                name="AddingHttpGet"
binding="s0:AddingHttpGet">
      <http:address
location="http://localhost/AddTwoNumbers/Adding.asmx"
/>
    </port>
    <port                               name="AddingHttpPost"
binding="s0:AddingHttpPost">
      <http:address
location="http://localhost/AddTwoNumbers/Adding.asmx"
/>
    </port>
  </service>
</definitions>
```

Có nhiều định nghĩa trong file này phải ở đó cho bạn để tương tác với server này. Hãy tưởng tượng nếu bạn phải tự viết tất cả

mã này! File được phát sinh tự động bởi .NET khi bạn tạo một dịch vụ Web. Điều này tiết kiệm được thời gian đáng kể.

Khám phá các dịch vụ

Nếu bạn muốn dùng một dịch vụ mà bạn phải biết định vị của dịch vụ Web. Bạn tới định vị này bằng cách dùng một URL chuẩn bạn sẽ gõ vào trên Internet. Bên dưới là một thí dụ.

```
http://localhost/AddTwoNumbers/AddTwoNumbers.vsdisco
```

File ở cuối được gọi là một file vsdisco (Discovery). File khám phá này cho bạn định vị của file WSDL mô tả phần còn lại của dịch vụ này.

```
<?xml version="1.0" encoding="utf-8"?>
<discovery  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/disco/">
  <contractRef
ref="http://localhost/AddTwoNumbers/Adding.asmx?wsdl"
docRef="http://localhost/AddTwoNumbers/Adding.asmx"
xmlns="http://schemas.xmlsoap.org/disco/scl/" />
</discovery>
```

UDDI

Nếu bạn không biết các dịch vụ Web nào có trên Internet, bạn nên kiểm tra www.uddi.org. Đây là một tổ chức các chuẩn mới sẽ tạo CSDL của tất cả các dịch vụ Web sẵn có trên thế giới. Bạn có thể tới URL này và đăng ký bất kỳ các dịch vụ Web nào bạn tạo cũng như tìm các dịch vụ Web khác đã tạo. Các chuẩn UDDI thay cho Universal Description, Discovery và Integration.

Các thí dụ về Web Services

Có nhiều thứ bạn có thể dùng Web services. Bên dưới là một số thí dụ về những kiểu dữ liệu nào bạn có thể tạo.

- TimeStamp chuẩn trên nhiều múi giờ.
- Trình chuyển đổi tỉ giá tiền tệ.
- Cho trở lại quyền sử dụng trên thẻ tín dụng.
- Cho trở lại tình trạng gửi hàng.
- Cho trở lại các xác nhận đơn đặt hàng.
- Cho trở lại bản yết giá chứng khoán.
- Cho trở lại thông tin danh mục/sản phẩm.
- Các ứng dụng phân phối.

Tóm tắt

Web Services có thể được dùng để truyền thông từ một máy này tới máy khác và sử dụng các dịch vụ từ các máy khác. Tất cả điều này đều được thực hiện qua giao thức Internet HTTP chuẩn. Nó tạo cho kiểu ứng dụng này là chính một nền chéo và dễ gọi.

Câu hỏi ôn tập

1. Bạn dùng chuẩn nào để gói một lời gọi dịch vụ Web?

2. Bạn dùng giao thức truyền nào để gọi một dịch vụ Web?

3. Đúng hay sai: Một dịch vụ Web chỉ có thể được viết trong .NET?

4. WSDL thay cho cái gì?

5. Bạn tìm các dịch vụ Web ở đâu trên Internet?

Trả lời câu hỏi ôn tập

1. SOAP.

2. HTTP.

3. Sai.

4. Web Service Description Language.

5. www.uddi.org.

Chương 33

Tạo và sử dụng các dịch vụ Web

- Tìm hiểu cách tạo một dịch vụ Web đơn giản.
- Tạo một dịch vụ Web cho trở lại dữ liệu Product.
- Sử dụng dịch vụ Web từ một ứng dụng WinForm.
- Sử dụng dịch vụ Web từ một ứng dụng Web.

Tạo một dịch vụ Web đơn giản

Bạn sẽ tạo dịch vụ Web đơn giản là cộng hai số với nhau và cho trở lại giá trị. Dù đây không phải là một dịch vụ thật hữu dụng nhưng nó sẽ cho bạn các bước cần thiết để tiến hành, bất kể bạn tạo kiểu dịch vụ nào.

1. Mở Visual Studio.NET và lựa ASP.NET Web Service từ danh sách các template.

2. Đặt tên là AddTwoNumbers. Nhấp OK để tạo project dịch vụ Web mới.

3. Đặt lại tên thành phần dịch vụ mặc định từ Service.asmx thành **Adding.asmx**.

4. Xem mã của file asmx này. Nó sẽ thể hiện như listing bên dưới.

