I. LỜI NÓI ĐẦU

Lập trình hệ thống trên nền tảng .Net FrameWork là một môi trường phát triển mạnh mẽ dựa trên các ngôn ngữ lập trình như Visual Bacsic.Net , Visual C#.Net.

Trong tài liệu này hướng dẫn chi tiết người học về cách tạo ra các ứng dụng dịch vụ chạy trên Server trong môi trường .Net Framework, giúp người học tự thao tác và lập trình ra các ứng dụng cơ bản của hệ thống như:

- Lập trình với windows server
- Lập trình với đối tượng .Net Remoting
- Lập trình với XML Web Service

Để tìm hiểu rõ hơn về các phần đề cập ở trên, người học phải lần lượt đi qua các nội dung của giáo trình từ bài học số 1 đến bài học số 3.

Bài học số 1: Giới thiệu đầy đủ các lý thuyết liên quan đến windows service và cũng trong bài học này người học sẽ được hướng dẫn thao tác để tạo ra các windows service trên 2 ngôn ngữ Visual C#.NET và Visual Basic.NET

Bài học số 2: Giới thiệu tới người học về kiến thức liên quan đến XML Web Service và hướng dẫn cách tạo ra các ứng dụng trên XML Web Service.

Bài học số 3: Giới thiệu tới người học các kiến thức liên quan đến .NET Remoting và vận dụng những kiến thức đó để tạo ra và triển khai các ứng dụng .Net Remoting theo trình tự thực hiện chi tiết.

Trong giáo trình sử dụng tham khảo các bài viết của giáo trình MCAD/MCSD, trong quá trình dịch các bài giảng có rất nhiều thuật ngữ không thể truyền tải sang tiếng việt, đó là các thuật ngữ kỹ thuật đặc trung của Công Nghệ Thông

Tin, chính thế vì trong quá trình biên soạn tác giả để nguyên bản một số cụm từ tiếng anh nhằm giữ nguyên ý nghĩa cũng như ý đồ của giáo trình.

Bài 1:Tạo và quản lý windows service

I. Mục tiêu

Học xong bài này người học sẽ đạt được những kỹ năng sau:

- Tạo được một windows service
- Quản lý ứng dụng windows service

Trước khi học yêu cầu:

-Phải quen thuộc với Microsoft. NET Framework và kiến trúc của nó

-Phải có kiến thức về lập trình trong Microsoft Visual Basic.NET hoặc Microsoft Visual C #

II. Nội dung

1. Tạo window service

1.1. Lý thuyết liên quan

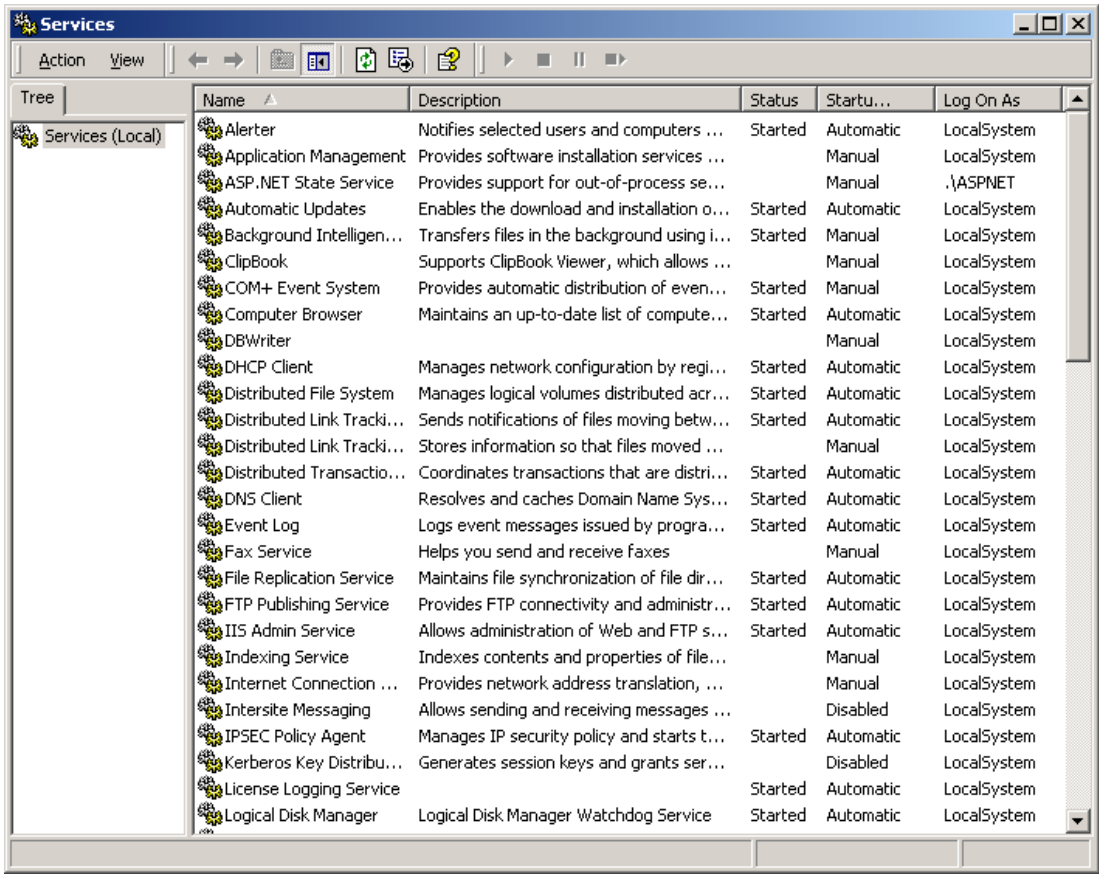
1.1.1Tổng quan về các dịch vụ của Windows

Các dịch vụ Windows chạy như một tiến trình nền tảng. Các ứng dụng này không có một giao diện người dùng, chúng làm những nhiệm vụ ngầm định mà không yêu cầu bất kỳ sự tương tác của người dùng. Bạn có thể cài đặt một dịch vụ Windows trên bất kỳ máy chủ hoặc máy tính đang chạy Windows 2000, Windows XP, hoặc Windows NT. Bạn cũng có thể chỉ định một dịch vụ Windows để chạy trong bối cảnh an ninh (chế độ bảo mật) của một tài khoản người dùng cụ thể hoặc chạy mặc định theo một tài khoản hiện hành của windows. Ví dụ, bạn có thể tạo ra một dịch vụ của Windows để giám sát hiệu suất truy cập dữ liệu và hoạt động của một cơ sở dữ liệu.

Bạn tạo ra các dịch vụ Windows để thực hiện các nhiệm vụ, chẳng hạn như quản lý các kết nối mạng và giám sát truy cập và sử dụng tài nguyên. Bạn cũng có thể sử dụng các dịch vụ Windows để thu thập và phân tích dữ liệu sử

dụng hệ thống và các sự kiện đăng nhập trong hệ thống hoặc ghi sự kiện tùy chỉnh. Bạn có thể xem danh sách các dịch vụ đang chạy trên một máy tính bất cứ lúc nào bằng cách mở Administrative Tools từ Control Panel, sau đó mở Services . Hình 2.1 hiển thị các cửa sổ Services.

Một dịch vụ Windows được cài đặt trong registry như một đối tượng thực thi. Service Control Manager (SCM) quản lý tất cả các dịch vụ Windows. Service Control Manager, đó là cuộc gọi thủ tục từ xa (RPC) máy chủ, hỗ trợ quản lý địa phương hoặc từ xa của một dịch vụ. Bạn có thể tạo ra các ứng dụng kiểm soát các dịch vụ Windows thông qua SCM bằng cách sử dụng Visual Studio. NET. NET Framework. Cung cấp các lớp cho phép bạn tạo ra, cài đặt, và kiểm soát các dịch vụ Windows một cách dễ dàng.



Kiến trúc dịch vụ Windows bao gồm ba thành phần:

- Dịch vụ ứng dụng :

Một ứng dụng bao gồm một hoặc nhiều dịch vụ cung cấp các chức năng mong muốn.

- Dịch vụ điều khiển ứng dụng:

Một ứng dụng cho phép bạn kiểm soát hành vi của một dịch vụ

- Kiểm soát dịch vụ quản lý:

Một tiện ích cho phép bạn kiểm soát các dịch vụ được cài đặt trên một máy tính.

Bạn có thể tạo ra các dịch vụ Windows bằng cách sử dụng bất kỳ ngôn ngữ NET, chẳng hạn như Visual C # hoặc Visual Basic. NET. Các không gian tên System.ServiceProcess của NET Framework. Chứa các lớp cho phép bạn tạo ra, cài đặt, thực hiện, và kiểm soát các dịch vụ Windows. Bạn sử dụng các phương thức của lớp ServiceBase để tạo ra một dịch vụ Windows. Sau khi bạn tạo một ứng dụng dịch vụ Windows, bạn cài đặt nó bằng cách đăng ký các ứng dụng trong registry của máy tính. Bạn sử dụng các lớp học ServiceInstaller và ServiceProcessInstaller để cài đặt các dịch vụ Windows. Bạn có thể xem tất cả các ứng dụng dịch vụ đăng ký trong registry của Windows dưới HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services. Sau khi bạn cài đặt một dịch vụ, bạn cần để bắt đầu nó. Bạn sử dụng lớp ServiceController và SCM để bắt đầu, dừng lại, tạm dừng, và tiếp tục một dịch vụ Windows. Các lớp ServiceController cũng cho phép bạn thực hiện các lệnh tùy chỉnh trên một dịch vụ.

Sau khi bạn bắt đầu một ứng dụng dịch vụ trên một máy tính, nó có thể tồn tại trong một chạy, tạm dừng hoặc dừng lại nhà nước. Ngoài ra, một dịch vụ có thể được trong trạng thái chờ. Một trạng thái chờ chỉ ra rằng một lệnh, chẳng hạn như các lệnh để tạm dừng một dịch vụ, đã được ban hành nhưng chưa hoàn tất.

Dịch vụ Windows được phân loại dựa trên số lượng các dịch vụ chạy trong một không gian của tiến trình (process). Bạn có thể có một dịch vụ đang chạy trong một không tiến trình hoặc nhiều dịch vụ chia sẻ một tiến trình không gian. Các dịch vụ mà sử dụng một tiến trình không gian duy nhất được gọi là dịch vụ Win32OwnProcess, trong khi các dịch vụ chia sẻ một tiến trình với các dịch vụ khác được gọi là Win32ShareProcess service.

Sự khác nhau giữa các ứng dụng windows service và Visual Studio.NET application.

Windows service khác nhau các ứng dụng Visual Studio.net khác.Các phần các đặc tính sau đây nói nên sự khác biệt.

Cài đặt Windows và dịch vụ

Không giống như các dự án Visual Studio khác, bạn cần phải cài đặt một ứng dụng dịch vụ trước khi dịch vụ đó có thể chạy. Để đăng ký và cài đặt dịch vụ, trước tiên bạn cần phải thêm các thành phần cài đặt một ứng dụng dịch vụ. Bạn sẽ học cách cài đặt một ứng dụng dịch vụ trong các bài sau.

Sửa lỗi

Bạn không thể gỡ lỗi một ứng dụng dịch vụ bằng cách nhấn F5 hoặc phím F11 như bạn vẫn làm với các ứng dụng application khác của Visual Studio.

NET. Để gỡ lỗi một ứng dụng dịch vụ, bạn cần cài đặt và bắt đầu dịch vụ, và sau đó đính kèm một debugger đến tiến trình của dịch vụ.

Thực hiện các dịch vụ của Windows

Trong một dịch vụ Windows, phương thức Run tải một dịch vụ vào SCM. Bạn gọi phương thức Run trong phương thức Main của một ứng dụng dịch vụ. Một sự khác biệt giữa thực hiện một dịch vụ Windows và ứng dụng Visual Studio.NET khác là các hộp thoại tùy chỉnh (*dialog boxes raised*) từ một dịch vụ Windows không hiển thị cho người dùng. Ngoài ra, các thông báo lỗi được tạo ra từ một dịch vụ Windows được đăng nhập trong các bản ghi sự kiện. Bạn có thể chỉ định các dịch vụ Windows chạy trong bối cảnh an ninh của chính họ. Vì vậy, một dịch vụ Windows có thể bắt đầu trước khi người dùng đăng nhập và tiếp tục chạy ngay cả sau khi người dùng đăng nhập.

Mô hình lập trình của các ứng dụng Window Services.

Chúng ta tạo các ứng dụng Window services bằng cách sử dụng các lớp của không gian tên (namespace) System.ServiceProcess. Chúng ta sử dụng các phương thức của lớp trong không gian tên này để cung cấp các chức năng tới ứng dụng dịch vụ của chúng ta.

Table 1-1. Some of the Classes of the System.ServiceProcess Namespace

Class	Description
ServiceBase	Bạn ghi đè lên các phương thức trong lớp ServiceBase để tạo ra một ứng dụng dịch vụ. Các phương thức của lớp này cho phép bạn xác định hành vi của các ứng dụng dịch vụ của bạn.
ServiceProcessInstaller và ServiceInstaller	Những lớp học này cho phép bạn xác định quá trình cài đặt và gỡ bỏ cài đặt ứng dụng dịch vụ

	của bạn
ServiceController	Các phương thức của lớp này cho phép bạn thao tác các (behavior) hành vi của ứng dụng dịch vụ của bạn tạo ra. Ví dụ, bạn có thể sử dụng các phương thức của lớp này để bắt đầu và dừng dịch vụ. Bạn sử dụng các phương thức của lớp này trong các ứng dụng mà bạn tạo ra để quản lý và kiểm soát một dịch vụ.

Ngoài các lớp của không gian tên System.ServiceProcess, chúng ta sử dụng các lớp của không gian tên System.Diagnostics để giám sát các trạng thái và gỡ lỗi dịch vụ khi nó đang chạy.

Table 1-2. Some of the Classes of the System.Diagnostics Namespace

Class	Description
EventLog	Các methods (phương thức) của lớp này cho phép các dịch vụ của bạn báo cáo lỗi trong bản ghi sự kiện(Event logs).
PerformanceCounter	Các methods(phương thức) của lớp này cho phép bạn sử dụng các counters (bộ đếm) hiệu suất , giám sát việc sử dụng tài nguyên.
Trace and Debug	Các phương thức của các lớp này cho phép bạn gỡ lỗi mã lập trình (code) của bạn và cung cấp cho bạn với các phương thức và properties (thuộc tính) giúp bạn theo dõi việc thực hiện của mã lệnh của bạn.

Để tạo một Window service, chúng ta tạo một lớp để mở rộng lớp System.ServiceProcess.ServiceBase. Chúng ta ghi đè các chức năng của lớp ServiceBase để tùy chỉnh hành vi ứng dụng dịch vụ của chúng ta.

Table 2-3. Methods of the ServiceBase Class

Sau khi bạn tạo một dịch vụ Windows bằng cách ghi đè các phương pháp của lớp ServiceBase, bạn sử dụng các lớp học ServiceProcessInstaller và ServiceInstaller để cài đặt các ứng dụng dịch vụ. Như đã đề cập trước đó, mỗi dịch vụ chạy trong quá trình không gian riêng của mình. Vì vậy, trước khi bạn cài đặt một ứng dụng dịch vụ, một quá trình dịch vụ cần phải được tạo ra và cài đặt. Lớp ServiceProcessInstaller tạo ra và đăng ký quá trình trong đó dịch vụ của bạn chạy, trong khi lớp ServiceInstaller cài đặt các lớp mở rộng lớp ServiceBase để thực hiện dịch vụ của bạn.

Class	Description
OnStart	Bạn ghi đè lên phương pháp này để xác định các nhiệm vụ dịch vụ của bạn thực hiện khi nó bắt đầu, chẳng hạn như khởi tạo các kết nối cơ sở dữ liệu, quầy thực hiện, bản ghi sự kiện, và các chủ đề con.
OnPause	Bạn ghi đè lên phương pháp này để xác định các nhiệm vụ dịch vụ của bạn thực hiện khi nó bắt đầu, chẳng hạn như khởi tạo các kết nối cơ sở dữ liệu, quầy thực hiện, bản ghi sự kiện, và các chủ đề con.
OnStop	Bạn ghi đè lên phương pháp này để xác định các nhiệm vụ dịch vụ của bạn thực hiện khi nó dừng lại. Ví dụ, bạn có thể giải phóng các tài nguyên hệ thống và kết nối cơ sở dữ liệu rằng các dịch vụ khởi tạo trong quá trình thực hiện của nó.
OnContinue	Bạn ghi đè lên phương pháp này để xác định dịch vụ của bạn cư xử như thế nào khi nó khởi động lại sau khi tạm dừng. Ví dụ, bạn có thể mở các kết nối cơ sở dữ liệu đã được đóng cửa trong phương pháp onPause.
OnShutDown	Bạn ghi đè lên phương pháp này để xác định nhiệm vụ mà dịch vụ của bạn thực hiện khi máy tính nó đang chạy trên bị đóng cửa.
OnCustomCommand	Bạn ghi đè lên phương pháp này khi bạn muốn có một dịch vụ để chấp nhận các lệnh tùy chỉnh. Một dịch vụ có thể đáp ứng một cách cụ thể tùy thuộc vào các đối số được thông qua với các lệnh tùy chỉnh.
OnPowerEvent	Bạn ghi đè lên phương pháp này để xác định dịch vụ của bạn thực hiện như thế nào khi nhận được một sự kiện quản lý điện năng như pin thấp.

Sau khi bạn tạo và cài đặt dịch vụ của bạn, bạn có thể sử dụng lớp ServiceController để lập trình điều khiển dịch vụ. Table. 2.4 mô tả các thức của lớp ServiceController mà bạn sử dụng để kiểm soát hành vi của một dịch vụ.

Table 2-4. Methods of the ServiceController Class

Class	Description
Close	Ngắt kết nối thể hiện của lớp ServiceController từ dịch vụ và phát hành tất cả các nguồn lực phân bổ cho các ứng dụng
Continue	Tiếp tục một dich vụ sau khi bạn đã tạm dừng nó
ExecuteCommand	Cho phép bạn thực thi một lệnh tùy chỉnh trên một dịch vụ
Pause	Dừng một dịch vụ đang hoạt động
Refresh	Refresh lại toàn bộ giá trị của các thuộc tính trong Service
Start	Khởi động một dịch vụ
Stop	Dừng dịch vụ và toàn bộ các ứng dụng dịch vụ phát triển theo.

1.1.2.Tạo các windows service

Tạo một dịch vụ ứng dụng Windows

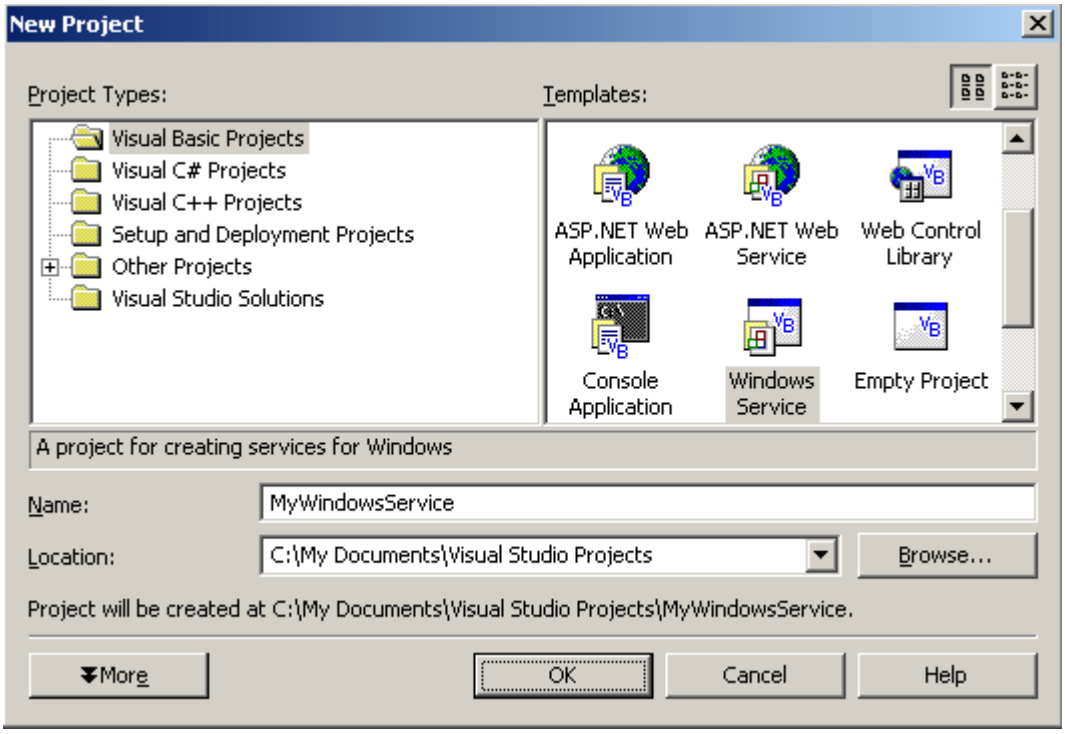
Để tạo ra một ứng dụng dịch vụ Windows, bạn cần phải tạo ra một dự án bằng cách sử dụng dịch vụ Windows mẫu trong Visual Basic NET hay Visual C #. Các Windows Service template tạo ra một ứng dụng dịch vụ trống cho bạn. Nó sẽ tự động thêm mã để phương thức **Main** của ứng dụng dịch vụ của bạn và chèn các **Funciton** cho các phương thức OnStart và OnStop. Để hoàn thành việc ứng dụng dịch vụ, bạn chỉ cần ghi đè lên các phương pháp OnStart và OnStop. Bạn cũng có thể ghi đè lên các phương thức khác của lớp ServiceBase, chẳng hạn như onPause và OnContinue, và thay đổi các thuộc tính của các ứng dụng dịch vụ tùy chỉnh nó theo ý của bạn. Để tạo ra một dịch vụ ứng dụng Windows trống, thực hiện các bước sau:

1)Khởi động Visual Studio .NET và tạo mới một project.