```
Imports System.Web.Services

Public Class Service1
    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web Services
Designer.
        InitializeComponent()

        'Add your own initialization code after the
InitializeComponent() call

    End Sub

    'Required by the Web Services Designer
    Private components As
System.ComponentModel.Container

    'NOTE: The following procedure is required by the
Web Services Designer
    'It can be modified using the Web Services
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private
Sub InitializeComponent()
        components = New
System.ComponentModel.Container()
    End Sub
```

```
    Protected Overloads Overrides Sub Dispose(ByVal
disposing As Boolean)
        'CODEGEN: This procedure is required by the Web
Services Designer
        'Do not modify it using the code editor.
    End Sub

#End Region

    ' WEB SERVICE EXAMPLE
    ' The HelloWorld() example service returns the
string Hello World.
    ' To build, uncomment the following lines then save
and build the project.
    ' To test this Web service, ensure that the .asmx
file is the start page
    ' and press F5.
    '
    '<WebMethod()> Public Function HelloWorld() As
String
    'HelloWorld = "Hello World"
    ' End Function

End Class
```

5. Đổi mã chỉ Public Class Service1 thành **Public Class Adding**.

6. Ở đáy lớp này bỏ chú thích 3 dòng là hàm HelloWorld.

7. Thay đổi mã này như sau:

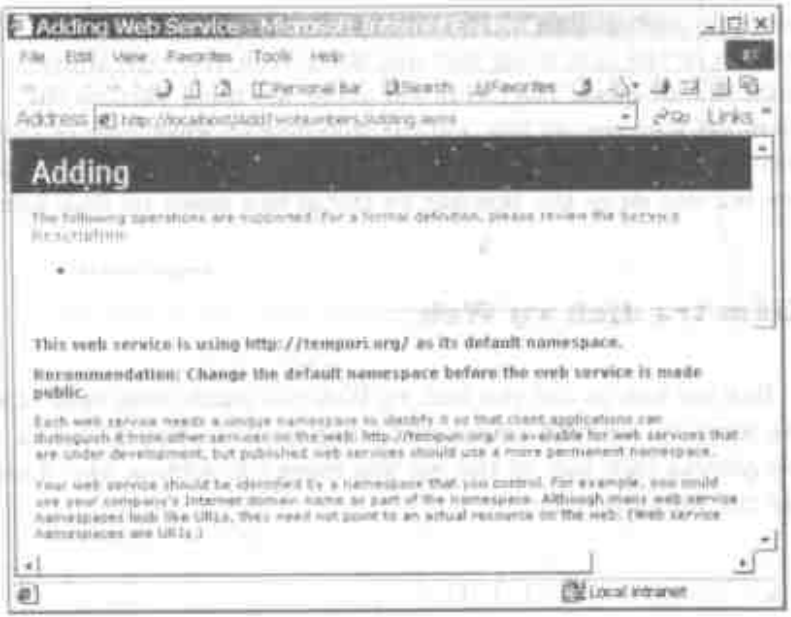
```
<WebMethod()> Public Function AddTwoNumbers( _
 ByVal intNumber1 As Integer, _
 ByVal intNumber2 As Integer) As Integer

    Return intNumber1 + intNumber2

End Function
```

Nếu bạn xét hàm này, thay vì là bí danh <WebMethod()> mà nó trông giống như mã VB.NET bình thường. Thực ra, nó chỉ là

mã bình thường. Bạn có thể làm bất kỳ thứ gì trong dịch vụ Web mà bạn có thể làm trong một ứng dụng Web, trừ hiển thị các phần tử trực quan. Thay vì là mã giống hệt nhau. Bạn có thể thêm mã truy tìm dữ liệu, gọi các thành phần khác, bạn có thể làm bất kỳ thứ gì! Với điều kiện bạn cho trở lại dịch vụ Web của bạn mã nào đó có thể làm bất kỳ thứ gì bạn muốn nó thực hiện.

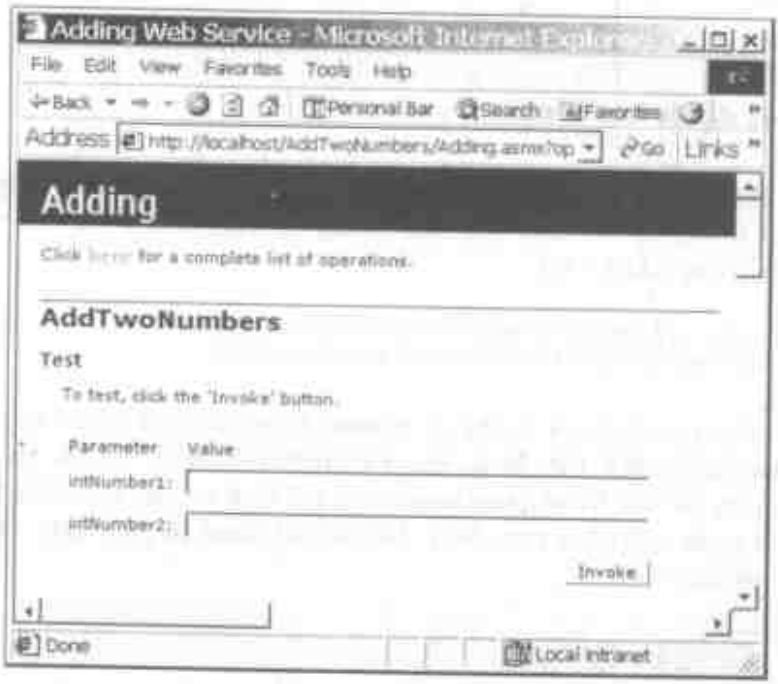
Kiểm tra dịch vụ Web

Bây giờ bạn có thể thử dịch vụ Web của mình bằng cách chạy ứng dụng ấy. VS.NET sẽ tạo một trang Web cho bạn để mô tả mỗi phương thức bạn có thể gọi bên trong lớp Adding này. Trang này thể hiện như sau:



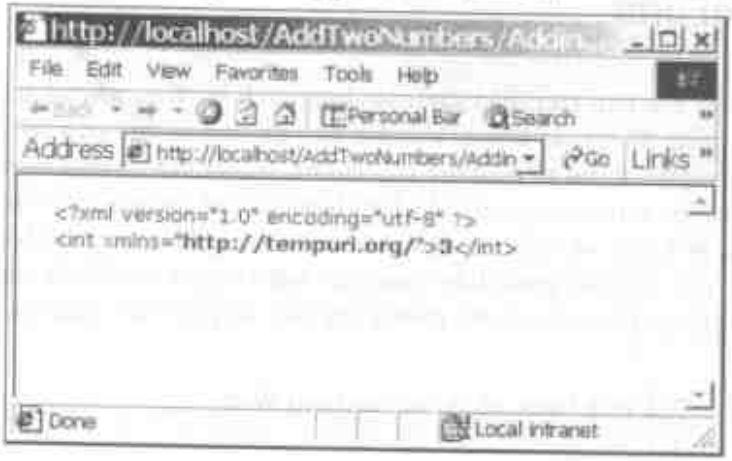
Hình 1: *Đây là trang mặc định được phát sinh khi bạn chạy một dịch vụ Web.*

Nếu bạn nhấp siêu liên kết AddTwoNumbers, dịch vụ ấy sẽ phát sinh một trang để cho phép bạn nhập hai giá trị. Nó tự động đọc dịch vụ của bạn và xác định các kiểu và bao nhiêu đoạn dữ liệu bạn cần nhập vào để gọi dịch vụ Web. Điều này được trình bày như ở hình 2.



***Hình 2:** Nhập hai số để kiểm tra phương thức AddTwoNumbers.*

Sau khi bạn nhập hai số, bạn có thể nhấp nút Invoke và sẽ thấy màn hình kết quả trông giống như hình 3.



Hình 3: Kết quả trở lại từ dịch vụ Web là XML.

Gọi dịch vụ Web

Nếu bạn xét dòng URL khi nhận kết quả trở lại bạn sẽ thấy một thứ gì đó thể hiện như sau. (Dòng đã được tách đôi để nó vừa bên trong trang này).