2)Chọn Visual Basic Projects or Visual C# Projects từ mục Project Types pane.

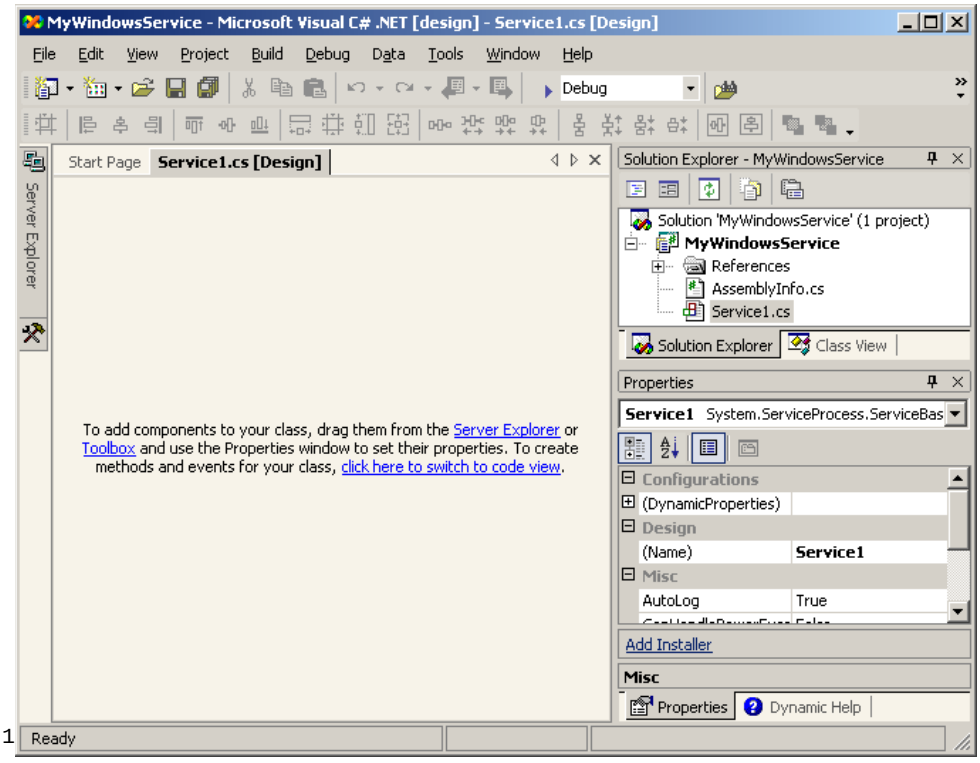
3)Chọn Windows Service template từ mục Templates pane trong New Project dialog box.

Hình 2.2 hiển thị một New Project dialog box.



4) Thay đổi tên của dự án để MyWindowsService trong hộp văn bản Name và nhấn OK. NET giải pháp mới MyWindowsSer[1]vice đặt tên được tạo ra.

Hình 2.3 hiển thị một dự án dịch vụ Windows.



1

Các mẫu dịch vụ Windows bao gồm các tài liệu tham khảo cho các không gian tên, chẳng hạn như không gian tên System.ServiceProcess, và cho biết thêm các tập tin nhất định trong dự án dịch vụ Windows, được yêu cầu để tạo ra dịch vụ của bạn. Các mẫu dịch vụ Windows cho biết thêm các tập tin sau đây cho dự án dịch vụ Windows của bạn:

Tập tin AssemblyInfo (AssemblyInfo.vb trong Visual Basic và AssemblyInfo.cs trong Visual C #).

Tập tin này mô tả assembly và chứa thông tin phiên bản.

Tập tin Service (Service1.vb trong Visual Basic và Service1.cs trong Visual C #).

Tập tin này có chứa mã cho ứng dụng dịch vụ của bạn. Điều này bao gồm các mã được tạo ra bởi các dịch vụ Windows mẫu và mã mà bạn thêm vào tùy chỉnh các ứng dụng dịch vụ.

Ngoài ra các tập tin tập tin và Dịch vụ AssemblyInfo, mẫu dịch vụ Windows cho biết thêm tài liệu tham khảo dưới đây không gian tên NET Framework, mà bạn sử dụng để tùy chỉnh các ứng dụng dịch vụ của bạn.

System.

Không gian tên này chứa các class định nghĩa các giá trị và tham chiếu các loại dữ liệu. Các Class của không gian tên này cũng định nghĩa các sự kiện và xử lý sự kiện, giao diện, và các thuộc tính. Ngoài ra, không gian tên này chứa các phương thức để xử lý các trường hợp ngoại lệ.

System.Data.

Không gian tên này định nghĩa kiến trúc ADO.NET và được sử dụng cho các nhiệm vụ liên quan đến cơ sở dữ liệu khác nhau. Bạn sẽ tìm hiểu về ADO.NET trong phần sau trong bài, "Cơ sở dữ liệu Lập trình Sử dụng ADO.NET."

System.ServiceProcess.

Các Class của không gian tên này cho phép bạn tạo , cài đặt, và chạy các dịch vụ Windows.

System.Xml.

Không gian tên này bao gồm các lớp học cung cấp hỗ trợ cho việc xử lý XML.

Bạn hãy chèn đoạn mã lệnh sau vào trong phương thứ Main của ứng dụng:

Visual Basic .NET

```
Shared Sub Main()
    Dim ServicesToRun() As System.ServiceProcess.ServiceBase
    ServicesToRun = New System.ServiceProcess.ServiceBase () {New Service1}
    System.ServiceProcess.ServiceBase.Run(ServicesToRun)
End Sub
```

Visual C#

```
static void Main()
{
    System.ServiceProcess.ServiceBase[] ServicesToRun;
    ServicesToRun = new System.ServiceProcess.ServiceBase[]
        { new Service1() };
    System.ServiceProcess.ServiceBase.Run(ServicesToRun);
}
```

Thay đổi các thuộc tính mặc định của một dịch vụ ứng dụng Windows

Sau khi bạn mở một dự án dịch vụ Windows mới, bạn cần phải xác định thuộc tính cần tùy chỉnh để ứng dụng dịch vụ theo yêu cầu của bạn. Trước tiên, bạn cần phải thay đổi các thuộc ServiceName và Name của ứng dụng dịch vụ. Bạn sử dụng thuộc tính ServiceName để chỉ định tên hiển thị trong SCM sử dụng để xác định dịch vụ Windows của bạn. Để thay đổi tên của các ứng dụng dịch vụ, hãy nhấp vào cửa sổ thiết kế và thay đổi thuộc ServiceName thành MyWindowsService_VB đối với ứng dụng viết bằng Visual Basic hoặc MyWindowsService_CSharp đối với ứng dụng viết bằng Visual C #) trong cửa sổ Properties. Bạn cũng cần phải thay đổi thuộc tính Name MyWindowsService_VB hoặc (MyWindowsService_CSharp trong Visual C #). Thuộc tính Name sẽ xác định tên của lớp mở rộng(extends) từ lớp System.ServiceProcess.ServiceBase trong ứng dụng dịch vụ của bạn. Sau khi bạn thay đổi thuộc tính Name và ServiceName, bạn cũng cần phải thay đổi tên dịch vụ trong các phương pháp chính. Các mã sau đây sẽ hiển thị mã thay đổi trong phương thức Main.

Visual Basic .NET

```
Shared Sub Main()
   Dim ServicesToRun() As System.ServiceProcess.ServiceBase
   ' Change the name of the instance of the service to match
   ' the ServiceName property
   ServicesToRun = New System.ServiceProcess.ServiceBase() _
      {New MyWindowsService_VB()}
   System.ServiceProcess.ServiceBase.Run(ServicesToRun)
End Sub
```

Visual C#

```
static void Main()
{
   System.ServiceProcess.ServiceBase[] ServicesToRun;
   /* Change the name of the instance of the service to match
      the ServiceName property */
   ServicesToRun = new System.ServiceProcess.ServiceBase[]
       { new MyWindowsService_CSharp() };
   System.ServiceProcess.ServiceBase.Run(ServicesToRun);
}
```

Bạn cần thay đổi các thuộc tính sau đây của ứng dụng dịch vụ:

AutoLog.

Thuộc tính này cho phép dịch vụ của bạn tạo ra một mục trong hệ thống Event logs để báo cáo các lỗi lệnh hoặc các thay đổi của trạng thái. Bạn có thể thiết lập thuộc tính AutoLog thành False nếu bạn muốn ứng dụng dịch vụ của bạn để sử dụng các bản ghi sự kiện tùy chỉnh. Bạn có thể log vào để sử dụng các mục được tạo ra trong phương thức OnContinue, onPause, hoặc phương thức OnStop của ứng dụng dịch vụ của bạn.

CanStop.

Thuộc tính này cho phép bạn chỉ định xem các dịch vụ của bạn có thể dừng lại sau khi nó đã được bắt đầu. Bạn có thể thiết lập tài sản CanStop False nếu bạn không muốn một người sử dụng để ngăn chặn các ứng dụng dịch vụ của bạn.

Nếu bạn thiết lập thuộc tính CanStop True, SCM gọi phương thức OnStop khi một yêu cầu dừng được gửi đến dịch vụ của bạn.

CanShutdown.

Bạn sử dụng thuộc tính để xác định xem các dịch vụ của bạn sẽ được thông báo khi hệ thống đang tắt. Nếu bạn thiết lập thuộc tính **CanShutdown** True, SCM thông báo dịch vụ của bạn khi hệ thống đang tắt. Điều này cho phép các dịch vụ của bạn gọi đến phương thức OnShutdown.

CanPauseAndContinue.

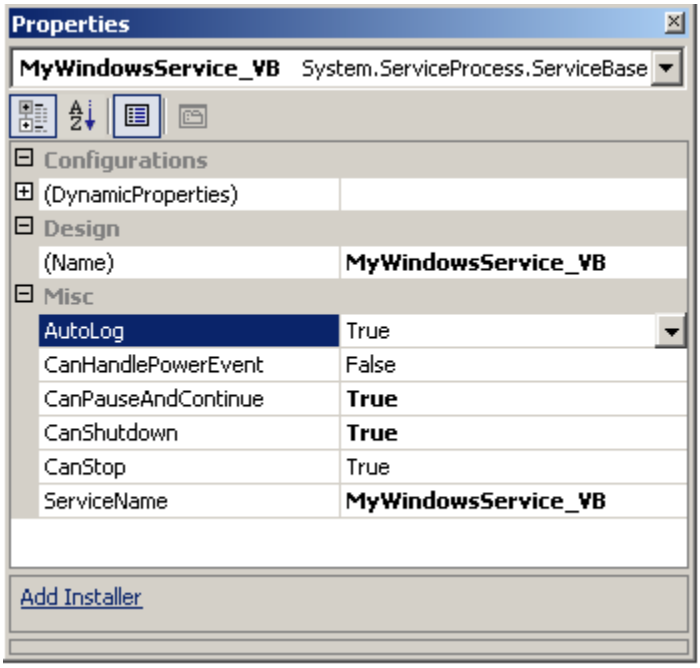
Bạn sử dụng thuộc tính để xác định xem người dùng có thể tạm dừng và tiếp tục dịch vụ của bạn. Nếu bạn thiết lập thuộc tính CanPauseAndContinue là True, bạn có thể ghi đè lên các phương thức onPause và OnContinue để xác định nhiệm vụ của dịch vụ cần thực hiện khi SCM gửi yêu cầu Pause và Continue tới dịch vụ của bạn. Nếu bạn thiết lập thuộc tính CanPauseAndContinue là False, SCM không gửi một yêu cầu Pause hoặc Continue tới dịch vụ của bạn. Vì vậy, các phương thức onPause và OnContinue không được gọi đến, ngay cả khi bạn đã định nghĩa chúng.

CanHandlePowerEvent.

Bạn sử dụng thuộc tính để xác định xem dịch vụ của bạn có thể xử lý các thay đổi trong trạng thái về nguồn điện máy tính, chẳng hạn như chế độ chờ hoặc máy laptopp đang hoạt động nhưng chuyển sang chế độ tiếc kiệm PIN. Thuộc tính này nhận giá trị kiểu Boolean.

Bạn có thể thay đổi các thuộc tính của một ứng dụng dịch vụ bằng cách sử dụng cửa sổ Properties của ứng dụng dịch vụ của bạn. Để truy cập vào cửa sổ Properties của ứng dụng dịch vụ, nhấp chuột phải vào cửa sổ thiết kế và chọn Properties. Các thuộc tính cho các ứng dụng dịch vụ của bạn xuất hiện.

Hình 2.4 hiển thị cửa sổ Properties của một ứng dụng dịch vụ.



Sau khi bạn thay đổi các thuộc tính của ứng dụng dịch vụ của bạn theo yêu cầu của bạn, bạn cần thay đổi tên của tập tin Service1.vb (Service1.cs đối với Visual C #). Để thay đổi tên của tập tin Service1.vb, nhấp chuột Service1.vb (Service1.cs đối với Visual C #) trong Solution Explorer, và sau đó thay đổi thuộc tính **File Name** MyWindowsService.vb (MyWindowsService.cs đối với Visual C #) trong cửa sổ Properties.

Thêm các hàm chức năng cho một dịch vụ ứng dụng

Để cung cấp các hàm cho dịch vụ của bạn, bạn ghi đè lên các phương thức OnStart và OnStop. Ví dụ, đoạn mã sau đây tạo ra một tập tin văn bản và ghi thông tin trong nó ngay trong phương thức OnStart.

```
Visual Basic .NET

Protected Overrides Sub OnStart(ByVal args() As String)

    Dim fs As New FileStream("C:\MyWindowsService_VB.txt", _
```

```
        FileMode.Append, FileAccess.Write)
    Dim sr As New StreamWriter(fs)
    sr.WriteLine("MyWindowsService_VB started")
    sr.Flush()
End Sub
```

Visual C#

```
protected override void OnStart(string[] args)
{
    FileStream fs = new FileStream(@"c:\MyWindowsService_CS.txt",
        FileMode.OpenOrCreate, FileAccess.Write);
    StreamWriter sr = new StreamWriter(fs);
    sr.WriteLine("MyWindowsService_CS started");
    sr.Flush();
}
```

Lưu ý :

Để tạo ra một tập tin, sử dụng các phương thức và các thuộc tính của lớp FileStream. Lớp FileStream được định nghĩa trong không gian tên System.IO. Vì vậy, bạn cần phải thêm một tham chiếu đến không gian tên System.IO để tạo ra một tập tin.

```
using System.IO
```

Visual Basic .NET

```
Protected Overrides Sub OnStop()
    Dim fs As New FileStream("C:\Temp\MyWindowsService_VB.txt", _
        FileMode.Append, FileAccess.Write)
    Dim sr As New StreamWriter(fs)
    sr.WriteLine("MyWindowsService_VB stopped")
    sr.Flush()
    sr.Close()
End Sub
```

Visual C#

```
protected override void OnStop()
{
    FileStream fs = new FileStream("C:\\Temp\\MyWindowsService_CS.txt",
        FileMode.OpenOrCreate, FileAccess.Write);
    StreamWriter sr = new StreamWriter(fs);
    sr.WriteLine("MyWindowsService_CS Stopped");
    sr.Flush();
    sr.Close();
}
```

Ngoài các phương thức OnStart và OnStop, bạn có thể định nghĩa phương thức OnCustomCommand để làm tăng chức năng của dịch vụ của bạn. Bạn sẽ được học cách để định nghĩa và sử dụng phương thức này trong các bài tới. Bên

cạnh các mã mà bạn viết, chương trình ***visual studio*** tự động viết đoạn code sau đây cho các ứng dụng dịch vụ của bạn.

Visual Basic .NET

```
Imports System.ServiceProcess
Public Class MyWindowsService_VB
    Inherits System.ServiceProcess.ServiceBase
#Region " Component Designer generated code "
    Public Sub New()
        MyBase.New()
        ' This call is required by the Component Designer.
        InitializeComponent()
        ' Add any initialization after the InitializeComponent() call
    End Sub
    ' UserService overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub
```

```
' The main entry point for the process
<MTAThread()> _
Shared Sub Main()
    Dim ServicesToRun() As System.ServiceProcess.ServiceBase


    ' More than one NT service may run within the same process. To add
    ' another service to this process, change the following line to
    ' create a second service object. For example,
    '
    ' ServicesToRun = New System.ServiceProcess.ServiceBase () {New
    '   Service1, New MySecondUserService}
    '
    ServicesToRun = New System.ServiceProcess.ServiceBase() {New _
        MyWindowsService_VB()}
    System.ServiceProcess.ServiceBase.Run(ServicesToRun)
End Sub
' Required by the Component Designer
Private components As System.ComponentModel.IContainer
' NOTE: The following procedure is required by the Component Designer
' It can be modified using the Component Designer.
' Do not modify it using the code editor.
```

```
    <System.Diagnostics.DebuggerStepThrough()> Private Sub _
        InitializeComponent()
        '
        ' MyWindowsService_VB
        '
        Me.ServiceName = "MyWindowsService_VB"
    End Sub
#End Region
Protected Overrides Sub OnStart(ByVal args() As String)
                Dim FS As New FileStream("C:\MyWindowsService_VB.txt",
FileMode.Append, _
        FileAccess.Write)
    Dim SR As New StreamWriter(FS)
    SR.WriteLine("MyWindowsService_VB started")
    SR.Flush()
End Sub
Protected Overrides Sub OnStop()
    Dim FS As New FileStream("C:\Temp\MyWindowsService_VB.txt", _
        FileMode.Append, FileAccess.Write)
    Dim SR As New StreamWriter(FS)
    SR.WriteLine("MyWindowsService_VB stopped")
    SR.Flush()
```

```
    SR.Close()
End Sub
End Class
```

Visual C#

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.ServiceProcess;
namespace MyWindowsService_CSharp
{
    public class MyWindowsService_CSharp : System.ServiceProcess.ServiceBase
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        public MyWindowsService_CSharp()
        {
            // This call is required by the Windows.Forms Component Designer.
```

```
    InitializeComponent();


    // TODO: Add any initialization after the InitComponent call
}
// The main entry point for the process
static void Main()
{
    System.ServiceProcess.ServiceBase[] ServicesToRun;
    // More than one user service may run within the same process.
    // To add another service to this process,
    // change the following line to
    // create a second service object. For example,
    //
    //    ServicesToRun = New System.ServiceProcess.ServiceBase[]
    //        {new Service1(), new MySecondUserService()};
    //
    ServicesToRun = new System.ServiceProcess.ServiceBase[] {
        new MyWindowsService_CSharp() };
    System.ServiceProcess.ServiceBase.Run(ServicesToRun);
}
/// <summary>
```

```
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    //
    // MyWindowsService_CSharp
    //
    this.ServiceName = "MyWindowsService_CSharp";
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
```

```
    }
    base.Dispose( disposing );
}
/// <summary>
/// Set things in motion so your service can do its work.
/// </summary>
protected override void OnStart(string[] args)
{
    FileStream fs = new FileStream(@"c:\MyWindowsService_CS.txt" ,
      FileMode.OpenOrCreate, FileAccess.Write);
    StreamWriter SR = new StreamWriter(fs);
    SR.WriteLine("MyWindowsService_CS started");
    SR.Flush();
}
/// <summary>
/// Stop this service.
/// </summary>
protected override void OnStop()
{
    FileStream fs = new FileStream(
        @"C:\Temp\MyWindowsService_CS.txt", FileMode.OpenOrCreate,
```