```
http://localhost/AddTwoNumbers/Adding.asmx/ _
AddTwoNumbers?intNumber1=1&intNumber2=2
```

Như bạn thấy nó sẽ chỉ gọi phương thức như một folder ảo dưới file .ASMX. Sau đó nó chuyển vào hai tham số với các giá trị bạn đã gõ vào. Vì vậy nếu bạn muốn gọi dịch vụ này trực tiếp từ một ngôn ngữ khác hơn .NET, bạn có thể dùng cú pháp này và nó sẽ làm việc.

Sử dụng dịch vụ Web từ một ứng dụng WinForm

Bây giờ bạn tìm hiểu cách sử dụng dịch vụ Web này từ một ứng dụng Windows Form. Vào lúc nào đó bạn có thể cần truy xuất một dịch vụ từ desktop của client. Nó có thể là một dịch vụ Web trên intranet cục bộ của bạn hoặc có thể là một dịch vụ được định vị ở một văn phòng cộng tác mà chỉ có thể truy xuất bằng Internet. Bất kể quan điểm nào, các bước thực hiện để sử dụng đều giống nhau. Các bước chung mà bạn sẽ phải làm như sau:

1. Tạo một tham chiếu với dịch vụ Web.

2. Tạo một đối tượng từ dịch vụ Web.

3. Gọi một phương thức trên dịch vụ Web.

Tạo ứng dụng Client

Bây giờ chúng ta hãy tiến hành các bước chi tiết để sử dụng dịch vụ AddTwoNumbers Web.

1. Mở một thể nghiệm của Visual Studio.NET.

2. Tạo một project mới như một kiểu template Windows Application.

3. Đặt tên là **AddTwoNumbersClient** và nhấp nút OK.

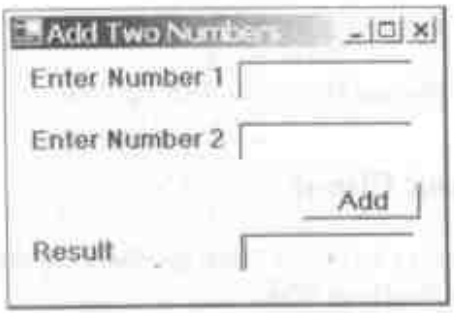
4. Đặt lại tên mặc định thành **frmAddTwoNumbers.vb**.

5. Xác lập đặc tính Name là **frmAddNumbers**.

6. Xác lập đặc tính Text là **Add Two Numbers**.

7. Nhấp nút chuột phải trên Project, lựa menu Properties và xác lập Startup Object là **frmAddNumbers**.

8. Bây giờ tạo form như hình 4.



Hình 4: *Dùng form này để sử dụng dịch vụ AddTwoNumbers Web.*

9. Tạo các hộp văn bản với các tên sau: **txtNum1**, **txtNum2** và **txtResult**.