```
            FileAccess.Write);
        StreamWriter SR = new StreamWriter(fs);
        SR.WriteLine("MyWindowsService_CS Stopped");
        SR.Flush();
        SR.Close();
      }
   }
}
```

Bài trên hoàn thành việc tạo ra các cấu trúc cơ bản của dịch vụ Windows của bạn. Sau khi bạn tạo ra các cấu trúc của dịch vụ Windows của bạn, bạn có thể tiếp tục tăng cường các chức năng của ứng dụng của bạn bằng cách cho phép nó để xử lý các sự kiện khác nhau và các thông tin báo cáo. Phần tiếp theo sẽ trình bày về xử lý các sự kiện và thông tin đăng nhập từ một dịch vụ Windows.

1.2.3.Điều khiển các sự kiện và thông tin đăng nhập từ các ứng dụng dịch vụ Windows

Xử lý sự kiện của một dịch vụ Windows

Giống như bất kỳ ứng dụng Windows khác, một dịch vụ Windows hỗ trợ các sự kiện. Các sự kiện của một dịch vụ Windows phụ thuộc vào trạng thái của dịch vụ. Khi trạng thái của một dịch vụ thay đổi, sự kiện tương ứng xảy ra. Ví dụ, khi một dịch vụ bị tạm dừng, thay đổi trạng thái của nó chạy bị tạm dừng, và sự kiện này Tạm dừng xảy ra. Một dịch vụ Windows hỗ trợ bốn sự kiện: Start, Stop, Pause, và Continue.

Start Event

Sự kiện này bắt đầu xảy ra khi bạn bắt đầu một dịch vụ bằng cách sử dụng SCM. Khi sự kiện bắt đầu xảy ra, hệ thống tìm các tập tin .exe đích của sự kiện đó và gọi phương thức OnStart cho dịch vụ này. Bạn sử dụng phương pháp OnStart để xử lý các sự kiện bắt đầu. Bạn chỉ rõ những nhiệm vụ mà bạn muốn dịch vụ của bạn để thực hiện khi nó bắt đầu.

Bạn có thể đặt dịch vụ của bạn để bắt đầu khi máy tính được c khởi động lại. Để làm điều này, sử dụng thuộc tính StartType của dịch vụ. Bạn có thể xác định rằng dịch vụ của bạn bắt đầu tự động, khi khởi động máy tính, hoặc bằng tay bởi người dùng thiết đặt. Bạn cũng có thể vô hiệu hóa một dịch vụ để người dùng không thể bắt đầu nó. Bạn sử dụng các giá trị kiểu liệt kê của ServiceStartMode để xác định liệu một dịch vụ tự động bắt đầu khi máy tính khởi động, bắt đầu khi người sử dụng một cách thủ công bắt đầu nó, hoặc bị vô hiệu hóa. ServiceStartMode cung cấp các tính năng Automatic, Manual, and Disabled.

Stop Event

Khi bạn ngừng một dịch vụ bằng cách sử dụng SCM, sự kiện Stop xảy ra, và được gọi tới phương thức OnStop cho các ứng dụng dịch vụ. Bạn sử dụng phương thức OnStop để xử lý các sự kiện Stop bằng cách xác định các nhiệm vụ mà bạn muốn dịch vụ của bạn để thực hiện khi nó dừng lại. Khi xảy ra sự kiện Stop, SCM kiểm tra giá trị của thuộc tính CanStop trong dịch vụ của bạn. Nếu thuộc tính CanStop được thiết lập là True, SCM thông qua lệnh Stop của dịch vụ và gọi phương thức OnStop. Nếu bạn không xác định phương thức OnStop cho các ứng dụng dịch vụ của bạn, SCM xử lý lệnh Stop. Nếu giá trị của tài sản CanStop được thiết lập để False, SCM không thông qua các lệnh Stop để dừng ứng dụng dịch vụ. Ngoài ra, khi bạn dừng lại một dịch vụ, tất cả các dịch vụ phụ thuộc cũng dừng theo.

Với sự kiên Start và sự kiện Stop, bạn cũng có thể viết mã để xử lý các sự kiên Pause và Continue.

Pause Event

Bạn sử dụng phương pháp onPause để xử lý các sự kiện Tạm dừng bằng cách xác định các nhiệm vụ mà bạn muốn dịch vụ của bạn để thực hiện khi nó bị tạm dừng. Khi dịch vụ bị tạm dừng, SCM kiểm tra giá trị của thuộc tính CanPauseAndContinue. Nếu thuộc tính CanPauseAndContinue được thiết lập là True, SCM sẽ gửi một yêu cầu tạm dừng dịch vụ của bạn và gọi phương thức onPause. Bạn sử dụng phương thức onPause để bảo tồn các tài nguyên hệ thống được sử dụng khi các dịch vụ đang chạylại. Ví dụ, nếu bạn khai báo các biến cố định khi dịch vụ bắt đầu, tạm dừng một dịch vụ cho phép các biến vẫn còn trong bộ nhớ.

Continue Event

Bạn sử dụng phương thưc OnContinue thực hiện nhiệm vụ khi bạn hoạt động lại hoặc để ngăn chặn các nhiệm vụ mà bạn bắt đầu khi dịch vụ đã bị tạm dừng. Như với sự kiện Pause, SCM kiểm tra giá trị của thuộc tính CanPauseAndContinue khi một dịch vụ được phục hồi lại từ trạng thái tạm dừng. Nếu thuộc tính CanPauseAndContinue được thiết lập là True, SCM thông qua lệnh Continue với dịch vụ và gọi phương thức OnContinue.

Đăng nhập thông tin trong System Event Logs

Bạn có thể sử dụng tính năng đăng nhập sự kiện để đăng nhập các sự kiện phần mềm và phần cứng, chẳng hạn như sự thất bại của một dịch vụ khi khởi động, điều kiện bộ nhớ thấp, hoặc tạm dừng một dịch vụ. Các bản ghi sự kiện sau đó có thể giúp bạn xác định loại và nguyên nhân của lỗi. Bạn sử dụng các thành

phần EventLog để truy cập vào các bản ghi sự kiện trên cả hai máy tính địa phương và từ mày tình từ xa và viết các bản ghi. Ngoài ra, bạn có thể sử dụng Server Explorer để xem một danh sách các sự kiện đăng nhập mà bạn có thể truy cập. Có ba loại bản ghi sự kiện có sẵn theo mặc định mà bạn có thể đăng nhập các mục.

System Log.

Chứa các thông báo liên quan đến các sự kiện xảy ra trên các thành phần hệ thống như trình điều khiển thiết bị.

Security Log.

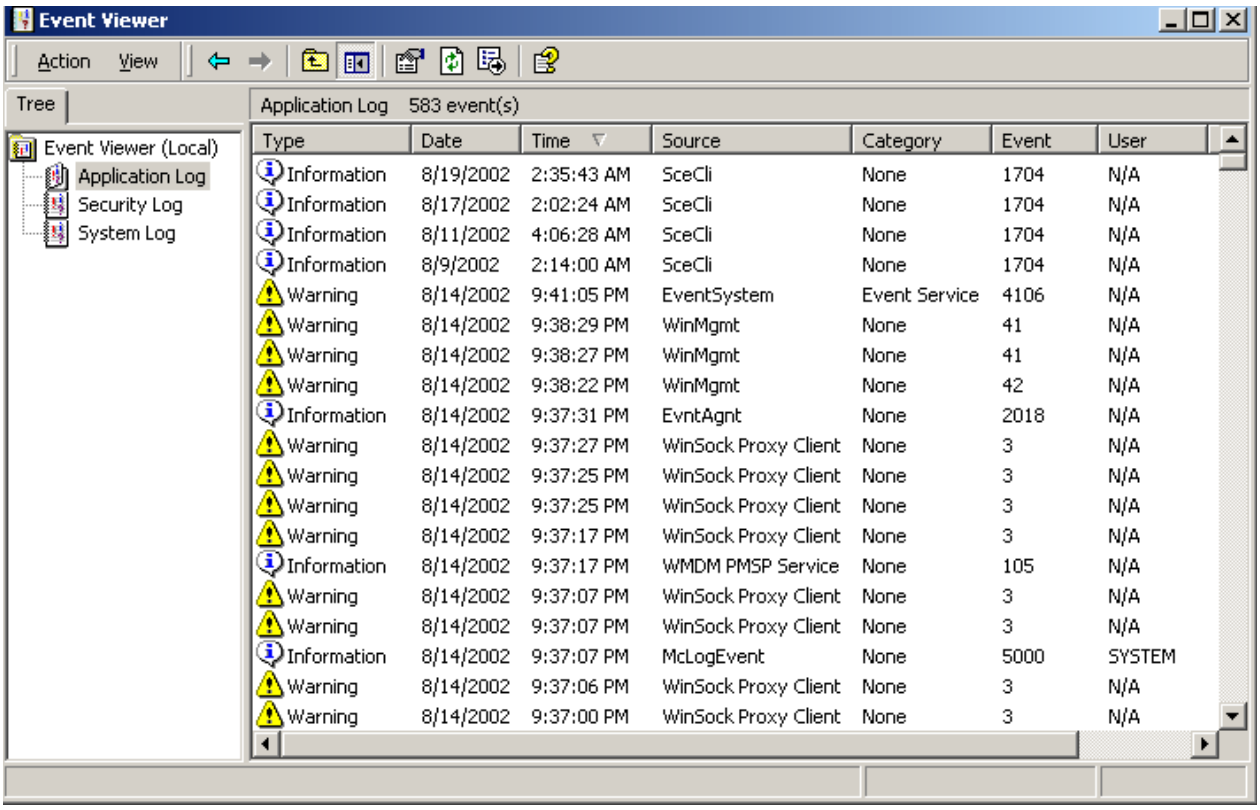
Chứa các thông báo liên quan đến thay đổi trạng thái bảo mật.

Application Log.

Chứa các thông báo liên quan đến các sự kiện xảy ra trong các ứng dụng được cài đặt trên máy tính.

Những Event logs sẽ được hiển thị tất cả trong các mục như thông báo lỗi, cảnh báo, thông tin, kiểm soát hoạt động thành công, hoặc kiểm soát các hoạt động thất bại. Bạn có thể xem các mục của các bản ghi sự kiện khác nhau trong cửa sổ Event Viewer.

Hình 2.5 hiển thị cửa sổ Event Viewer.



Bạn có thể chỉ định một ứng dụng dịch vụ để truy cập vào các bản ghi sự kiện mặc định để đăng nhập các thông tin như lỗi, trường hợp ngoại lệ, và những thay đổi trong trạng thái của các ứng dụng dịch vụ. Để kích hoạt ứng dụng dịch vụ của bạn để truy cập vào các bản ghi sự kiện hệ thống và tạo ra các mục trong đó, thiết lập thuộc tính AutoLog của một ứng dụng dịch vụ là True.

Khi bạn cài đặt ứng dụng dịch vụ của bạn, trình cài đặt cho ứng dụng dịch vụ của bạn kiểm tra giá trị thuộc tính AutoLog. Nếu thuộc tính AutoLog được thiết lập là True, trình cài đặt của ứng dụng dịch vụ của bạn đăng ký dịch vụ của bạn trong Nhật ký ứng dụng(Application log) của máy tính như là một nguồn của các sự kiện. Các dịch vụ có thể sau đó tự động đăng nhập thông tin mỗi khi dịch vụ started, stopped, paused, resumed, installed, or uninstalled. Ngoài

ra, các dịch vụ có thể ghi các ngoại lệ, các thất bại, và các lỗi. Các mã sau đây cho thấy làm thế nào để đăng nhập các mục vào các bản ghi sự kiện mặc định.

Visual Basic .NET

```
Protected Overrides Sub OnStart(ByVal args() As String)
    EventLog.WriteEntry("MyWindowsService Started")
End Sub

Protected Overrides Sub OnStop()
    EventLog.WriteEntry("MyWindowsService Stopped")
End Sub
```

Visual C#

```
protected override void OnStart(string[] args)
{
    // TODO: Add code here to start your service.
    EventLog.WriteEntry("Starting MyWindowsServices");
}

protected override void OnStop()
{
    // TODO: Add code here to perform any tear-down
    // necessary to stop your service.
    EventLog.WriteEntry("Stopping MyWindowsServices");
```

```
}
```

Tạo Custom Event Logs

Ngoài các bản ghi sự kiện hệ thống, Studio Visual.NET cho phép bạn tạo ra các bản ghi sự kiện tùy chỉnh để đăng nhập các sự kiện dành riêng cho các ứng dụng của bạn. Ví dụ, nếu bạn cần phải lưu tất cả các mục đăng nhập có liên quan đến dịch vụ của bạn, bạn có thể tạo ra một bản ghi sự kiện tùy chỉnh. Điều này cho phép bạn lưu trữ tất cả các mục đăng nhập liên quan đến dịch vụ của bạn ở một nơi. Ngoài ra, các mục đăng nhập không bị mất nếu các bản ghi mặc định sẽ bị xóa.

Bạn sử dụng phương thức CreateEventSource của lớp EventLog để tạo ra một bản ghi sự kiện tùy chỉnh. Phương thức này tạo ra một nguồn (source) và cho phép bạn xác định các bản ghi bên trong đó mà dịch vụ của bạn sẽ ghi vào. Để sử dụng phương thức CreateEventSource, thêm một tham chiếu đến không gian tên System.Diagnostics.

Để tạo một bản ghi sự kiện tùy chỉnh, thực hiện các bước sau:

1)Thiết lập thuộc tính AutoLog của ứng dụng dịch vụ của bạn là False.

2)Thêm một thể hiện của các thành phần EventLog từ hộp công cụ các thành phần (Components toolbox)trong ứng dụng dịch vụ của bạn.

3)Chỉ định một giá trị cho thuộc tính Source và tên của file bản ghi mà bạn muốn tạo ra trong các thuộc tính Log.

Bạn cũng có thể tạo ra một bản ghi sự kiện tùy chỉnh cho chương trình. Các mã sau đây cho thấy làm thế nào để tạo ra một bản ghi sự kiện tùy chỉnh chương trình.

Visual Basic .NET

```
Private EventLog1 as New System.Diagnostics.EventLog

.

.

.

Sub CreateEventLog()

    ' Initialize the instance of the EventLog component



    ' Create an event log for your service.

    ' The code also checks if an event log already exists

    ' for your service application

    If Not EventLog1.SourceExists("Transaction Service") Then

        EventLog1.CreateEventSource("Transaction Service", "Transaction Log")

    End If



    ' Specify the event log to use your service application as source

    EventLog1.Source = " Transaction Service"

End Sub
```

Visual C#

```
private System.Diagnostics.EventLog eventLog1;
```

```
.
.
.

private void CreateEventLog()
{
   //Initialize the instance of the EventLog component
   eventLog1 = new System.Diagnostics.EventLog();

   /* Create an event log for your service. The code also checks if an
      event log already exists for your service application */
   if (!System.Diagnostics.EventLog.SourceExists("Transaction Service"))
   {
      System.Diagnostics.EventLog.CreateEventSource("Transaction Service",
         "Transaction Log");
   }
   //Specify the event log to use your service application as source
   eventLog1.Source = "Transaction Service";
}
```

Bạn có thể gọi phương thức CreateEventLog() trong phương thức OnStart. Sau khi bạn tạo một bản ghi sự kiện tùy chỉnh, bạn có thể dùng dịch vụ

của bạn để tạo trực tiếp ra các mục trong đó. Các mã sau đây cho thấy làm thế nào để ghit trong một bản ghi sự kiện tùy chỉnh.

Visual Basic .NET

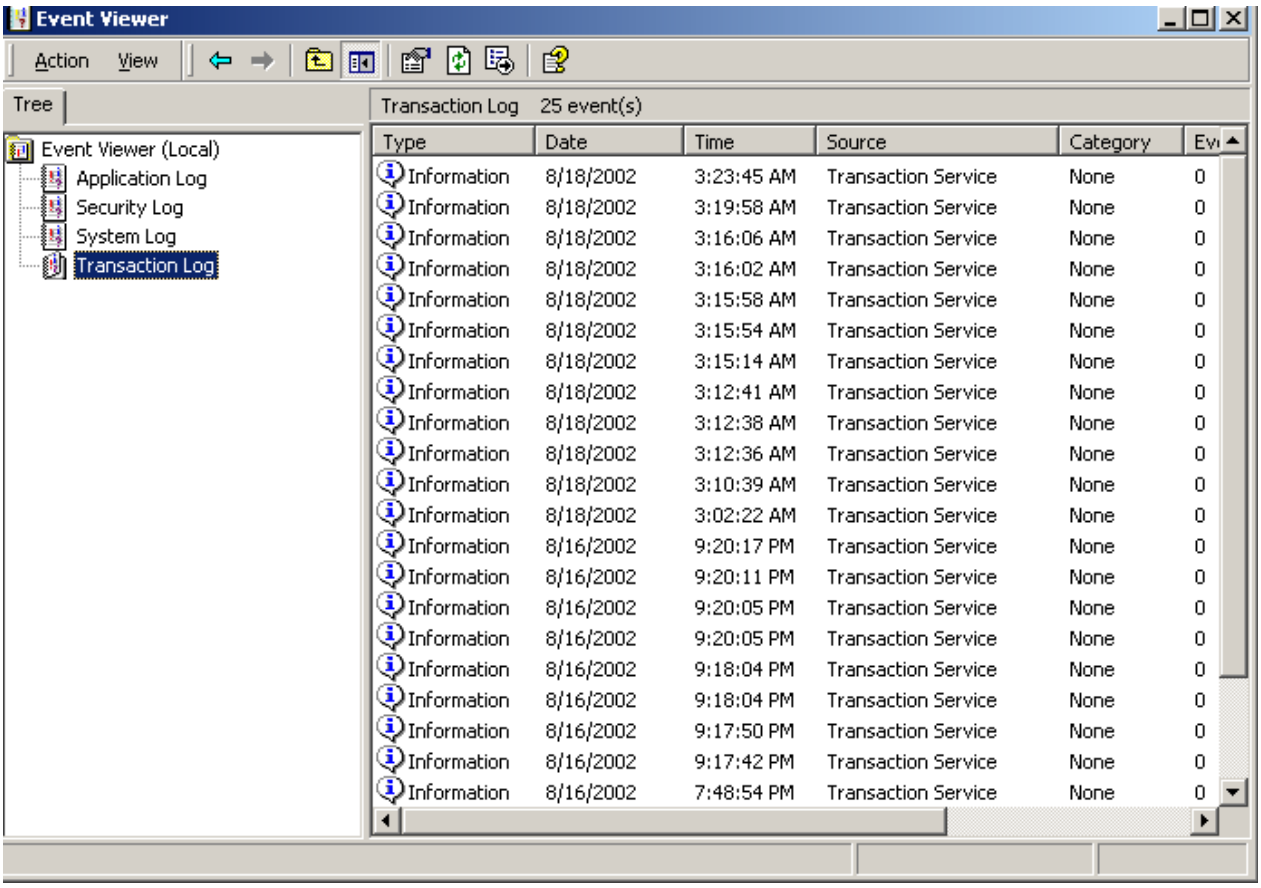
```
Protected Overrides Sub OnStart(ByVal args() As String)
   CreateEventLog()
   EventLog1.WriteEntry("MyWindowsService_VB Started", _
      EventLogEntryType.Information)
End Sub
```

Visual C#

```
protected override void OnStart(string[] args)
{
   CreateEventLog();
   eventLog1.WriteEntry("MyWindowsService_CSharp Started",
      EventLogEntryType.Information);
}
```

Sau đó, bạn có thể sử dụng cửa sổ Event Viewer để xem các mục trong bản ghi sự kiện tùy chỉnh mà bạn đã tạo.

Hình 2.6 hiển thị các mục trong một bản ghi sự kiện tùy chỉnh



Sử dụng các bộ đếm hiệu suất (Performance Counters)

Performance Counters cho phép bạn giám sát hiệu suất của các ứng dụng của bạn và giúp bạn để xác định và loại bỏ performance bottlenecks (hiện tượng thắt cổ chai – Hiện tượng này là do tài nguyên ứng dụng chiếm quá nhiều CPU).

Bạn có thể kích hoạt các ứng dụng của bạn để public các dữ liệu liên quan đến hoạt động bằng cách sử dụng các bộ đếm hiệu suất. Ví dụ, bạn có thể kích hoạt một ứng dụng để public các dữ liệu về việc sử dụng bộ nhớ, số lượng các kết nối cơ sở dữ liệu đang hoạt động, hoặc số các thành phần đang hoạt động trong một tiến trình (process) . Bạn có thể thu thập dữ liệu hiệu suất

mà bạn muốn từ ứng dụng của bạn để public ra bên ngoài và phân tích nó để tinh chỉnh ứng dụng của bạn cho phù hợp.

NET Framework. Cung cấp các lớp PerformanceCounter trong không gian tên System.Diagnostics. Lớp này cho phép bạn xuất bản(publish), thu thập(collect), và phân tích (analyze) các dữ liệu hiệu suất của một ứng dụng. Lớp PerformanceCounter cũng cho phép bạn sử dụng các bộ đếm hiệu suất hiện có trong ứng dụng của bạn. Ngoài ra, nó cho phép bạn sử dụng Performance Conuter tùy chỉnh mà bạn tạo cho các ứng dụng của bạn.

Để sử dụng các bộ đếm hiệu suất hiện có trong ứng dụng dịch vụ của bạn, hoàn thành các bước sau:

1) Mở Windows service của bạn trong chế độ Design view

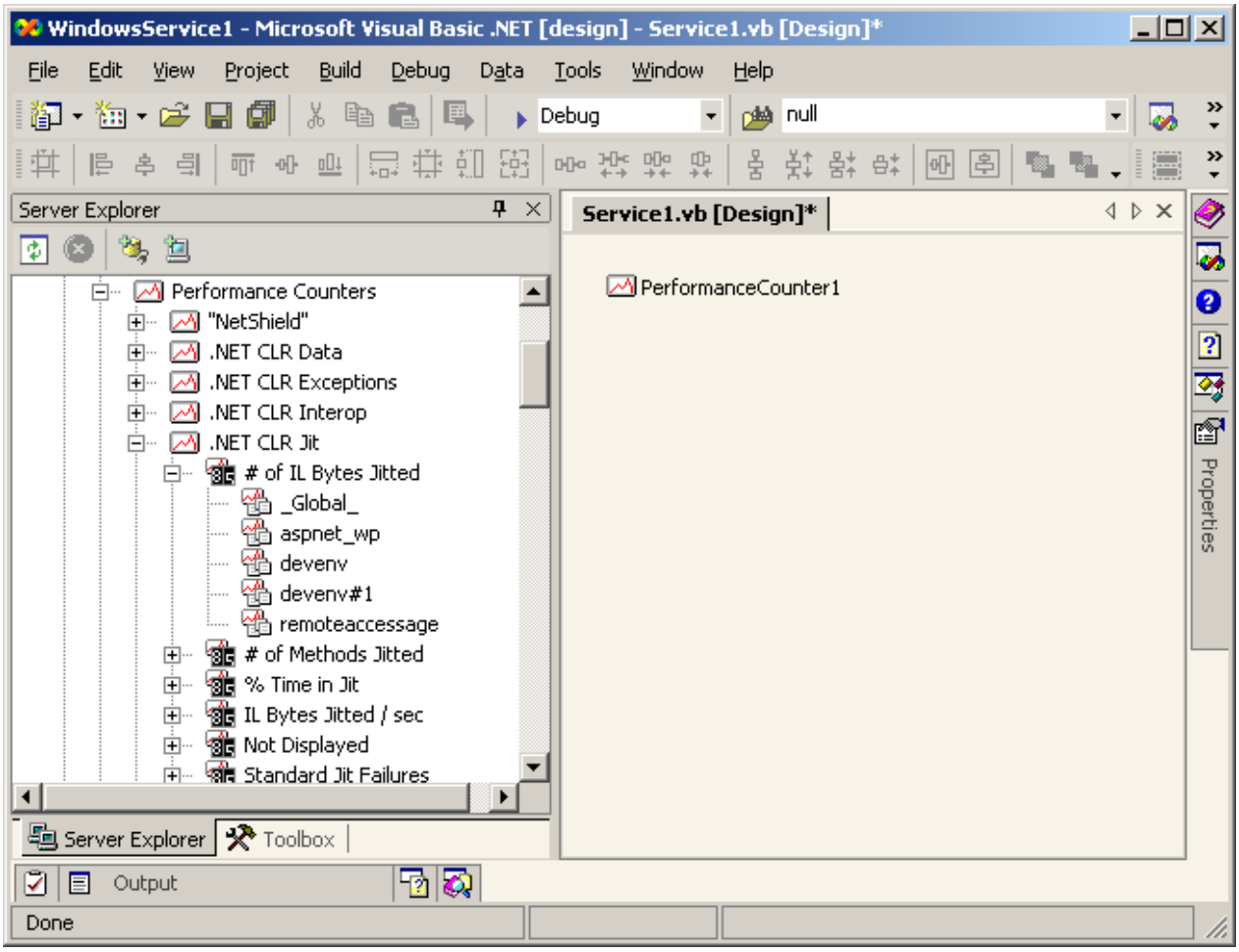
2) Mở cửa sổ Server Explorer (vào menu View/ Server Explorer)

3) Mở rộng nút Server (nhấn vào dáu +), Xác định tên máy chủ của bạn, click vào đó, vào mở rộng nút PerformanceCounter (nhấn vào dấu +)

4) Chọn một danh mục PerformanceCounter truy cập và mở rộng nó.

5)Xác định vị trí các PerformanceCounter mà bạn muốn sử dụng. Ví dụ, bạn có thể sử dụng ***Global # Of IL Bytes Jitted counter*** truy cập từ ***.NET CLR Jit category***, trong đó quy định cụ thể số lượng các byte JIT biên dịch.

6)Kéo và thả các Counter vào cửa sổ thiết kế như được hiển thị ở đây.



Visual Studio NET cho thêm các mã sau vào lớp dịch vụ của bạn để tạo ra một thể hiện của Performance Counter mà bạn thêm vào ứng dụng dịch vụ của bạn.

Visual Basic .NET