10. Tạo nút với đặc tính Name được xác lập là **btnAdd**.

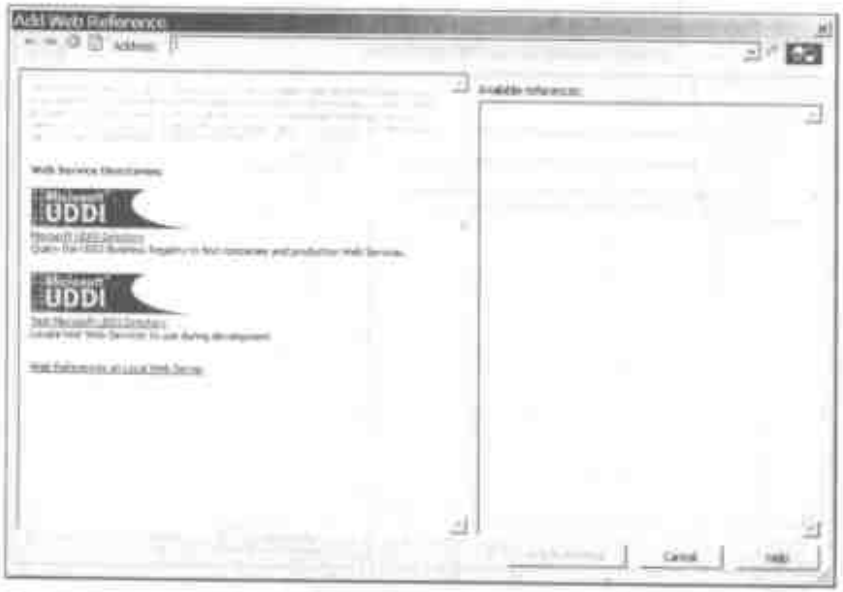
11. Bây giờ bạn sẵn sàng móc nối nút ấy để khi nó được nhấp nó sẽ gọi dịch vụ Web.

Xác lập tham chiếu Web

Để sử dụng dịch vụ Web từ ứng dụng này, bạn sẽ phải xác lập một tham chiếu với file vsdisco cho project dịch vụ Web. Thực hiện các bước sau để đưa vào tham chiếu với dịch vụ này.

1. Nhấp Project và lựa Project | Add Web Reference từ menu.

2. Bây giờ bạn sẽ thấy một màn hình như ở hình 5.



Hình 5: *Sử dụng màn hình này để gõ vào URL trỏ tới file VSDISCO của dịch vụ Web.*

Trên màn hình này bạn có thể tìm thư mục Microsoft UDDI hoặc bạn có thể nhấp liên kết cuối cùng để tìm các tham chiếu Web trên Web server cục bộ. Một khi bạn tìm thấy dịch vụ Web bạn muốn sử dụng, trong trường hợp của bạn là dịch vụ AddTwoNumbers, bạn sẽ lựa nó từ danh sách ở cửa sổ bên phải. Bây giờ sẽ thấy một màn hình như ở hình 6.

Hình 6: *AddTwoNumbers.vsdisco hiển thị tham chiếu của nó cần được thêm vào project.*

Nhấp Add Reference để thêm tham chiếu này vào ứng dụng Windows của bạn.

Bây giờ cửa sổ Solution Explorer sẽ trình bày một folder Web References mới. Nếu bạn mở rộng folder này, bạn sẽ thấy một localhost với .WSDL, .DISCO và file Reference.map ở trong nó như được trình bày ở hình 7.

***Hình 7:** Một tham chiếu Web sẽ trình bày tất cả file wsdl và disco được phát hiện ở tham chiếu bạn đưa vào.*

Dù bạn không cần làm thử gì với các file này nhưng bạn nên biết chúng ở đó.

Gọi dịch vụ Web

Bây giờ bạn đã tạo tham chiếu này, bạn có thể viết mã sẽ sử dụng dịch vụ Web.

1. Nhấp đôi nút Add.

2. Thêm mã sau vào thủ tục sự kiện Click.