```
Friend WithEvents PerformanceCounter1 As
    System.Diagnostics.PerformanceCounter
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()


    Me.PerformanceCounter1 = New System.Diagnostics.PerformanceCounter()
```

```
    CType(Me.PerformanceCounter1, _
        System.ComponentModel.ISupportInitialize).BeginInit()
    Me.PerformanceCounter1.CategoryName = ".NET CLR Jit"
    Me.PerformanceCounter1.CounterName = "Total # of IL Bytes Jitted"
    Me.PerformanceCounter1.InstanceName = "_Global_"
End Sub
```

Visual C#

```
private System.Diagnostics.PerformanceCounter performanceCounter1;


private void InitializeComponent()
{
    this.performanceCounter1 = new System.Diagnostics.PerformanceCounter();
    ((System.ComponentModel.ISupportInitialize)
        (this.performanceCounter1)).BeginInit();
    this.performanceCounter1.CategoryName = ".NET CLR Jit";
    this.performanceCounter1.CounterName = "Total # of IL Bytes Jitted";
    this.performanceCounter1.InstanceName = "_Global_";
}
```

Lưu ý:

Theo mặc định thiết kế thêm thuộc tính MachineName, xác định tên máy mà truy cập được cài đặt. Để thực hiện các cài performanceCounter, bạn có thể thiết lập thuộc tính MachineName của performanceCounter để trống trong cửa sổ Properties. Khi bạn thiết lập thuộc tính MachineName của performanceCounter, chườn trình không tự động thêm mã để thiết lập thuộc tính MachineName.

Bạn cũng có thể tạo ra Custom Performance Counters tuỳ chỉnh. Để tạo một Custom Performance Counters, hoàn thành các bước sau:

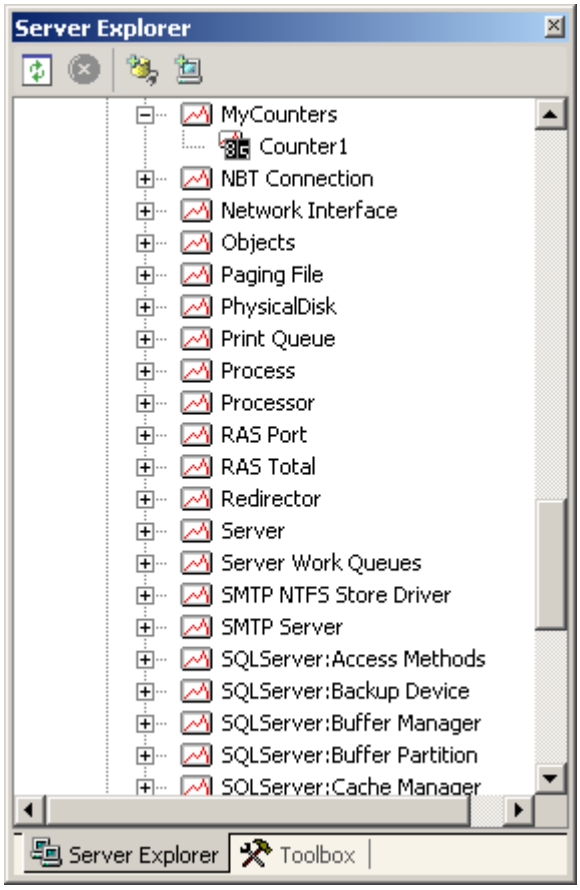
1)Kích chuột phải vào nút Performance Counters trong Server Explorer.

2)Chọn Create New Category từ menu chuột phải.

3)Trong cửa sổ Performance Counter Builder, chỉ định một tên cho Performance Counters Category bạn tạo ra.

4.Nhấn New để tạo ra một Counter mới. Chỉ định giá trị cho thuộc tính Name và thuộc tính Type của Conuter vừa tạo ra.

NET Studio Visual tạo ra một Category mới của Performacne Counters và cho thêm các Performancer Counter vào trong Category đó. Bạn có thể Performance Counter tùy chỉnh trong Server Explorer. Một Performance Counters tùy chỉnh với tên Counter1 được hiển thị trong hình sau đây.



Sau khi bạn tạo ra các Performance Counters tùy chỉnh, thêm nó vào ứng dụng của bạn bằng cách kéo Performance Counter từ Server Explorer vào ứng dụng của bạn. Sau đó, bạn có thể thêm mã lệnh để tăng, giảm, hoặc thiết lập giá trị của hiệu suất truy cập như trong đoạn mã sau.

Visual Basic .NET

```
' Increments the value of counter by 1

PerformanceCounter1.Increment()

' Increments the value of counter by 5

PerformanceCounter1.IncrementBy(5)

' Decreases the value of counter by 1
```

```
PerformanceCounter1.Decrement()

' Sets the value of counter to 10

PerformanceCounter1.RawValue = 10

' Gets the value of counter

Dim val as Integer = PerformanceCounter1.RawValue
```

Visual C#

```
// Increments the value of counter by 1

performanceCounter1.Increment();

// Increments the value of counter by 5

performanceCounter1.IncrementBy(5);

// Decreases the value of counter by 1

performanceCounter1.Decrement();

// Sets the value of counter to 10

performanceCounter1.RawValue = 10;

// Gets the value of counter

int val = performanceCounter1.RawValue;
```

Sau khi bạn tạo một ứng dụng dịch vụ, thêm chức năng cho nó, và cho phép nó để xử lý các sự kiện và thông tin đăng nhập, bạn cần phải cài đặt các ứng dụng dịch vụ, và sau đó bắt đầu nó. Bài học kế tiếp sẽ hướng dẫn cách tạo bộ cài cho dịch vụ và cách cài đặt.

1.1.4. Thêm trình cài đặt, thiết đặt bảo mật,hủy cài đặt một dịch vụ Windows

Tim về vai trò của chương trình cài đặt

NET Framework. Bao gồm các thành phần cài đặt, cho phép bạn cài đặt một dịch vụ Windows , chẳng hạn như các bản ghi tùy chỉnh và Performanace Counter, mà một ứng dụng dịch vụ sử dụng. Các thành phần cài đặt tự động cài đặt các tài nguyên khi bạn cài đặt một dịch vụ và gỡ bỏ chúng khi bạn gỡ bỏ cài đặt các dịch vụ. Bạn sử dụng các thành phần cài đặt có mục đích chung và được xác định trước để cài đặt các dịch vụ của bạn và các thành phần mà các dịch vụ sử dụng. Ngoài ra, bạn có thể tạo các trình cài đặt tùy chỉnh để cài đặt các thành phần của bạn.

Phần cài đặt bao gồm các lớp cài đặt cho phép bạn thực hiện các tác vụ như xác định vị trí để cài đặt các ứng dụng dịch vụ và các thành phần liên quan. Trình cài đặt được xác định trước bao gồm các thành phần cài đặt mà bạn có thể sử dụng để cài đặt các bản ghi sự kiện Event Logs, Performance Counter, Windows Service, và message queues. Sử dụng các trình cài đặt được xác định trước để cấu hình các thành phần này. Bạn cũng có thể sử dụng trình cài đặt được xác định trước để cài đặt và cấu hình các bản ghi sự kiện tùy chỉnh, Performance Counter, và message queues mà bạn tạo ra trong ứng dụng của bạn. Các lớp cài đặt do NET Framework. Cung cấp bao gồm:

System.Diagnostics.EventLogInstaller.

Lớp này cho phép bạn cấu hình các EventLogs mặc định, chẳng hạn như các bản ghi ứng dụng và hệ thống ứng dụng của bạn sử dụng.

Lớp System.Diagnostics.EventLogInstaller cũng cho phép bạn cài đặt và cấu hình các bản ghi sự kiện tùy chỉnh mà bạn tạo cho dịch vụ của bạn.

System.Diagnostics.PerformanceCounterInstaller.

Lớp này cho phép bạn cấu hình các PerformanceCounter cài đặt trên hệ thống. Nó cũng cho phép bạn cài đặt và cấu hình các PerformanceCounter, các tùy chỉnh mà bạn tạo cho dịch vụ của bạn.

System.ServiceProcess.ServiceInstaller và

System.ServiceProcess.Ser-viceProcessInstaller.

Hai lớp này cho phép bạn cài đặt và cấu hình một dịch vụ Windows trên một máy tính. Bạn cần phải thêm vào một thể hiện của lớp ***System.ServiceProcess.ServiceProcessInstaller*** cho các ứng dụng dịch vụ của bạn và một thể hiện của lớp System.ServiceProcess.ServiceInstaller cho mỗi dịch vụ trong ứng dụng dịch vụ của bạn. Lớp ServiceProcessInstaller thực hiện các nhiệm vụ được phổ biến cho tất cả các dịch vụ trong một ứng dụng dịch vụ, chẳng hạn như viết mục trong hệ thống registry, trong khi lớp ServiceInstaller thực hiện các nhiệm vụ cụ thể cho một dịch vụ. Ví dụ, lớp ServiceInstaller thông qua tên của thuộc tính ServiceName của một dịch vụ tiện ích cài đặt. Các tiện ích cài đặt sử dụng ServiceName để tạo ra một mục đăng ký cho dịch vụ trong \ HKEY_LOCAL_MACHINE System \ CurrentControlSet \ Services registry key.

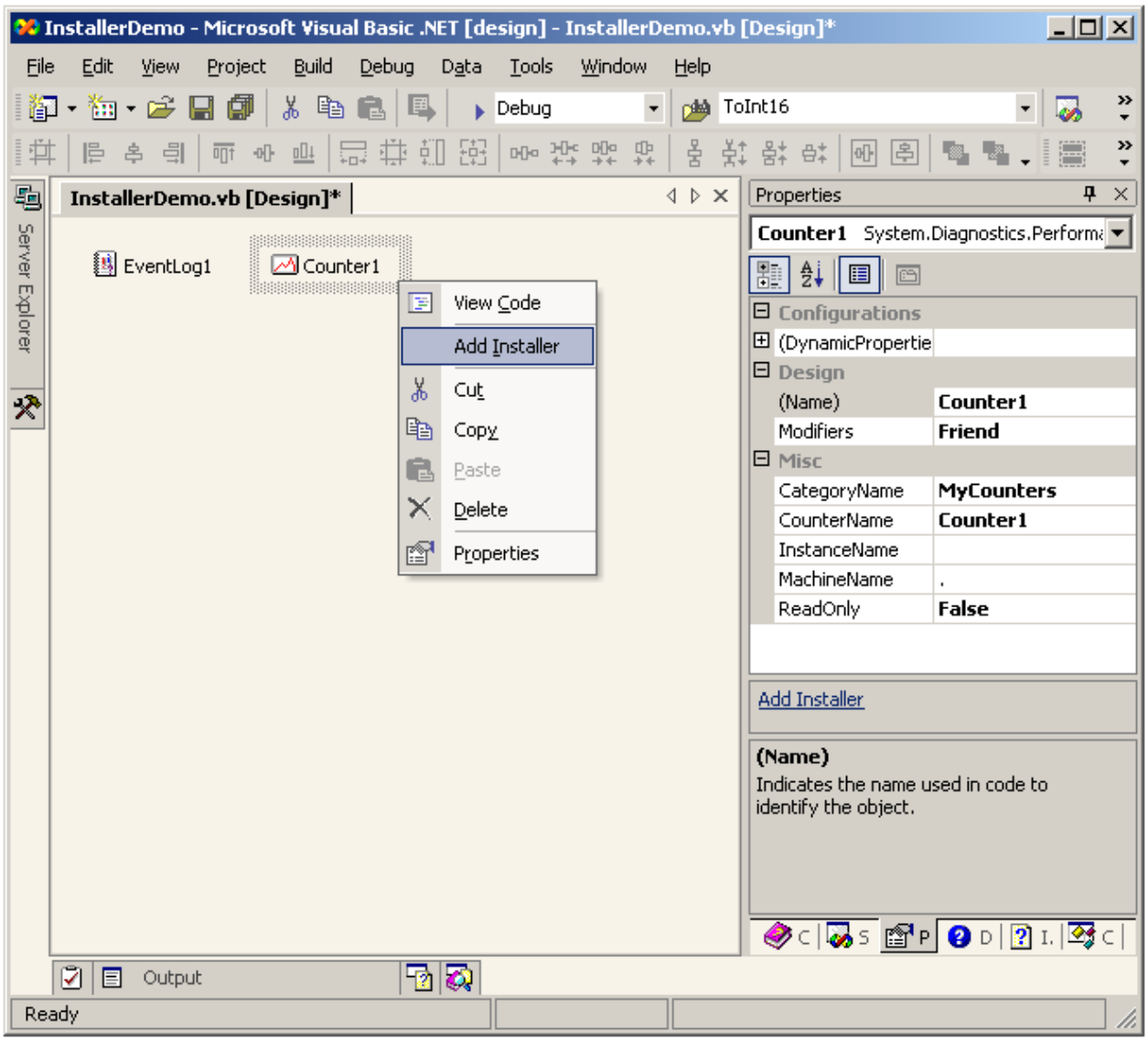
System.Messaging.MessageQueueInstaller.

Lớp này cho phép bạn cấu hình và cài đặt hàng đợi tin nhắn là ứng dụng dịch vụ của bạn sử dụng để gửi và nhận tin nhắn.

Thêm trình cài đặt

Để cài đặt ứng dụng dịch vụ của bạn, bạn cần phải tạo ra một lớp cài đặt, xuất phát từ lớp System.Configuration.Install.Installer. Các lớp cài đặt có chứa một tập hợp (collection) được gọi là trình cài đặt, trong đó có các thành phần cài đặt bao gồm các nguồn tài nguyên của ứng dụng dịch vụ của bạn sử dụng thời gian chạy. Bạn thêm các EventLogInstallers, PerformanceCounterInstallers, ServiceInstallers,-Installer ServiceProcess, và các MessageQueueInstallers vào trình cài đặt.

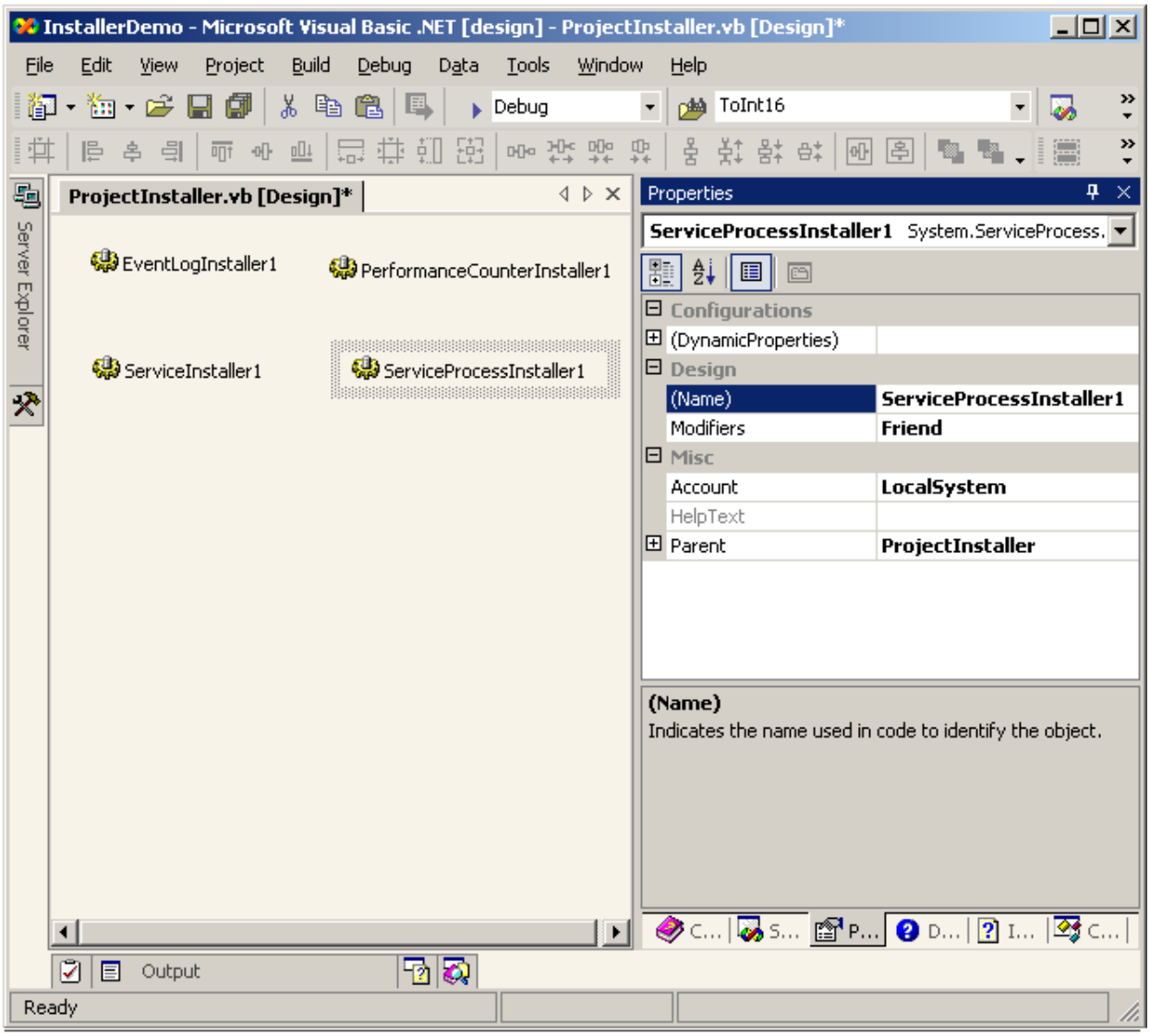
Bạn thêm các trình cài đặt cho dự án ứng dụng dịch vụ của bạn cho các thành phần khác nhau bao gồm các bản ghi sự kiện và các bộ đếm hiệu suất cho các ứng dụng dịch vụ bằng cách sử dụng Visual Studio NET Component Designer. Để thêm trình cài đặt cho một thành phần, nhấp vào liên kết cài đặt Add trong cửa sổ Properties. Bạn cũng có thể nhấp chuột phải vào các thành phần mà bạn muốn thêm vào một trình cài đặt, và sau đó chọn Add Installer từ menu chuột phải. Các tùy chọn cài đặt Add được thể hiện trong hình 2.7.



Như đã đề cập trước đó, bạn cần thêm một thể hiện của lớp ServiceProcessInstaller cho một ứng dụng dịch vụ Windows và một thể hiện của lớp ServiceInstaller cho mỗi dịch vụ Windows trong ứng dụng dịch vụ. Để thêm các cài đặt cho ứng dụng dịch vụ của bạn, hãy nhấp vào cửa sổ thiết kế (Design windows) cho các ứng dụng dịch vụ của bạn, và sau đó nhấp vào liên kết cài đặt Add trong cửa sổ Properties. Visual Studio. NET sẽ tự động tạo ra một lớp ProjectInstaller trong đó bao gồm một thể hiện của lớp ServiceProcess-Installer và một thể hiện của lớp ServiceInstaller. Hình 2.8 cho thấy lớp

ProjectInstaller cho một dự án dịch vụ Windows bao gồm cài đặt cho một Event logs và PerformanceCounter.

Lớp ProjectInstaller mã được tạo ra bởi Studio Visual.NET . Lưu ý rằng các EventLogInstaller, PerformanceCounterInstaller, ServiceInstaller, và các ServiceProcessInstaller được thêm vào tập hợp các thành phần cài đặt của lớp ProjectInstaller.



Hình 2-8. Cài đặt các Class trong cửa sổ thiết kế

Visual Basic .NET

```
Imports System.ComponentModel
Imports System.Configuration.Install
<RunInstaller(True)> Public Class ProjectInstaller
Inherits System.Configuration.Install.Installer


#Region " Component Designer generated code "
    Public Sub New()
        MyBase.New()
        'This call is required by the Component Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
    End Sub
    'Installer overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
```

```
End Sub
'Required by the Component Designer
Private components As System.ComponentModel.IContainer
'NOTE: The following procedure is required by the Component Designer
'It can be modified using the Component Designer.
'Do not modify it using the code editor.
Friend WithEvents EventLogInstaller1 As_
    System.Diagnostics.EventLogInstaller
Friend WithEvents ServiceProcessInstaller1 As _
    System.ServiceProcess.ServiceProcessInstaller
Friend WithEvents ServiceInstaller1 As _
    System.ServiceProcess.ServiceInstaller
Friend WithEvents PerformanceCounterInstaller1 As _
    System.Diagnostics.PerformanceCounterInstaller
<System.Diagnostics.DebuggerStepThrough()> Private Sub _
    InitializeComponent()
    Me.EventLogInstaller1 = New System.Diagnostics.EventLogInstaller()
    Me.ServiceProcessInstaller1 = New
    System.ServiceProcess.ServiceProcessInstaller()
    Me.ServiceInstaller1 = New System.ServiceProcess.ServiceInstaller()
    Me.PerformanceCounterInstaller1 = New _
```

```
    System.Diagnostics.PerformanceCounterInstaller()
'
'EventLogInstaller1
'
Me.EventLogInstaller1.Log = "Application"
Me.EventLogInstaller1.Source = "MyService"
'
'ServiceProcessInstaller1
'
Me.ServiceProcessInstaller1.Account = _
    System.ServiceProcess.ServiceAccount.LocalSystem
Me.ServiceProcessInstaller1.Password = Nothing
Me.ServiceProcessInstaller1.Username = Nothing
'
'ServiceInstaller1
'
Me.ServiceInstaller1.ServiceName = "MyService"
'
'PerformanceCounterInstaller1
'
Me.PerformanceCounterInstaller1.CategoryHelp = "None"
```

```
        Me.PerformanceCounterInstaller1.CategoryName = "MyCounters"
        Me.PerformanceCounterInstaller1.Counters.AddRange(New _
            System.Diagnostics.CounterCreationData() {New _
            System.Diagnostics.CounterCreationData("MyAppCounter", "", _
            System.Diagnostics.PerformanceCounterType.NumberOfItems32)})
        '
        'ProjectInstaller
        '
        Me.Installers.AddRange(New System.Configuration.Install.Installer() _
            {Me.EventLogInstaller1, Me.ServiceProcessInstaller1, _
            Me.ServiceInstaller1, Me.PerformanceCounterInstaller1})
    End Sub
#End Region
End Class
```

Visual C#

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Configuration.Install;


namespace InstallerDemoCS
```

```
{
    /// <summary>
    /// Summary description for ProjectInstaller.
    /// </summary>
    [RunInstaller(true)]
    public class ProjectInstaller : System.Configuration.Install.Installer
    {
        private System.ServiceProcess.ServiceProcessInstaller
            serviceProcessInstaller1;
        private System.ServiceProcess.ServiceInstaller serviceInstaller1;
        private System.Diagnostics.EventLogInstaller eventLogInstaller1;
        private System.Diagnostics.PerformanceCounterInstaller
            performanceCounterInstaller1;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        public ProjectInstaller()
        {
            // This call is required by the Designer.
            InitializeComponent();
```

```
    // TODO: Add any initialization after the InitComponent call
}
#region Component Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.serviceProcessInstaller1 = new
        System.ServiceProcess.ServiceProcessInstaller();
    this.serviceInstaller1 = new
        System.ServiceProcess.ServiceInstaller();
    this.eventLogInstaller1 = new
        System.Diagnostics.EventLogInstaller();
    this.performanceCounterInstaller1 = new
        System.Diagnostics.PerformanceCounterInstaller();
    //
    // serviceProcessInstaller1
    //
```

```
this.serviceProcessInstaller1.Password = null;
this.serviceProcessInstaller1.Username = null;
//
// serviceInstaller1
//
this.serviceInstaller1.ServiceName = "MyService";
//
// eventLogInstaller1
//
this.eventLogInstaller1.Log = "Application";
this.eventLogInstaller1.Source = "MyService";
//
// performanceCounterInstaller1
//
this.performanceCounterInstaller1.CategoryHelp = "None";
this.performanceCounterInstaller1.CategoryName = "MyCounters";
this.performanceCounterInstaller1.Counters.AddRange(new
    System.Diagnostics.CounterCreationData[] { new
    System.Diagnostics.CounterCreationData("MyAppCounter",
    "MyAppCounter help",
    System.Diagnostics.PerformanceCounterType.NumberOfItems32
```

```
            )});
        //
        // ProjectInstaller
        //
        this.Installers.AddRange(new
            System.Configuration.Install.Installer[] {
            this.serviceProcessInstaller1,
            this.serviceInstaller1,
            this.eventLogInstaller1,
                this.performanceCounterInstaller1});
    }
    #endregion
  }
}
```

Lớp ProjectInstaller được nhận biết với thuộc tính [RunInstaller (true)] trong Visual C # và <RunInstaller(True)> trong Visual Basic.NET. Điều này đòi hỏi bạn phải cài đặt assembly và các lớp ProjectInstaller bằng cách sử dụng công cụ Installutil.exe. Gọi một trình cài đặt đảm bảo rằng tất cả các thành phần đang có, hoặc cài đặt thành công hoặc không được cài đặt . Nếu một thành phần nào không cài đặt trong quá trình cài đặt, trình cài đặt rollback lại các quá trình cài đặt bằng cách loại bỏ tất cả các thành phần đã được cài đặt trước đó.

Thiết đặt chế độ bảo mật cho một dịch vụ ứng dụng

Ngoài việc đăng ký và cài đặt ứng dụng dịch vụ của bạn, trình cài đặt cho phép bạn xác định bối cảnh an ninh mà trong đó các dịch vụ trong ứng dụng dịch vụ của bạn chạy. Điều này cho phép bạn xác định xem những dịch vụ mà bạn cài đặt trên một máy tính sẽ chạy cho tất cả người dùng đăng nhập vào máy tính hoặc cho một người dùng cụ thể. Bạn sử dụng thuộc tính Account của lớp ServiceProcessInstaller để xác định bối cảnh an ninh cho các ứng dụng dịch vụ của bạn. Bạn có thể thiết lập thuộc tính Account của lớp ServiceProcessInstaller từ cửa sổ Properties. Cửa sổ bên phải của hình 2.8 hiển thị các thuộc tính của lớp ServiceProcessInstaller.

Bạn có thể thiết lập thuộc tính Account để các giá trị là LocalService, LocalSystem, NetworkService hoặc User.

Bảng 2.5 mô tả các giá trị của thuộc tính Account.

Methed	Description
LocalService	Cho phép một dịch vụ để chạy trong bối cảnh của một tài khoản cung cấp rộng rãi đặc quyền local privileges và cung cấp các thông tin quan trọng của máy tính đến một máy chủ từ xa.
LocalSystem	Cho phép một dịch vụ để chạy trong bối cảnh của một tài khoản hoạt động như một người sử dụng non-privileged (Không đặc quyền) trên máy tính local và cung cấp các thông tin nặc danh đến một máy chủ từ xa.
NetworkService	Cho phép một dịch vụ để chạy trong bối cảnh của một tài khoản hoạt động như một người sử dụng non-privileged trên máy tính local và cung cấp các thông tin quan trọng của máy tính đến một máy chủ từ xa.

User	Cho phép một dịch vụ để chạy trong bối cảnh của một người sử dụng. Bạn cần cung cấp một tên người dùng và mật khẩu hợp lệ khi bạn cài đặt ứng dụng dịch vụ trên một máy tính.

Cài đặt và Gỡ cài đặt một dịch vụ Windows

Sau khi bạn thêm các trình cài đặt ứng dụng dịch vụ của bạn và xác định bối cảnh an ninh, trong đó dịch vụ này sẽ chạy, bạn đã sẵn sàng để xây dựng các ứng dụng dịch vụ và cài đặt nó trên một máy tính. Để xây dựng một ứng dụng dịch vụ, mở trình đơn Build và chọn Build Solution.

Sau khi bạn xây dựng các ứng dụng dịch vụ của bạn, Visual Studio. NET tạo ra một tập tin .exe cho ứng dụng dịch vụ của bạn. Sau đó, bạn cần phải cài đặt các tập tin exe trên một máy tính. NET Framework. Cung cấp công cụ Installutil.exe, cho phép bạn cài đặt một ứng dụng dịch vụ. Bạn có thể chạy công cụ này từ dấu nhắc lệnh. Cú pháp cho việc sử dụng công cụ này để cài đặt một ứng dụng dịch vụ là:

Installutil <tên tập tin exe>

ví dụ, installutil MyWindowsService.exe.

Khi bạn chạy công cụ InstallUtil, nó chạy tất cả các trình cài đặt trong ứng dụng dịch vụ của bạn. Những phần cài đặt sau đó cài đặt ứng dụng dịch vụ của bạn và các tài nguyên liên quan bao gồm trong ứng dụng dịch vụ theo các thuộc tính bạn đã chỉ định. Nếu bạn chỉ định thuộc tính Account của các thể hiện ServiceProcessInstaller trong ứng dụng dịch vụ của bạn như là User, bạn sẽ được nhắc nhở cho một tên người dùng và mật khẩu(sẽ xuất hiện một hộp

thoại yêu cầu nhập username và password). Hình 2.9 hiển thị các dịch vụ hộp thoạ yêu cầu bạn nhập tên người dùng và mật khẩu hợp lệ để tiến hành cài đặt.

Hình 2-9. The Set Service Login dialog box

Sau khi bạn nhập tên người dùng và mật khẩu của tài khoản, công cụ installutil hoàn tất quá trình cài đặt, và một tin nhắn được hiển thị trong cửa sổ Command Prompt. Hình 2,10 hiển thị cửa sổ Command Prompt sau khi một ứng dụng dịch vụ được cài đặt trên một máy tính.

Các công cụ installutil cũng cho phép bạn gỡ bỏ cài đặt một ứng dụng dịch vụ. Để gỡ bỏ một ứng dụng dịch vụ, chạy công cụ installutil với tùy chọn / u. Cú pháp để gỡ bỏ cài đặt một ứng dụng dịch vụ bằng cách sử dụng công cụ installutil

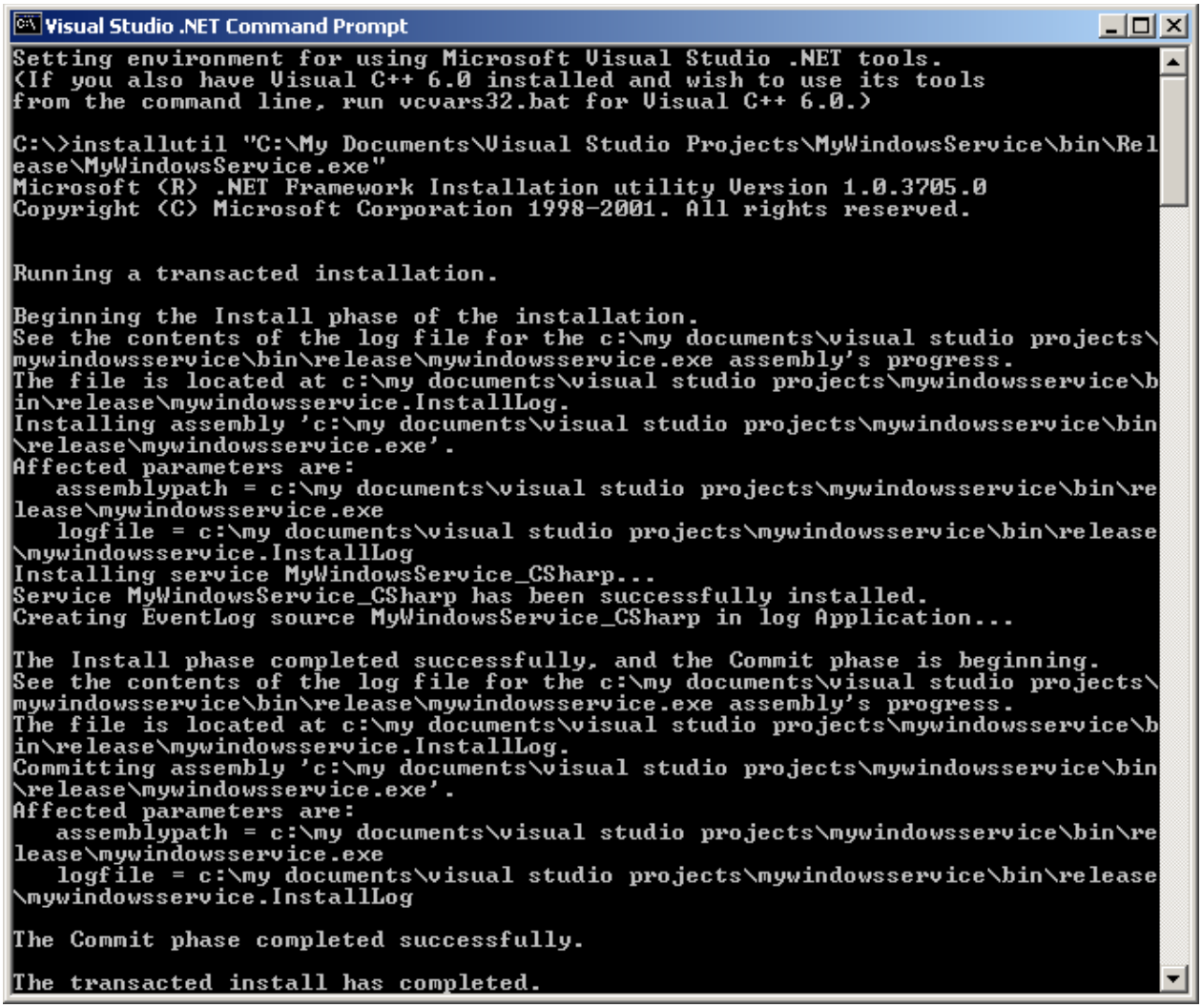
Installutil / u <tên tập tin exe.>

ví dụ, installutil / u MyWindowsService.exe.

Ngoài ra bạn cũng có thể sử dụng tên của Service sau khi đã cài đặt hiển thị trong SCM, chẳng hạn sau khi cài đặt nó hiển thị với tên ” temp service” tiến hành xóa như sau:

Sc delete ” temp service”.

Sau khi bạn cài đặt thành công một dịch vụ, bạn có thể quản lý và kiểm soát hành vi của mình theo cách thủ công bằng cách sử dụng SCM. Ngoài ra, bạn cũng có thể quản lý và kiểm soát một dịch vụ lập trình bằng cách sử dụng các phương thức của lớp ServiceController. Bạn sẽ tìm hiểu về việc quản lý các dịch vụ Windows trong bài học kế tiếp.



Hình 2-10. The Visual Studio .NET Command Prompt window

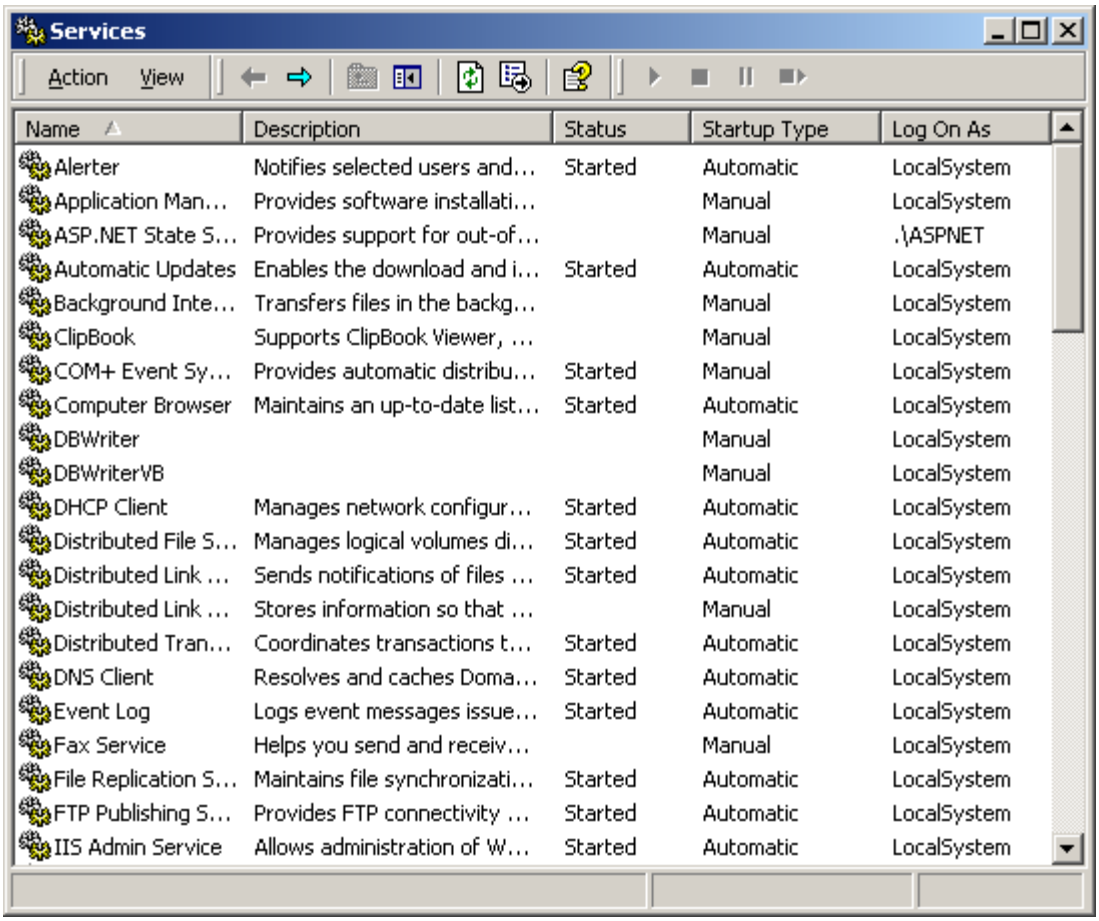
1.1.5.Quản lý một window service

Sau khi bạn tạo ra và cài đặt một dịch vụ Windows, bạn có thể quản lý dịch vụ của bạn bằng cách thực hiện các nhiệm vụ như thay đổi trạng thái của các dịch

vụ của bạn. Bạn có thể quản lý các dịch vụ của bạn một cách thủ công bằng cách sử dụng SCM hoặc lập trình bằng cách sử dụng một thành phần của lớp ServiceController trong một ứng dụng.

Bạn truy cập vào SCM bằng cách sử dụng cửa sổ Services. Để mở cửa sổ Services, mở Control Panel, kích đúp vào Administrative Tools, và sau đó nhấp đúp vào Service. Hình 2,11 hiển thị cửa sổ Services.

Cửa sổ Services sẽ hiển thị tất cả các dịch vụ được cài đặt. Để thay đổi trạng thái của một dịch vụ, nhấp vào dịch vụ, và sau đó sử dụng các nút tương ứng trên thanh công cụ Start, Pause, Stop, hoặc Continue.



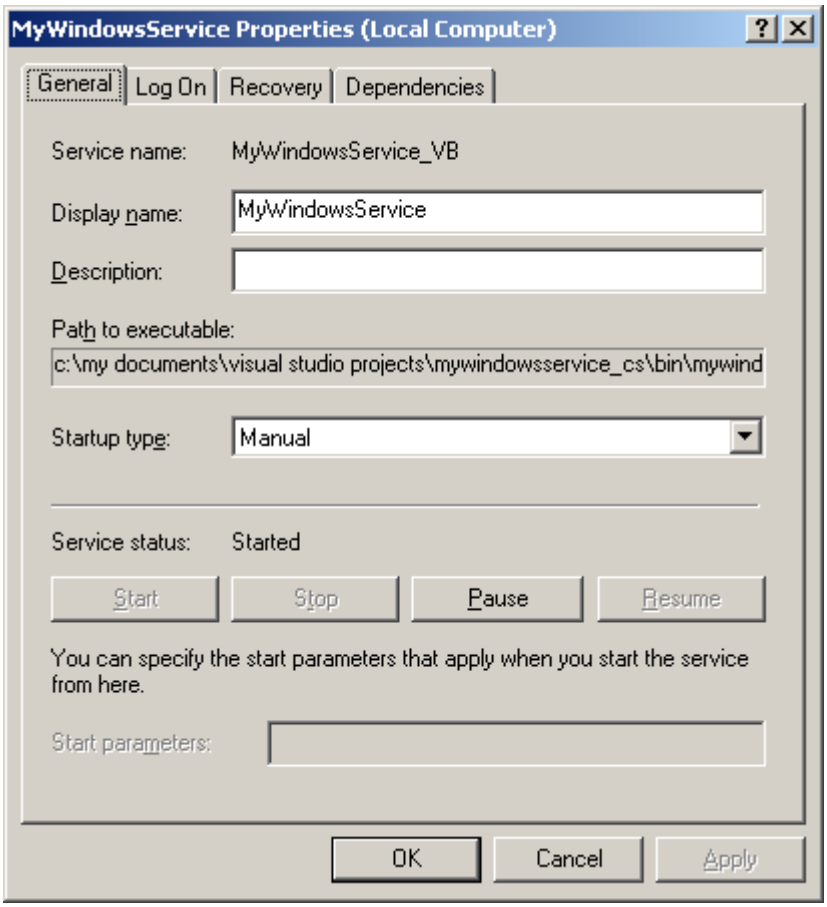
Hình 2-11. A list of services displayed in the Services window

Lưu ý

Bạn có thể thay đổi trạng thái của một dịch vụ nếu các thuộc tính thích hợp của dịch vụ được quy định cụ thể. Ví dụ, bạn có thể tạm dừng và tiếp tục lại một dịch vụ nếu thuộc tính CanPauseAndContinue của các ứng dụng dịch vụ được thiết lập là True.