```
Private Sub btnAdd_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAdd.Click
        Dim ws As localhost.Adding

        ws = New localhost.Adding()
```

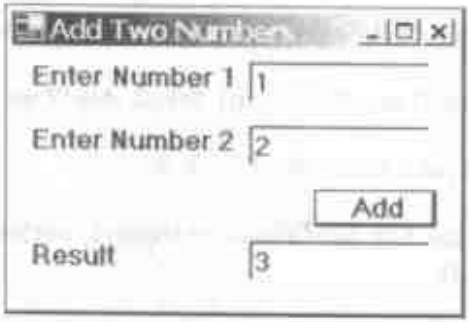
```
        txtResult.Text = _
            ws.AddTwoNumbers(CInt(txtNum1.Text), _
            CInt(txtNum2.Text))

    End Sub
```

Ở mã trên bạn khai báo biến *ws*, như tên của dịch vụ Web. Bạn sử dụng tên bạn thấy trong Solution Explorer, *localhost* ở trường hợp này. Bạn có thể đổi tên này nếu muốn. Sau đó bạn tạo một thể nghiệm mới của lớp Adding, vì vậy bây giờ *ws* trở thành đối tượng *Adding*. Đối tượng *ws* này thực ra là một đối tượng proxy để biểu thị đối tượng thực trên server. Nó sẽ cho bạn một listing cảm ứng thông minh về tất cả các phương thức và các tham số giống như thể đây là một đối tượng cục bộ.

Sau đó bạn gọi phương thức AddTwoNumbers trên lớp Adding bằng cách chuyển vào hai giá trị từ hai hộp văn bản trên form. Bạn sẽ đặt kết quả trở lại từ phương thức này vào hộp văn bản txtResult.

1. Ấn **F5** để chạy ứng dụng phía client này.
2. Nhập số 1 vào hộp văn bản đầu tiên.
3. Nhập số 2 vào hộp văn bản thứ hai.
4. Nhấp nút Add. Có thể mất vài giây để gọi dịch vụ, vì vậy hãy kiên nhẫn.

Bây giờ bạn sẽ thấy kết quả bằng 3 trở lại hộp văn bản txtResult. Nó giống như hình 8.



Hình 8: *Đây là kết quả gọi dịch vụ Web từ một ứng dụng Windows.*

Điều tuyệt vời về kiểu dịch vụ này là người dùng không biết hoặc không quan tâm cách bạn đã đến kết quả cuối cùng như thế nào, họ chỉ quan tâm là bạn đã nhận được kết quả. Đây là một ưu điểm thú vị của các dịch vụ Web. Chúng có thể cư trú ở bất cứ đâu và bạn đều có thể gọi chúng. Thậm chí bạn không cần tự tạo các dịch vụ này mà bạn có thể gọi các dịch vụ của một người khác tạo.

Sử dụng dịch vụ Web từ một ứng dụng Web

Bạn có thể sử dụng các dịch vụ Web từ một ứng dụng Web giống như từ một ứng dụng Windows. Thực ra, các bước hoàn toàn giống nhau. Bạn sẽ phải tạo một Web form giống như client WinForms bạn vừa tạo.

1. Mở Visual Studio.NET.
2. Tạo một ứng dụng Web mới của ASP.NET.
3. Đặt tên nó là AddTwoNumbersClient.sln.

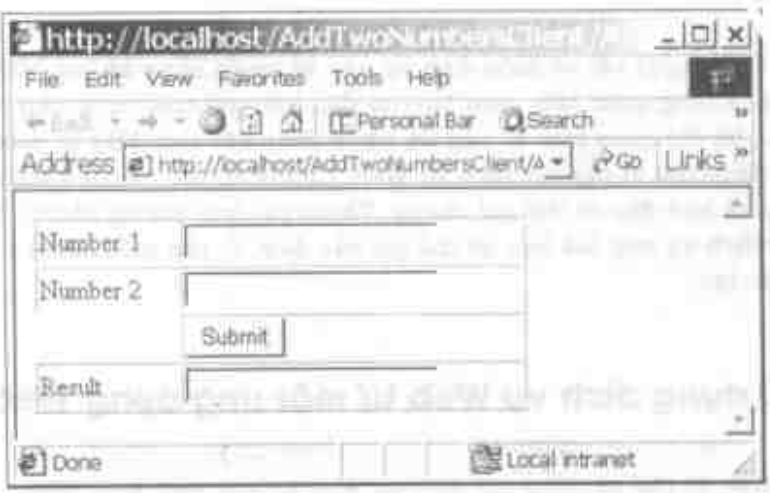
4. Đặt lại tên WebForm1.aspx là **AddTwoNumbers.aspx**.

5. Đổi Public Class WebForm1 thành **AddTwoNumbers**.

6. Tạo một màn hình như ở hình 9.

7. Đặt tên các hộp văn bản là **txtNum1**, **txtNum2** và **txtResult**.

8. Đặt tên nút là **btnSubmit**.



Hình 9: *Ứng dụng Web này có thể gọi Web Service giống như một ứng dụng Windows.*

1. Thêm một tham chiếu với dịch vụ Web bạn đã tạo.

2. Viết mã sau giống như bạn đã làm cho ứng dụng Windows.