Bạn cũng có thể xác định cách các dịch vụ bắt đầu bằng cách thay đổi kiểu khởi động của dịch vụ. Để thay đổi kiểu khởi động cho một ứng dụng dịch vụ, hoàn thành các bước sau:

1-Chọn một dịch vụ và chọn Properties trên thanh công cụ. Hình 2,12 hiển thị hộp thoại thuộc tính cho một ứng dụng dịch vụ.



Hình 2-12. The properties dialog box

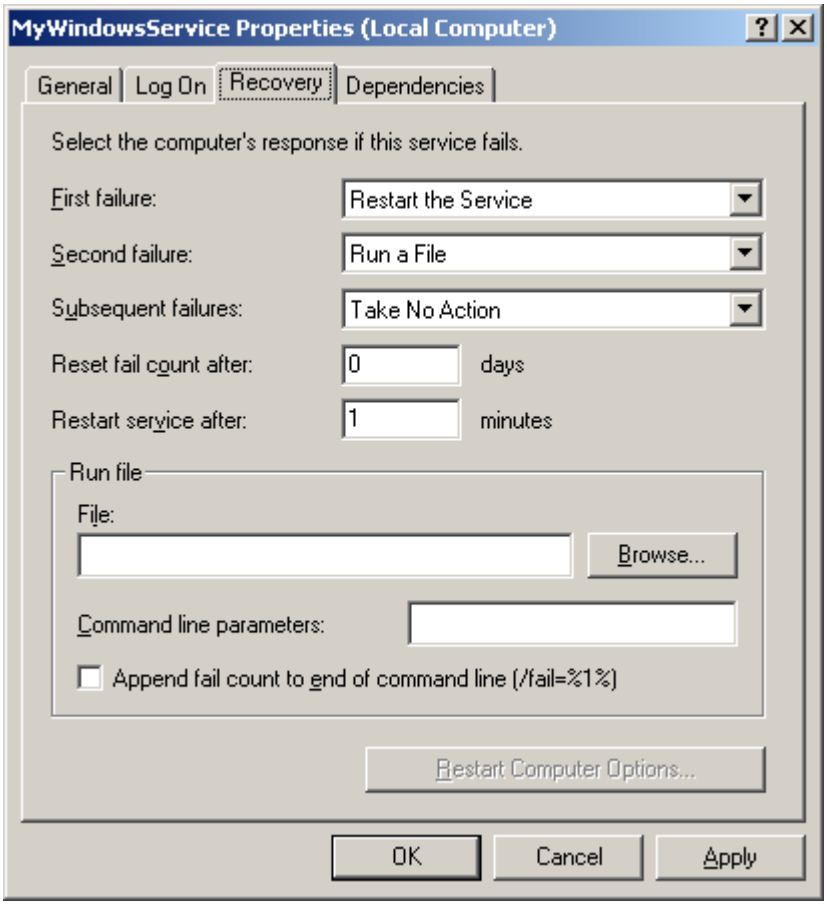
2- Chọn loại thích hợp khởi động từ Startup Type danh sách thả xuống trên tab General.

3-Click vào OK để đóng hộp thoại thuộc tính.

SCM cũng cho phép bạn để xác định các hành động phục hồi trong trường hợp một dịch vụ không thành công. Để xác định các hành động phục hồi cho một dịch vụ, thực hiện các bước sau:

1-Chọn một dịch vụ và chọn Properties trên thanh công cụ. Hộp thoại thuộc tính mở ra.

2-Chọn tab Recovery. Hình 2,13 hiển thị các tab phục hồi của hộp thoại thuộc tính.



Hình 2-13. The Recovery tab

3-Xác định các hành động mà bạn muốn thực hiện những thất bại đầu tiên, thứ hai, và sau đó subsequent failures của service. Bạn có thể chỉ định SCM khởi động lại dịch vụ, chạy một chương trình, khởi động lại máy tính, hoặc không có hành động.

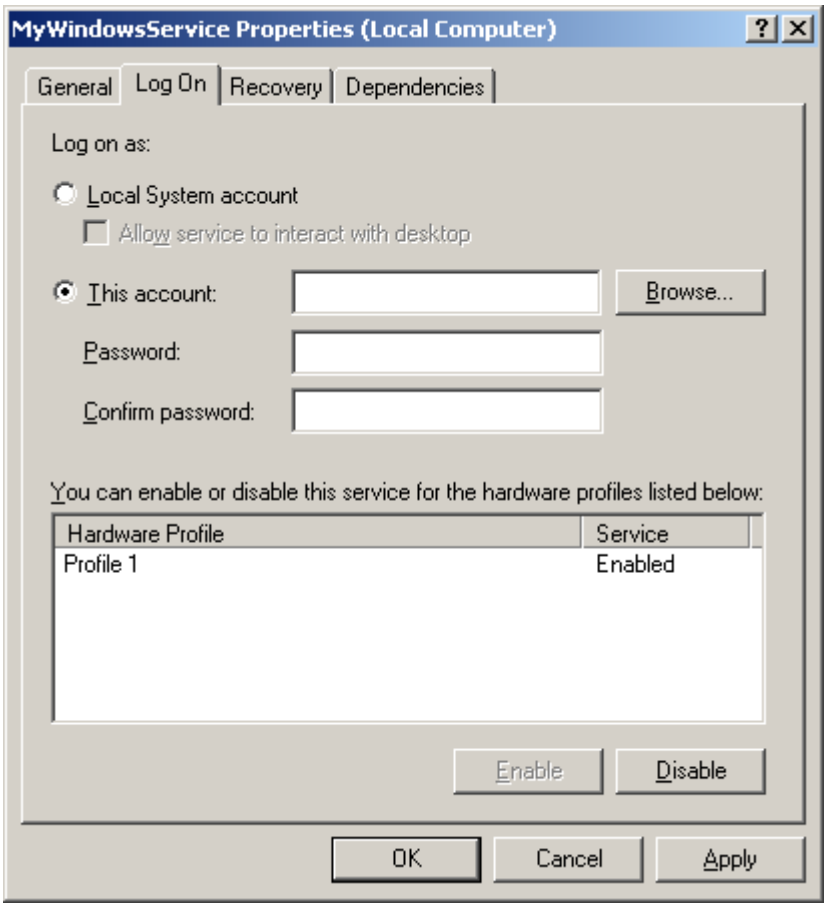
4-Click vào OK để đóng hộp thoại thuộc tính.

Bạn cũng có thể sử dụng SCM để xác định tài khoản mà một dịch vụ sử dụng để đăng nhập vào một máy tính. Điều này cho phép bạn thay đổi các đặc quyền mà dịch vụ của bạn có trên máy tính. Tài khoản mà bạn chỉ định cũng xác định một dịch vụ tương tác như thế nào với mạng. Bạn có thể xác định rằng một dịch vụ chạy trong bối cảnh an ninh của một tài khoản LocalSystem hoặc tài khoản

người dùng. Để xác định các tài khoản mà một dịch vụ sử dụng để đăng nhập vào một máy tính, hoàn thành các bước sau:

1-Chọn một dịch vụ và chọn Properties trên thanh công cụ. Hộp thoại thuộc tính mở ra.

2-Nhấp vào tab Log On. Hình 2,14 hiển thị Log On tab của hộp thoại thuộc tính.



Hình 2-14. The Log On tab

3-Chọn tùy chọn Local System Account nếu bạn muốn chạy các dịch vụ trong bối cảnh an ninh của một tài khoản LocalSystem. Ngoài ra, bạn có thể chọn tài khoản này và chỉ định tài khoản người dùng mà bạn muốn dịch vụ của bạn để sử dụng. Bạn cũng có thể chọn một tài khoản người dùng bằng cách duyệt tất cả các tài khoản người dùng trong mạng.

4-Click vào OK để đóng hộp thoại thuộc tính.

Ngoài SCM, bạn có thể sử dụng các phương pháp của lớp ServerController để kiểm soát một dịch vụ lập trình .

Sử dụng Class ServiceController

Các lớp ServiceController cho phép bạn kết nối và kiểm soát các dịch vụ Windows đang chạy trên một máy tính. Giống như SCM, bạn có thể sử dụng các phương thức của lớp ServiceController để thực hiện các nhiệm vụ khác nhau, chẳng hạn như starting, stopping, pausing, and resuming services. Tuy nhiên, không giống như SCM, bạn có thể thực hiện các lệnh tùy chỉnh bằng cách sử dụng các phương thức của lớp ServiceController. Bạn có thể thực hiện các nhiệm vụ sau đây bằng cách sử dụng các lớp ServiceController:

1-Xem danh sách các dịch vụ trên một máy tính

2-Bắt đầu và ngừng một dịch vụ

3-Tạm dừng và tiếp tục lại một dịch vụ

4-Truy vấn và lấy các thuộc tính của một dịch vụ

5-Chỉ định một lệnh tùy chỉnh để thực hiện trên dịch vụ của bạn

Để kiểm soát hành vi của một dịch vụ, bạn viết mã để xác định những nhiệm vụ rằng các dịch vụ cần thực hiện khi một sự kiện xảy ra. Ví dụ, bạn có thể xác định phương thức onPause, được gọi đến khi sự kiện tạm dừng được gửi đến ứng dụng dịch vụ của bạn. Tuy nhiên, bạn cần phải đảm bảo rằng các thuộc tính tương ứng cho các sự kiện mà bạn muốn kiểm soát được thiết lập cho phù hợp. Ví dụ, nếu bạn thiết lập thuộc tính CanPause cho một dịch vụ là False, bạn không thể dừng dịch vụ bằng cách sử dụng ServiceController hoặc kiểm soát dịch vụ quản lý.

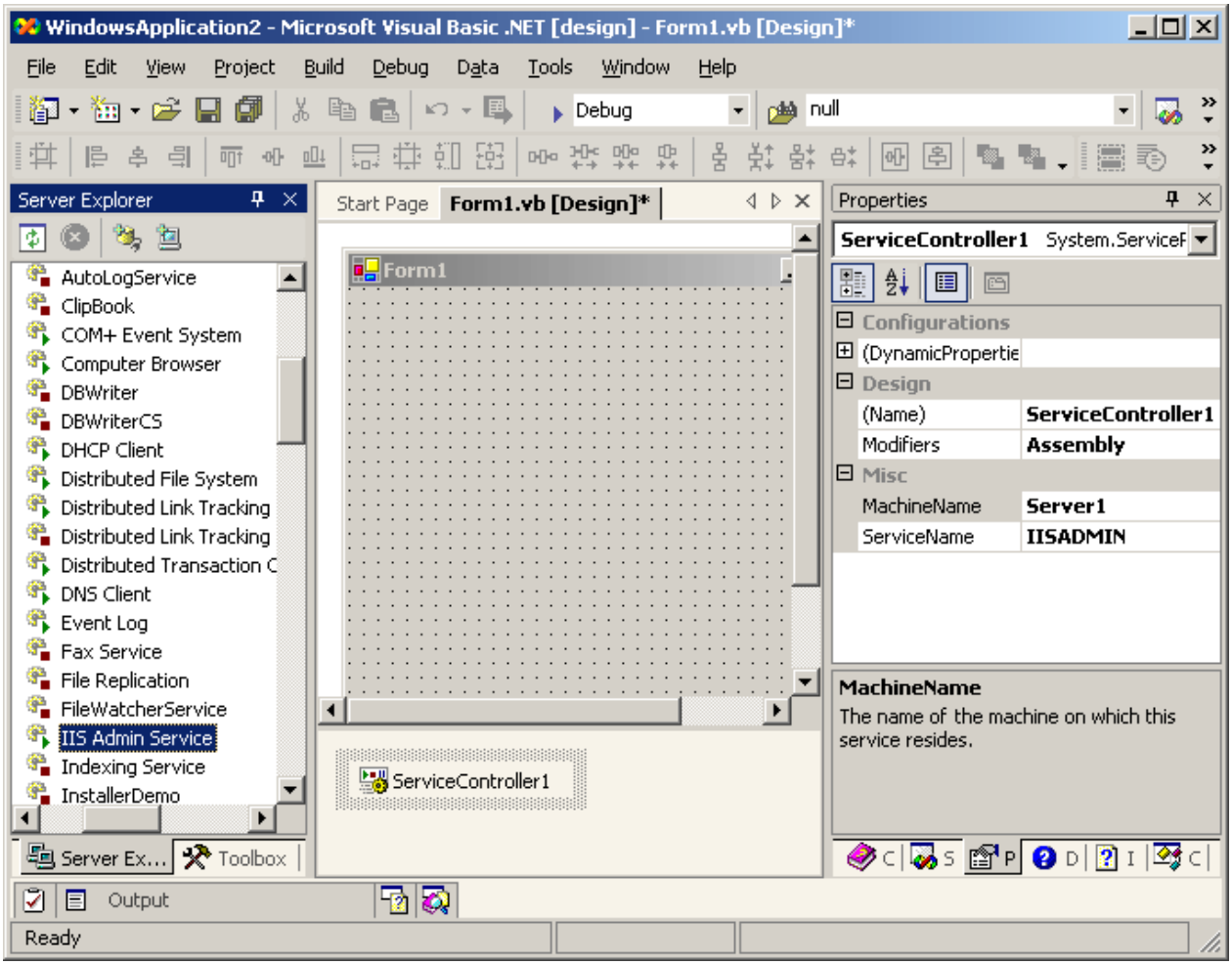
Để sử dụng các phương thức của lớp ServiceController, bạn tạo ra một thể hiện của các thành phần ServiceController. Tiếp theo, bạn thiết lập các thuộc tính cho thể hiện này, cho phép bạn xác định các dịch vụ mà bạn muốn kết nối tới các dịch vụ đang chạy. Sau đó, bạn có thể sử dụng các phương thức của lớp ServiceController để kiểm soát hành vi của dịch vụ bằng cách gọi các phương thức thích hợp của lớp ServiceController. Khi bạn gọi một phương thức của lớp ServiceController, các thể hiện của các ServiceController component thông qua một yêu cầu hành động tới SCM. SCM sau đó thông qua các yêu cầu với dịch vụ, và các ứng dụng dịch vụ thực hiện các phương thức tương ứng với sự kiện này.

Lưu ý

Bạn nên luôn luôn tạo ra một ứng dụng riêng biệt có chứa một thành phần của lớp ServiceController để kiểm soát một ứng dụng dịch vụ.

Trước tiên hãy để ý cách tạo ra một thể hiện của các thành phần ServiceController. Bạn có thể kéo một thể hiện của các thành phần ServiceController từ tab Components của hộp công cụ ToolBox và thả nó vào dưới dạng một ứng dụng Windows. Sau đó, bạn cần phải xác định tên của các dịch vụ mà bạn muốn kiểm soát và máy tính mà trên đó các dịch vụ đang chạy bằng cách sử dụng ServiceName và thuộc tính MachineName, tương ứng.

Hình 2,16 hiển thị một ứng dụng Windows với một thể hiện của các thành phần ServiceController.



Hình 2-16. A Windows application with an instance of the ServiceController component

Bạn cũng có thể thêm một thể hiện của các thành phần ServiceController từ Server Explorer. Để thêm một thể hiện của các thành phần ServiceController từ Server Explorer, hoàn tất các bước sau đây:

1-Trong Server Explorer, xác định vị trí máy chủ mà trên đó các dịch vụ mà bạn muốn kiểm soát đang chạy.

2-Mở rộng nút Dịch vụ cho máy chủ đó, và xác định vị trí các dịch vụ mà bạn muốn kiểm soát.

3-Kích chuột phải vào tên của dịch vụ và chọn Add To Designer từ menu chuột phải. Một thành phần ServiceController xuất hiện trong dự án ứng dụng Windows của bạn, và thành phần được cấu hình để tương tác với các dịch vụ

mà bạn cần phải kiểm soát. Vì vậy, bạn không cần phải xác định các thuộc tính MachineName và ServiceName này thể hiện của các thành phần ServiceController. Hình 2,17 hiển thị danh sách các dịch vụ trong Server Explorer.

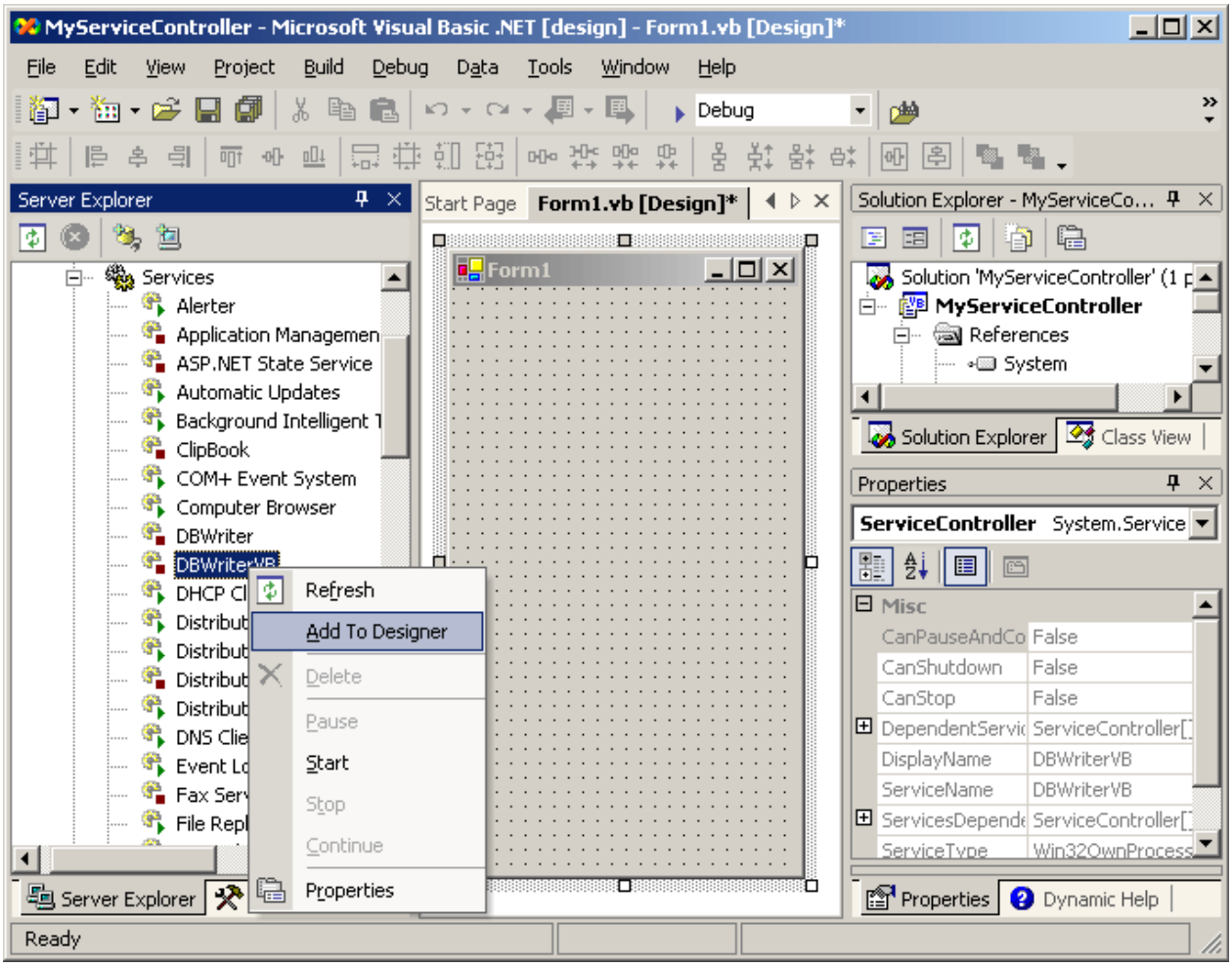


Figure 2-17. Server Explorer with a list of services

Bạn cũng có thể tạo ra một thể hiện của các thành phần ServiceController lập trình. Các mã sau đây cho thấy làm thế nào để tạo ra một thể hiện của một thành phần ServiceController trong một ứng dụng Windows.

Visual Basic .NET

```
Dim DBWriterController As System.ServiceProcess.ServiceController
Me.DBWriterController = New System.ServiceProcess.ServiceController()
Me.DBWriterController.MachineName = "NancyD"
Me.DBWriterController.ServiceName = "DBWriterVB"
```

Visual C#

```
private System.ServiceProcess.ServiceController serviceController1;
this.serviceController1 = new System.ServiceProcess.ServiceController();
this.serviceController1.MachineName = " NancyD";
this.serviceController1.ServiceName = " DBWriterCS";
```

Sau khi bạn tạo một thể hiện của các thành phần ServiceController trong ứng dụng Windows của bạn, bạn gọi các phương thức của lớp ServiceController.

Bảng 2.4 trong phần 1.1.1 mô tả một số trong những phương thức của lớp ServiceController. Bây giờ chúng ta hãy nhìn vào kiểm soát một dịch vụ bằng cách sử dụng các phương thức của lớp ServiceController. Các mã sau đây sẽ hiển thị một phương pháp mà bắt đầu một dịch vụ từ một ứng dụng Windows.

Lưu ý

Để cho các ứng dụng sau đây để chạy một cách chính xác, bạn cần thêm một tham chiếu đến không gian tên System.ServiceProcess.

Visual Basic .NET

```
Sub StartService()
   ' Check the current status of the service before you
   ' try to change the current status of the service
   If (DBWriterController.Status = ServiceControllerStatus.Stopped) Then
      DBWriterController.Start()
      MessageBox.Show("Service Started")
   Else
      MessageBox.Show("Service Running")
   End If
End Sub
```

Visual C#

```
private void StartService()
{
   /* Check the current status of the service before you
   try to change the current status of the service */
   if (DBWriterController.Status == ServiceControllerStatus.Stopped)
   {
      DBWriterController.Start();
      MessageBox.Show("Service Started");
   }
   else
   {
      MessageBox.Show("Service Running");
   }
}
```

Tương tự như vậy, bạn có thể sử dụng các phương thức khác của lớp ServiceController, như Pause, Continue , và Stop, để kiểm soát một dịch vụ. Các mã sau đây sẽ hiển thị một phương thức mà dừng lại một dịch vụ đang chạy trên một máy tính.

Visual Basic .NET

```
Sub PauseService()
   ' Check the current status of the service before you try
   ' to change the current status of the service*/
   If (DBWriterController.Status = ServiceControllerStatus.Running) Then
      ' Check whether you can pause the service
```

```
        If (DBWriterController.CanPauseAndContinue = True) Then
            DBWriterController.Pause()
            MessageBox.Show("Service Paused")
        Else
          MessageBox.Show("You Can not Pause the Service")
        End If
    Else
        MessageBox.Show("Service not Running")
    End If
End Sub
```

Visual C#

```
private void PauseService()
{
    /* Check the current status of the service before you try to
       change the current status of the service*/
    if (DBWriterController.Status == ServiceControllerStatus.Running)
    {
        //Check whether you can pause the service
        if(DBWriterController.CanPauseAndContinue==true)
        {
            DBWriterController.Pause();
```

```
        MessageBox.Show("Service Paused");
    }
    else
        MessageBox.Show("You Can not Pause the Service");
    }
    else
        MessageBox.Show("Service not Running");
}
```

Khi thêm vào các phương thức Start, Stop, Pause, và Continue, bạn có thể sử dụng phương thức ExecuteCommand để chạy các lệnh tùy chỉnh trên dịch vụ của bạn. Để chạy các lệnh tùy chỉnh trên ứng dụng dịch vụ của bạn, hoàn thành các bước sau:

1-Tạo ra một phương pháp gọi là phương pháp ServiceController.ExecuteCommand trong các ứng dụng mà bạn sử dụng để kiểm soát ứng dụng dịch vụ của bạn.

2-Ghi đè lên các phương pháp OnCustomCommand trong ứng dụng dịch vụ của bạn để xác định các nhiệm vụ mà bạn muốn ứng dụng dịch vụ của bạn để thực hiện.

Các mã sau đây cho thấy phương thức RunCommand, mà khi gọi tới phương thức Controller.ExecuteCommand của dịch vụ. Phương pháp ServiceController.ExecuteCommand có một giá trị số như một tham số. Giá trị số này nhận giá trị trong khoảng từ 128 tới 256.

```
' The method in the controller application that calls ExecuteCommand method
Public Sub RunCommand()
    If Date.Now.Hour <= 12 Then
        MyServiceController.ExecuteCommand(200)
    ElseIf Date.Now.Hour > 12 Then
        MyServiceController.ExecuteCommand(220)
    End If
End Sub
```

Visual C#

```
void RunCommand()
{
    if(System.DateTime.Now.Hour <= 12 )
        MyServiceController.ExecuteCommand(200);
    else
        if(System.DateTime.Now.Hour <= 24 )
            MyServiceController.ExecuteCommand(220);
}
```

Các mã sau đây cho thấy làm thế nào để ghi đè lên các phương thức OnCustomCommand. Trong phương thức này, bạn có thể sử dụng If Then để thực hiện các nhiệm vụ khác nhau, tùy thuộc vào giá trị của tham số truyền theo phương thức RunCommand.

Visual Basic .NET

```
Protected Overrides Sub OnCustomCommand(ByVal command As Integer)
    Dim FS As New FileStream("C:\Temp\MyWindowsService_VB.txt", _
        FileMode.Append, FileAccess.Write)
    Dim SR As New StreamWriter(FS)
    If command = 200 Then
        SR.WriteLine("Good Morning")
        SR.Flush()
    ElseIf command = 220 Then
        SR.WriteLine("Good Evening")
        SR.Flush()
    End If
End Sub
```

Visual C#

```
protected override void OnCustomCommand(int command)
{
    FileStream FS = new FileStream(@"c:\Temp\MyWindowsService_CSharp.txt",
        FileMode.OpenOrCreate, FileAccess.Write);
    StreamWriter SR = new StreamWriter(FS);
    if(command == 200)
    {
        SR.WriteLine("Good Morning");
```

```
        SR.Flush();
    }
    else
    {
        if (command==220)
        {
            SR.WriteLine("Good Evening");
            SR.Flush();
        }
    }
}
```

Các ví dụ của các ServiceController cũng cho phép bạn để lấy một danh sách các dịch vụ đang chạy trên một máy tính bằng cách sử dụng phương thức GetServices. Các mã sau đây cho thấy làm thế nào để lấy danh sách các dịch vụ đang chạy trên một máy tính.

Visual Basic .NET

```
Imports System.ServiceProcess


Module Module1


    Sub Main()
```

```
        Dim MyServiceController As New ServiceController()
        Dim services As ServiceController()
        services = MyServiceController.GetServices()
        Dim enumerator As IEnumerator = services.GetEnumerator
        While enumerator.MoveNext
            Console.WriteLine(CType(enumerator.Current,  _
                ServiceController).ServiceName)
        End While
        Console.WriteLine("Press the <Enter> key to exit")
        Console.Read()
End Sub
End Module
```

Visual C#

```
using System;
using System.ServiceProcess;
using System.Collections;
namespace ConsoleApplication5
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
```

```
class Class1
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
        ServiceController MyServiceController = new
            ServiceController();
        ServiceController[] services= ServiceController.GetServices();
        IEnumerator enumerator   = services.GetEnumerator();
        while (enumerator.MoveNext())
        {
            Console.WriteLine(
                ((ServiceController)enumerator.Current).ServiceName);
        }
        Console.WriteLine("Press <Enter> to exit");
        Console.Read();
    }
}
```

```
}
```

Sau khi bạn tạo một dịch vụ Windows, bạn có thể cần phải cấu hình ứng dụng dịch vụ của bạn để cài đặt nó trên một máy tính. Bạn cũng có thể cần phải gỡ lỗi dịch vụ của bạn một khi nó được cài đặt. Bạn sẽ học cách cấu hình và gỡ lỗi các ứng dụng dịch vụ của bạn trong phần tiếp theo.

1.1.6. Cấu hình và gỡ lỗi các dịch vụ của Windows

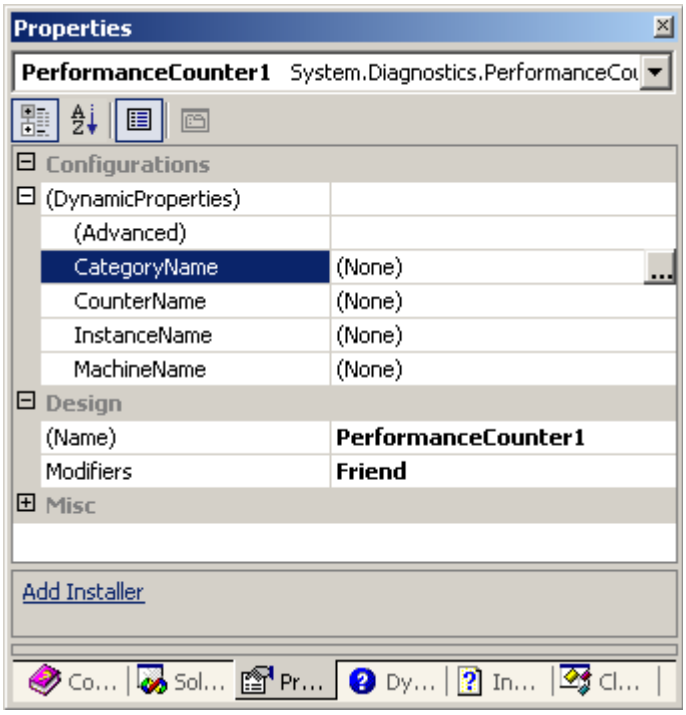
Sau khi bạn tạo ra và cài đặt một dịch vụ Windows, bạn có thể thay đổi các thuộc tính của nó tự động. NET Framework cung cấp tính năng động cho các dịch vụ Windows cho phép bạn thay đổi các thiết lập như AutoLog, CanShutdown, và CanPauseAndContinue mà không cần biên dịch lại mã. Ngoài ra, bạn có thể gỡ lỗi một dịch vụ Windows để đảm bảo rằng nó thực hiện tất cả các công việc mà bạn muốn ứng dụng dịch vụ để thực hiện.

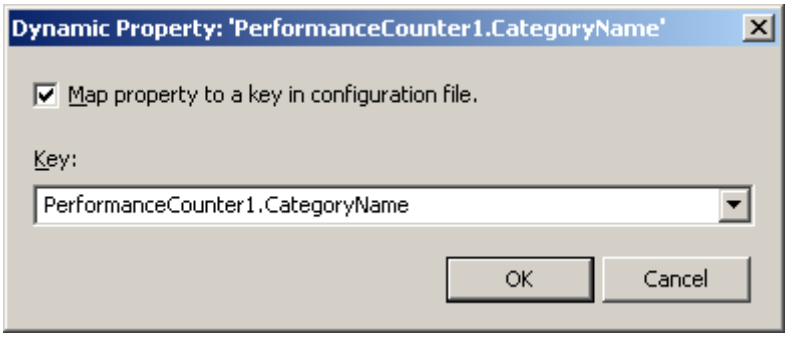
Cấu hình một dịch vụ Windows

Các dịch vụ Windows có thể sử dụng các tài nguyên như bản ghi sự kiện, Performance Counter, và các kết nối cơ sở dữ liệu. Bạn viết mã lệnh trong các Class của dịch vụ để truy cập vào các nguồn tài nguyên này. Nếu bạn muốn thay đổi bản ghi sự kiện Event logg, hiệu suất truy cập Performance Counter, hoặc database server mà dịch vụ của bạn sử dụng, bạn cần phải thay đổi mã và biên dịch lại các ứng dụng dịch vụ. Ngoài ra, bạn cần phải gỡ bỏ cài đặt các dịch vụ và cài đặt phiên bản mới của ứng dụng dịch vụ. Vì vậy, nhiệm vụ của việc duy trì các dịch vụ trở nên khó khăn nếu sự phụ thuộc của một dịch vụ trên tài nguyên bên ngoài thay đổi thường xuyên.

NET Framework cung cấp các file cấu hình mà làm cho việc duy trì dịch vụ là một nhiệm vụ đơn giản. Các tập tin cấu hình chứa các thuộc tính thiết đặt(property settings), cho phép bạn tự động cấu hình một dịch vụ mà không cần biên dịch lại ứng dụng dịch vụ. Mỗi ứng dụng dịch vụ có một tập tin cấu hình. Bằng cách thay đổi các thiết lập thuộc tính trong tập tin cấu hình, bạn có thể cấu hình lại các ứng dụng dịch vụ. Ví dụ, bằng cách thay đổi các thuộc tính kết nối cơ sở dữ liệu trong các tập tin cấu hình, bạn có thể cho phép ứng dụng của bạn để kết nối với một cơ sở dữ liệu khác nhau. Bạn có thể xác định thuộc tính

có thể được tự động cấu hình bằng cách sử dụng cửa sổ Properties lúc thiết kế. Các hình ảnh sau đây cho thấy làm thế nào bạn có thể thiết lập tính tự động(dynamic) cho một hiệu suất truy cập.





Khi bạn cấu hình tự động, NET Studio Visual thêm mã để ứng dụng dịch vụ của bạn, cho phép các ứng dụng dịch vụ để đọc các giá trị của các thuộc tính được định nghĩa trong file cấu hình tại thời gian chạy. Visual Studio NET thêm vào các mã sau khi bạn xác định tính tự động cho ứng dụng dịch vụ của bạn.

Visual Basic .NET

```
Dim configurationAppSettings As System.Configuration.AppSettingsReader =
New _
    System.Configuration.AppSettingsReader()


Me.PerformanceCounter1.CategoryName = CType( _
    configurationAppSettings.GetValue( "PerformanceCounter1.CategoryName", _
    GetType(System.String)), String)


Me.PerformanceCounter1.CounterName = CType( _
    configurationAppSettings.GetValue("PerformanceCounter1.CounterName", _
    GetType(System.String)),String)
```

Visual C#

```
System.Configuration.AppSettingsReader configurationAppSettings = new
    System.Configuration.AppSettingsReader();


this.performanceCounter1.CategoryName = ((string)
    (configurationAppSettings.GetValue("performanceCounter1.CategoryName",
    typeof(string))));


this.performanceCounter1.CounterName = ((string)
```

```
(configurationAppSettings.GetValue("performanceCounter1.CounterName",
typeof(string))));
```

Khi bạn cấu hình tự động, NET Studio Visual tạo ra một tập tin cấu hình có chứa cặp giá trị cho các thuộc tính mà bạn có thể cấu hình tự động. Các mã sau đây cho thấy các tập tin cấu hình được tạo ra khi bạn cấu hình CategoryName và tài sản CounterName của một hiệu suất truy cập Performance Counter tự động.

```
XML
<?xml version="1.0" encoding="Windows-1252"?>
<configuration>
  <appSettings>
    <!-- User application and configured property settings go here.-->
    <!-- Example: <add key="settingName" value="settingValue"/> -->
    <add key="PerformanceCounter1.CategoryName" value=".NET CLR Jit" />
    <add key="PerformanceCounter1.CounterName" value="Total # of IL Bytes
      Jitted" />
  </appSettings>
</configuration>
```

Sau khi xác định các tính chất tự động của một ứng dụng dịch vụ, bạn chỉ cần thay đổi giá trị của key thích hợp trong file cấu hình để thay đổi giá trị của các thuộc tính động.

Gỡ lỗi một dịch vụ Windows

Để gỡ lỗi một dịch vụ Windows, bạn bắt đầu các dịch vụ và sau đó đính kèm một trình gỡ lỗi cho quá trình mà trong đó các dịch vụ đang chạy. Bạn không thể gỡ lỗi một dịch vụ từ bên trong NET Studio Visual bằng cách nhấn phím F5 và F11 hoặc bằng cách sử dụng menu Debug vì hai lý do sau.

Đầu tiên, bạn cần phải thêm một trình cài đặt và cài đặt một ứng dụng dịch vụ trước khi một dịch vụ có thể chạy.

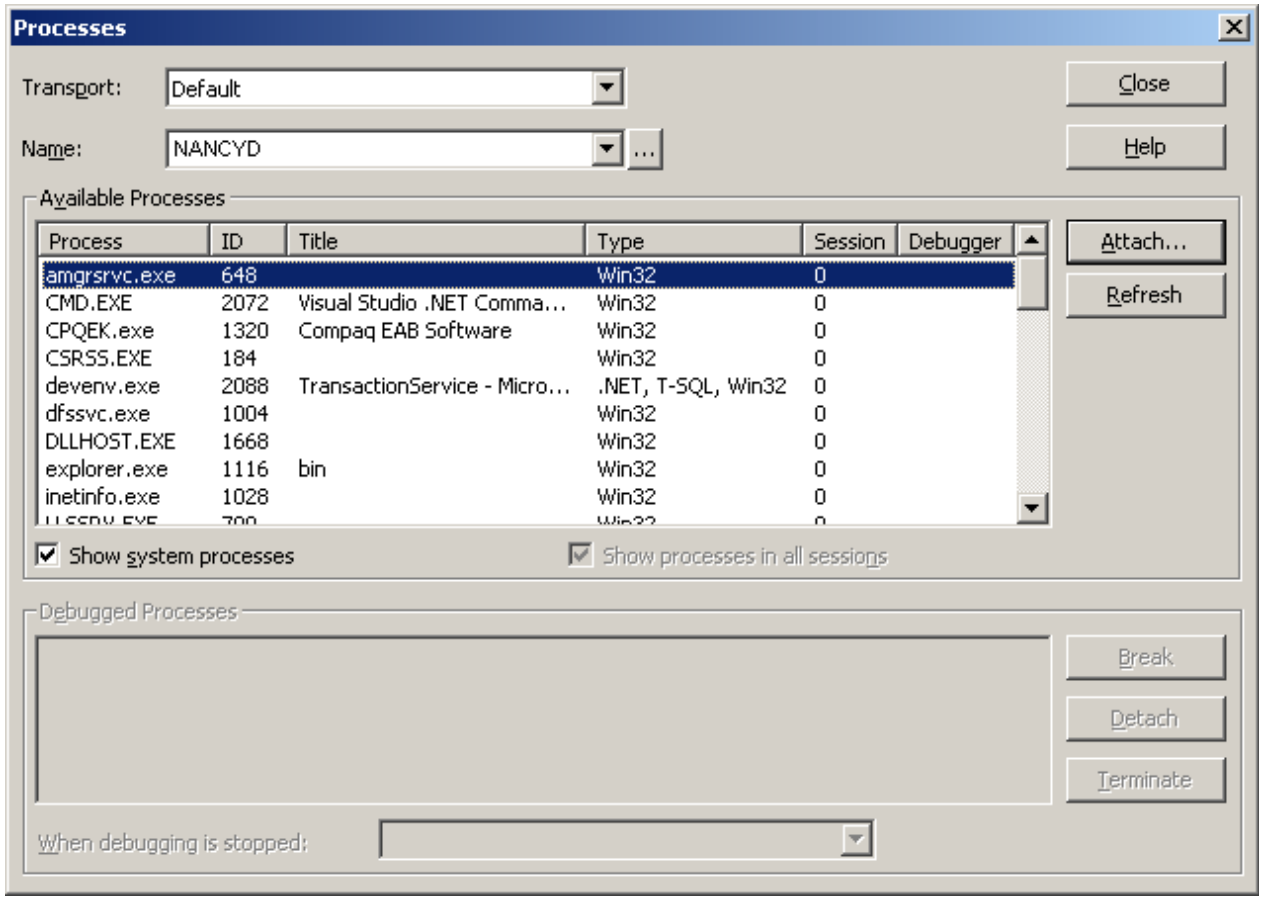
Thứ hai, một dịch vụ chạy trong một chế độ bảo mật khác nhau hơn so với của Visual Studio NET.

Sau khi bạn đính kèm một trình gỡ lỗi cho một quá trình, bạn có thể sử dụng tất cả các tính năng gỡ lỗi tiêu chuẩn, chẳng hạn như thiết lập cá ***breakpoints or stepping*** vào Visual Studio.NET cung cấp để gỡ lỗi một ứng dụng. Để đính kèm một trình gỡ lỗi cho một tiến trình Process, hoàn thành các bước sau:

1-Bắt đầu dịch vụ của bạn bằng cách sử dụng SCM.

2- Chọn Process từ menu Debug. Hộp thoại tiến trình (process) xuất hiện.

3-Chọn hiển thị các tiến trình hệ thống. Hình 2,18 hiển thị hộp thoại quá trình.

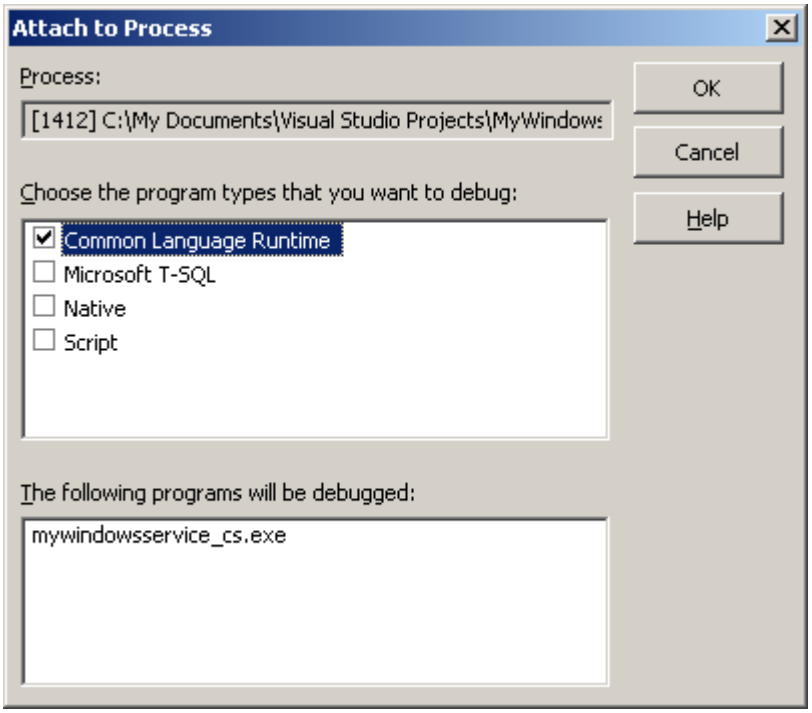


Hình 2-18. The Processes dialog box

4-Chọn các tiến trình cho các ứng dụng dịch vụ của bạn và nhấp vào đính kèm(Attacth). Attacth To Process hộp thoại xuất hiện.