```
Private Sub btnSubmit_Click( _
 ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles btnSubmit.Click
    Dim ws As localhost.Adding
```

```
    ws = New localhost.Adding()

    txtResult.Text = ws.AddTwoNumbers( _
     CInt(txtNum1.Text), _
     CInt(txtNum2.Text))
End Sub
```

Như bạn thấy điều này hầu như cũng giống mã bạn đã viết cho client Windows.

Dịch vụ Web Product

Bây giờ bạn sẽ tạo một dịch vụ Web để cho trở lại thông tin CSDL từ bảng Products. Bạn sẽ viết ba hàm trong dịch vụ Web mới này. Bạn sẽ viết một hàm để cho trở lại các đơn vị hàng kho cho ID sản phẩm đã cho. Bạn sẽ viết một phương thức để cho trở lại tất cả các cột trong bảng Products cho ID sản phẩm cá biệt. Bạn sẽ viết một phương thức để truy tìm tất cả các dòng và bốn cột từ bảng Products. Bạn sẽ có thể viết mỗi phương thức này khá dễ dàng vì nó chỉ là các thủ tục truy xuất CSDL chuẩn. Sự khác nhau duy nhất là bạn sẽ thêm bí danh <WebMethod()> vào trước mỗi phương thức này.

1. Tạo project Web Service mới và đặt tên nó là **ProductsWebService**.

2. Đặt lại tên file Service1.asmx là **Products.asmx**.

3. Đặt lại tên Public Class Service1 là **Products**.

4. Sau khi bạn tạo project mới này, bạn có thể tạo các phương thức sau.

Phương thức Units in Stock

Phương thức đầu tiên sẽ là UnitsInStock gọi hàm. Phương thức này sẽ nhận giá trị gọi *intProductID*. Tham số này sẽ được dùng trong câu lệnh SELECT để truy tìm số đơn vị hàng kho cho một sản phẩm cá biệt. Thêm phương thức sau vào lớp Products.

```
<WebMethod()> Public Function UnitsInStock( _
 ByVal intProductID As Integer) As String
    Dim oCmd As OleDb.OleDbCommand
    Dim oDR As OleDb.OleDbDataReader
    Dim strConn As String
    Dim strSQL As String
    Dim strUnits As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User ID=sa;"
    strConn &= "Password="

    strSQL = "SELECT UnitsInStock FROM Products "
    strSQL &= "WHERE ProductID = " & intProductID

    oCmd = New OleDb.OleDbCommand()
    With oCmd
        .Connection = New
OleDb.OleDbConnection(strConn)
        .Connection.Open()
        .CommandText = strSQL
        oDR = _

.ExecuteReader(CommandBehavior.SequentialAccess)
    End With

    If oDR.Read() Then
       strUnits = oDR.Item("UnitsInStock").ToString()
    End If
    oDR.Close()
    oCmd.Connection.Close()

    Return strUnits
End Function
```

Bạn nên kiểm tra phương thức mới này sau khi bạn đã tạo nó.

1. Ấn **F5** để chạy đối tượng.
2. Chọn siêu liên kết UnitsInStock.
3. Khi được nhắc về một giá trị, nhập ID sản phẩm là 1.
4. Nhấp nút Invoke.

Bạn sẽ thấy giá trị 39 được cho trở lại nếu bạn chưa sửa đổi dữ liệu bảng Products.

Phương thức truy tìm ProductInfo

Phương thức kế tiếp bạn sẽ viết sẽ truy tìm tất cả các cột cho một ID sản phẩm chỉ định mà bạn đã chuyển tới phương thức này và cho trở lại khách hàng DataSet. Viết phương thức sau trong lớp Products của bạn.

```
<WebMethod()> Public Function ProductInfo( _
 ByVal intProductID As Integer) As DataSet
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oDS As DataSet
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User ID=sa"

    strSQL = "SELECT * FROM Products "
    strSQL &= "WHERE ProductID = " & intProductID
```

```
    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oDS = New DataSet()

    oAdapter.Fill(oDS, "Products")

    Return oDS
End Function
```

Lưu ý, bạn có thể có giá trị trở lại từ hàm này là đối tượng DataSet. Nó là một trong các đối tượng có thể được nối tiếp hóa như một chuỗi XML. Thí dụ, bạn không có thể cho trở lại một DataTable, nhưng DataSet vẫn làm việc tốt.

Bạn nên kiểm tra phương thức mới này sau khi bạn đã tạo nó.

1. Ấn **F5** để chạy project.
2. Chọn siêu liên kết ProductInfo.
3. Khi được nhắc cho một giá trị, nhập ID sản phẩm là 1.
4. Nhấp nút Invoke.

Bạn sẽ thấy dữ liệu XML được xác lập trở lại là tất cả các cột cho một ID sản phẩm này.

Phương thức AllProducts

Phương thức cuối cùng bạn sẽ thêm vào lớp Products này sẽ cho trở lại tất cả các dòng và bốn cột từ bảng Products. Hãy tạo phương thức sau bên trong lớp Products.

```
<WebMethod()> Public Function AllProducts() As DataSet
    Dim oAdapter As OleDb.OleDbDataAdapter
    Dim oDS As DataSet
```

```
    Dim strSQL As String
    Dim strConn As String

    strConn = "Provider=sqloledb;"
    strConn &= "Data Source=(local);"
    strConn &= "Initial Catalog=Northwind;"
    strConn &= "User ID=sa"

    strSQL = "SELECT ProductID, ProductName, "
    strSQL &= "UnitPrice, UnitsInStock FROM Products "

    oAdapter = New OleDb.OleDbDataAdapter(strSQL,
strConn)
    oDS = New DataSet()

    oAdapter.Fill(oDS, "Products")

    Return oDS
End Function
```

Phương thức này cũng cho trở lại đối tượng DataSet, nhưng nó được điền với tất cả các dòng trong bảng Products và bốn cột là ProductID, ProductName, UnitPrice và UnitsInStock cho mỗi dòng.

Bạn nên kiểm tra phương thức mới này sau khi bạn đã tạo nó.

1. Ấn **F5** để chạy project.

2. Chọn siêu liên kết AllProducts.

3. Nhấp nút Invoke.

Bạn sẽ thấy dữ liệu XML xác lập cho trở lại bốn cột cho tất cả các dòng trong bảng Products.

Tóm tắt

Việc tạo các dịch vụ Web rất đơn giản. Bạn không cần tìm hiểu XML hoặc SOAP, bạn để .NET xử lý tất cả cho bạn. Chỉ cần xác lập Web Reference, sau đó bạn có thể sử dụng dịch vụ đó từ ứng dụng Windows hoặc ứng dụng Web.

Câu hỏi ôn tập

1. Bạn thêm bí danh nào vào trước tên phương thức để trình bày nó như một phương thức Web?
2. Đúng hay sai: Để kiểm tra dịch vụ Web bạn phải tạo một ứng dụng Windows hoặc ứng dụng Web để sử dụng dịch vụ này?
3. Các đuôi mở rộng nào của ba file được đưa vào khi bạn xác lập một tham chiếu Web?

Bài tập

- Tạo dịch vụ Web ở chương này.
- Tạo một ứng dụng Windows hoặc Windows Web và kiểm tra dịch vụ Web này.

Trả lời câu hỏi ôn tập

1. <WebMethod()>.
2. Sai.
3. .WSDL, .DISCO, .MAP.

LÊ MINH TRÍ

Microsoft DOT NET

Kỹ năng lập trình ứng dụng với Visual Basic .NET

Giá : 65.000đ