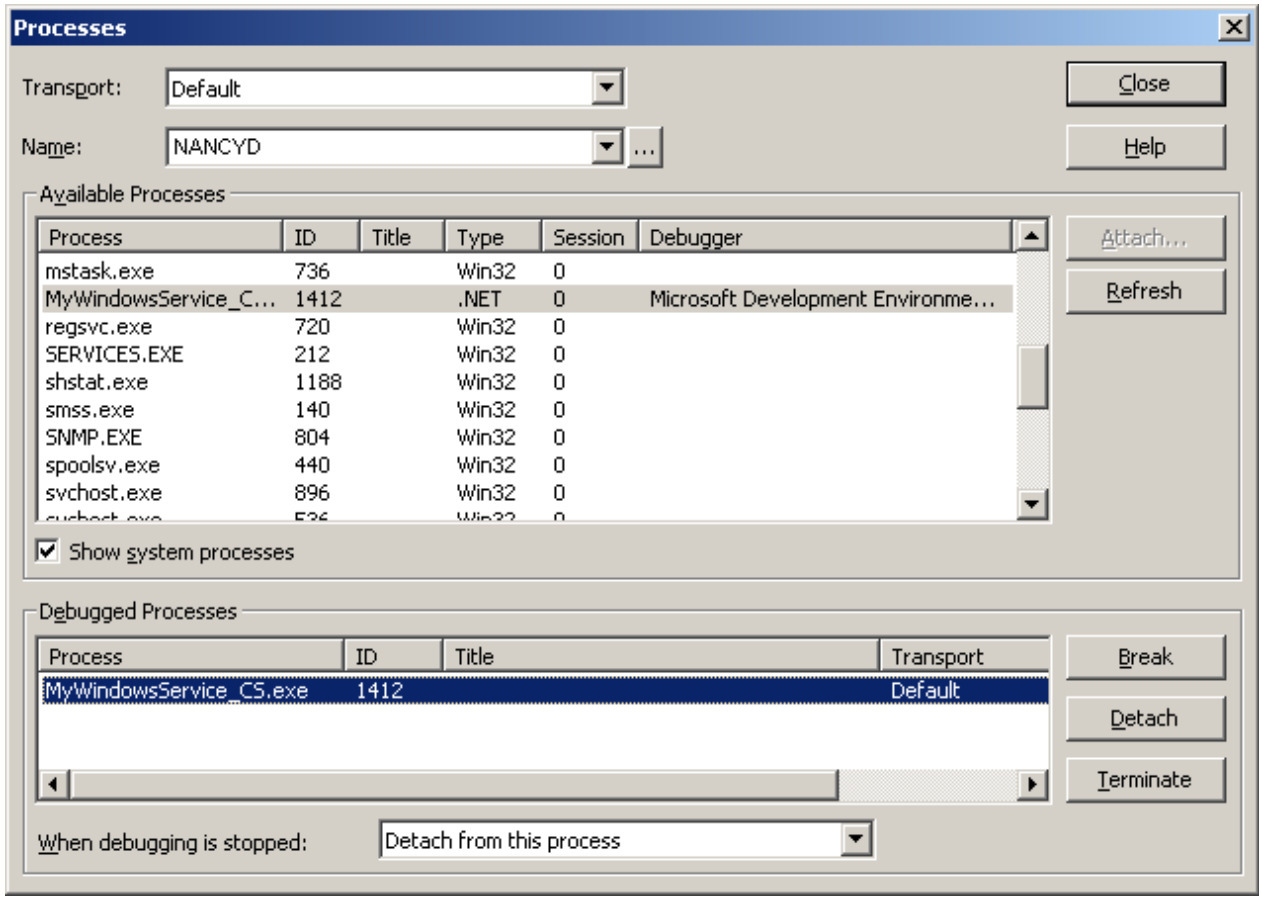
5-Chọn Common Language Runtime, nhấn OK để xác định một trình gỡ lỗi, và đóng cửa sổ ”Attach To Process dialog box” lại.

Hình 2,19 hiển thị hợp thoại ” Attach To Process dialog box”.



Hình 2-19. The Attach To Process dialog box

Sau khi bạn đóng của sổ ” Attach To Process dialog box”, ứng dụng dịch vụ của bạn vào chế độ gỡ lỗi, và môi trường Microsoft phát triển gắn liền với ứng dụng dịch vụ của bạn như các trình gỡ lỗi. Khi bạn đính kèm một trình gỡ lỗi vào một ứng dụng dịch vụ, trình gỡ lỗi sẽ phân tích và dịch vụ đang chạy trong process. Hộp thoại quá trình hiển thị gỡ rối kèm theo cho các ứng dụng dịch vụ thoại khác nhau. Hình 2,20 hiển thị hộp thoại quá trình.



Hình 2-20. The Processes dialog box after you have attached to a process

Sau đó, bạn có thể chèn các điểm ngắt BreackPoint trong mã của ứng dụng dịch vụ của bạn. Để gỡ lỗi các phương thức khác nhau trong mã của bạn, bạn có thể sử dụng SCM để gửi các lệnh Stop, Pause, và Continue tơi dịch vụ của bạn.

Khi bạn đính kèm một trình gỡ lỗi một ứng dụng dịch vụ, bạn chỉ có thể gỡ lỗi onPause, OnContinue, và phương thức OnStop. Bạn không thể gỡ lỗi các phương thức OnStart và chính bởi vì dịch vụ bắt đầu trước khi bạn đính kèm một trình gỡ lỗi để ứng dụng dịch vụ của bạn. Để gỡ lỗi các phương pháp OnStart và Main, bạn có thể tạo ra một dịch vụ khai thác thử nghiệm trong ứng dụng dịch vụ của bạn để giúp gỡ bỏ dịch vụ của bạn. Bạn có thể cài đặt cả hai dịch vụ, và sau đó bắt đầu dịch vụ khai thác thử nghiệm để tải các tiến trình dịch vụ có chứa cả hai dịch vụ. Sau khi dịch vụ khai thác thử nghiệm bắt đầu

quá trình này, bạn có thể sử dụng trình đơn Debug trong Visual Studio NET để đính kèm một trình gỡ lỗi để theo dõi tiến trình dịch vụ.

Tóm tắt nội dung của bài

Các dịch vụ Windows cho phép bạn thực hiện các nhiệm vụ thực hiện trong tiến trình nền tảng. Bạn có thể sử dụng dịch vụ Windows để thực hiện các nhiệm vụ như giám sát việc sử dụng một cơ sở dữ liệu. Một dịch vụ Windows thực hiện trong tiến trình riêng của nó và không có một giao diện người dùng. Bạn sử dụng các phương thức của ServiceBase, ServiceProcessInstaller, ServiceInstaller, và các lớp ServiceController trong không gian tên System.ServiceProcess để tạo ra một ứng dụng dịch vụ Windows.

Để tạo ra một dịch vụ Windows, bạn cần phải tạo ra một dự án bằng cách sử dụng dịch vụ Windows mẫu trong Visual Basic NET hay Visual C #. Các mẫu dịch vụ Windows sẽ tự động thêm mã phương thức Main của ứng dụng dịch vụ của bạn và chèn Function cho các phương thức OnStart và OnStop. Bạn ghi đè lên các phương thức OnStart và OnStop thêm các Function cho ứng dụng dịch vụ của bạn. Ngoài ra, bạn có thể ghi đè lên các phương thức onPause và OnContinue, và thay đổi các thuộc tính của các ứng dụng dịch vụ để tùy chỉnh các dịch vụ theo nhu cầu của bạn.

Bạn có thể tùy chỉnh dịch vụ Windows của bạn để xử lý các sự kiện cho một dịch vụ. Bạn cũng có thể cho phép các ứng dụng dịch vụ của bạn để đăng nhập thông tin, chẳng hạn như một sự thay đổi trong trạng thái của dịch vụ hay sai sót trong các bản ghi sự kiện hệ thống. Ngoài ra, bạn có thể tạo ra một bản ghi sự

kiện tùy chỉnh để ghi lại thông tin về dịch vụ của bạn. Những tính năng của dịch vụ Windows cho phép bạn xác định ứng dụng dịch vụ của bạn sẽ chạy như thế nào.

Để bắt đầu một dịch vụ Windows trên một máy tính, trước tiên bạn cần phải cài đặt các ứng dụng dịch vụ. Để làm như vậy, bạn cần phải thêm các trình cài đặt các ứng dụng dịch vụ và sử dụng các công cụ installutil để cài đặt các ứng dụng dịch vụ. Trình cài đặt, cài đặt và đăng ký dịch vụ của bạn trên máy tính. Ngoài ra, trình cài đặt, cài đặt và cấu hình các nguồn lực(tài nguyên liên quan), chẳng hạn như bộ đếm hiệu suất performance counter và các bản ghi sự kiện event log, vào dịch vụ của bạn để sử dụng. Trình cài đặt cũng cho phép bạn cấu hình bối cảnh an ninh(chế độ bảo mật) của các ứng dụng dịch vụ của bạn.

Sau khi bạn tạo ra và cài đặt một dịch vụ Windows, bạn có thể theo dõi và kiểm soát hành vi của các dịch vụ của bạn. Ví dụ, bạn có thể thay đổi trạng thái của các dịch vụ của bạn. Bạn cũng có thể quản lý dịch vụ của bạn theo cách thủ công bằng cách sử dụng SCM hoặc lập trình bằng cách sử dụng một thành phần của lớp ServiceController trong một ứng dụng.

Sau khi bạn tạo ra và cài đặt một dịch vụ Windows, bạn có thể gỡ lỗi một dịch vụ Windows để đảm bảo rằng nó thực hiện tất cả các công việc mà bạn muốn ứng dụng dịch vụ để thực hiện. Để gỡ lỗi một ứng dụng dịch vụ, bạn bắt đầu một dịch vụ và đính kèm một trình gỡ lỗi nó. Ngoài ra, bạn có thể cấu hình các thuộc tính nhất định của một ứng dụng dịch vụ tự động. Các tính năng động của

các ứng dụng dịch vụ này cho phép bạn thay đổi các thiết lập, chẳng hạn như AutoLog, CanShutdown, và CanPauseAndContinue.

1.2.Trình tự thao tác

Tạo và quản lý các dịch vụ của Windows

Trong phần này sẽ hướng dẫn, bạn sẽ tạo và cài đặt các dịch vụ Windows. Ngoài ra, bạn sẽ tạo ra các ứng dụng của client để giao tiếp với các ứng dụng dịch vụ. Bạn cũng sẽ tạo ra một ứng dụng Windows để quản lý các dịch vụ đang chạy trên một máy tính local hoặc từ xa. Các giải pháp cho các bài tập trong phòng thực hành cần có đĩa CD windows hệ thống, để thuận tiện việc cài thêm và backup dữ liệu.

A. Creating and Installing a Windows Service

Trong bài tập này, bạn sẽ tạo ra và cài đặt một dịch vụ Windows. Các dịch vụ Windows, gọi tới TransactionService, bao gồm các dịch vụ của Windows gọi là DBWriter. Khi các dịch vụ DBWriter bắt đầu, nó tạo ra hai tập tin: Transaction.tmp và Customers.db. Các ứng dụng client viết hồ sơ Client vào tập tin Transaction.tmp. ứng dụng Client có thể yêu cầu dịch vụ DBWriter cam kết hoặc quay trở lại (commit or roll back) các hồ sơ Client bằng văn bản trong các tập tin Transaction.tmp. Khi Client yêu cầu ứng dụng để thực hiện các hồ sơ, dịch vụ DBWriter viết hồ sơ Client từ file Transaction.tmp tới các tập tin Customers.db. Sau khi viết các bản ghi trong các tập tin Customers.db, DBWriter loại bỏ tất cả các bản ghi từ các tập tin Transaction.tmp. Tuy nhiên, nếu Client yêu cầu ứng dụng để quay trở lại tất cả các hồ sơ, DBWriter loại bỏ tất cả các bản ghi từ các tập tin Transaction.tmp mà không cần lưu chúng trong các tập tin Customers.db.

Các dịch vụ DBWriter sử dụng nhật ký sự kiện tùy chỉnh được gọi là Transaction Log để viết vào mục này ở các giai đoạn thực hiện khác nhau. Các dịch vụ DBWriter cũng sử dụng một hiệu suất truy cập tùy chỉnh(performance counter) được gọi Ctr1 trong các loại MyCounters theo dõi số lượng các giao dịch đã cam kết. Khi các dịch vụ DBWriter dừng lại, nó sẽ xóa tập tin Transaction.tmp và đặt giá trị Ctr1 là 0.

Để tạo Windows service DBWriter trong ứng dụng TransactionService, hoàn thành các bước sau:

1- Vào Start/ Visual Studio. NET và mở cửa sổ New Project từ trình đơn File.

2- Chọn dự án Visual Basic hoặc C # Visual từ lựa chọn các loại dự án.

3- Chọn mẫu dịch vụ Windows (windows service template).

4- Đặt tên là TransactionService trong trường Name.

5- Thay đổi tên tập tin của Service1.vb hoặc Service.cs thành DBWriter.vb hoặc DBWriter.cs.

6- Trong Solution Explorer, kích chuột phải vào DBWriter.vb hoặc DBWriter.cs, và chọn View Designer để mở cửa sổ thiết kế.

7- Kích chuột phải vào bất cứ nơi nào trong cửa sổ thiết kế, và chọn Properties từ menu chuột phải.

8- Thay đổi thuộc tính ServiceName và Name của các dịch vụ từ Service1 thành DBWriter. Thiết lập thuộc tính AutoLog thành False.

9- Kích chuột phải vào dự án TransactionService trong Solution Explorer, và chọn Properties. Đặt Startup Object là DBWriter.

Lưu ý:

Điều này được yêu cầu cho dự án Visual Basic NET.

10- Chọn View code từ menu chuột phải bằng cách nhấp chuột phải trong cửa sổ thiết kế (Component Designer)để xem mã lệnh của lớp dịch vụ DBWriter. Bạn sẽ thấy đoạn mã sau trong phương thức Main.

Visual Basic .NET

```
ServicesToRun = New System.ServiceProcess.ServiceBase () {New Service1}
```

Visual C#

```
ServicesToRun = new System.ServiceProcess.ServiceBase[] { new Service1() };
```

11. Bằng cách tìm kiếm bạn tìm và thay thế Service1 thành DBWriter.

12. Kéo và thả Event Log vào trong của sổ thiết kế

13. Đặt giá trị cho thuộc tính Log là **Transaction log** và thuộc tính Source là **Transaction Service**

14. Kích chuột phải vào nút Performance Counters dưới nút Server trong Server Explorer và chọn Create New Category từ menu chuột phải. Trong cửa sổ Performance Counter Builder, viết MyCounters trong hộp văn bản của mục Name. Nhấn vào New và thay đổi CounterName thành Ctr1. Hãy để các loại vẫn là mặc định của NumberOfItems32. Nhấn OK để tạo ra các loại MyCounters và truy Ctr1 Counter.

15. Kéo và thả các Ctr1 Counter vào của sổ thiết kế. Thay đổi thuộc tính MachineName thành . (Dot)-(Không đặt gì mà ghi vào dấu ".") và thuộc tính ReadOnly là False.

16. Thêm mã lệnh sau đây lớp DBWriter.

Visual Basic .NET

```
Protected Overrides Sub OnStart(ByVal args() As String)
    ' Create a transaction file and close it
    Dim tfs As FileStream = File.Create("C:\Transaction.tmp")
    tfs.Close()
    ' Create Customers.db file if it does not exist
    If Not (File.Exists("C:\Customers.db")) Then
        Dim cfs As FileStream = File.Create("C:\Customers.db")
        cfs.Close()
    End If
    ' Write entry into the custom event log
    EventLog1.WriteEntry("DBWriterVB service started on " + _
        Date.Now.ToString)
End Sub

Protected Overrides Sub OnStop()
    ' Delete the Transaction.tmp file
    File.Delete("C:\Transaction.tmp")
    ' Set the performance counter value to 0
    PerformanceCounter1.RawValue = 0
    ' Write entry into the custom event log
    EventLog1.WriteEntry("DBWriterVB service stopped on " + _
```

```
        Date.Now.ToString)
    End Sub


    Protected Overrides Sub OnCustomCommand(ByVal command As Integer)
        If command = 201 Then
            Commit()
        ElseIf command = 200 Then
            Rollback()
        End If
    End Sub


    Private Sub Commit()
        ' Create a StreamReader to read data from the Transaction.tmp file
        Dim sr As New StreamReader(New FileStream("C:\Transaction.tmp", _
            FileMode.Open))
        ' Create a StreamWriter to append data to the Customers.db file
        Dim sw As New StreamWriter(New FileStream("C:\Customers.db", _
            FileMode.Append, FileAccess.Write))
        sw.WriteLine(sr.ReadToEnd)
        sw.Flush()
        ' Close the files
```

```
        sr.Close()
        sw.Close()
        ' Increment the counter and write entry in the EventLog
        PerformanceCounter1.Increment()
        truncateTransactionFile()
        EventLog1.WriteEntry("DBWriterVB service committed a " & _
            "transaction on " + Date.Now.ToString)
    End Sub

    Private Sub Rollback()
        truncateTransactionFile()
        EventLog1.WriteEntry("DBWriterVB service rolled back a " & _
            "transaction on " + Date.Now.ToString)
    End Sub

    Private Sub truncateTransactionFile()
        ' Delete data from the Transaction.tmp file
        Dim fs As New FileStream("C:\Transaction.tmp", FileMode.Truncate)
        fs.Flush()
        fs.Close()
    End Sub
```

Visual C#

```
protected override void OnStart(string[] args)
{
   // Create a transaction file and close it
   FileStream tfs = File.Create("C:\\Transaction.tmp");
   tfs.Close();
   // Create Customers.db file if it does not exist
   if  (!File.Exists("C:\\Customers.db"))
   {
       FileStream cfs= File.Create("C:\\Customers.db");
       cfs.Close();
   }
   // Write entry into the custom event log
   eventLog1.WriteEntry("DBWriterCS service started on "
      + System.DateTime.Now.ToString());
}

protected override void OnStop()
{
```

```
    // Delete the Transaction.tmp file
    File.Delete("C:\\Transaction.tmp");
    // Set the performance couter value to 0
    performanceCounter1.RawValue = 0;
    // Write entry into the custom event log
    eventLog1.WriteEntry("DBWriterCS service stopped on "
        + System.DateTime.Now.ToString());
}

protected override void OnCustomCommand(int command)
{
    if (command == 201)
        Commit();
    else if (command == 200)
        Rollback();
}

private void Commit()
{
    // Create a StreamReader to read data from the Transaction.tmp file
    StreamReader sr = new StreamReader(new FileStream
```

```
        ("C:\\Transaction.tmp",
       FileMode.Open));
    // Create a StreamWriter to append data to the Customers.db file
                    StreamWriter sw = new StreamWriter(new
FileStream("C:\\Customers.db",
        FileMode.Append, FileAccess.Write));
    sw.WriteLine(sr.ReadToEnd());
    sw.Flush();
    //Close the files
    sr.Close();
    sw.Close();
    //Increment the counter and write entry in the EventLog
    performanceCounter1.Increment();
    truncateTransactionFile();
    eventLog1.WriteEntry("DBWriterCS service committed a transaction on "
       + System.DateTime.Now.ToString());
}

private void Rollback()
{
    truncateTransactionFile();
    eventLog1.WriteEntry("DBWriterCS service rolled back a
```

```
            transaction on "
            + System.DateTime.Now.ToString());
        }

        private void truncateTransactionFile()
        {
            // Delete data from the Transaction.tmp file
            FileStream fs =new FileStream("C:\\Transaction.tmp",
                FileMode.Truncate);
            fs.Flush();
            fs.Close();
}
```

17. Thêm cài đặt cho các sự kiện đăng nhập Event log, hiệu suất truy cập Performance Counter , và Windows service. Chọn Add Installer từ menu chuột phải bằng cách nhấn phải trên các thành phần hay Add Installer ở phía dưới cùng của cửa sổ Properties của các thành phần.

18. Trong Component Designer hiển thị ProjectInstaller và the EventLog-Installer, PerformanceCounterInstaller, ServiceProcessInstaller, và Service-Installer \ được bổ sung vào ProjectInstaller.

19. Thay đổi thuộc tính Account của ServiceProcessInstaller để là LocalSystem.

20. Build ứng dụng để tạo ra File TransactionService.exe assembly.

21. Cài đặt ứng dụng bằng lệnh trên Visual Studio .NET Command Prompt:

Installutil TransactionService.exe

22. Xác minh rằng dịch vụ DBWriter được cài đặt đúng, và Start nó bằng cách sử dụng SCM. Mở Event Viewer và xác minh rằng các bản ghi giao dịch đã được cài đặt đúng. Xác minh rằng dịch vụ DBWriter viết vào các mục chính xác trong nhật ký giao dịch Transaction Log.

23. Xác minh rằng dịch vụ DBWriter tạo ra các tập tin C: \ Customers.db và C: \ Transaction.tmp. Xác minh rằng các tập tin Transaction.tmp bị xóa khi bạn ngừng dịch vụ DBWriter.

B. Tạo một ứng dụng Client

Trong bài tập này, bạn sẽ tạo ra một ứng dụng Windows để viết hồ sơ trong file Transaction.tmp và thực hiện các lệnh tùy chỉnh trên các dịch vụ DBWriter.

1) Tạo một New Project window trong File menu.

2) Lựa chọn Windows Application trong mục Templates pane.

3) Đặt tên TransactionClient trong Name field.

4) Đổi tên file Form1.vb or Form1.cs thành CustomerForm.vb hoặc CustomerForm.cs trong Solution Explorer ngay dưới TransactionClient project.

5) Mở CustomerForm trong Component Designer.

6) Thay đổi thuộc tính Name thành CustomerForm và thuộc tính Text thành Customer Form.

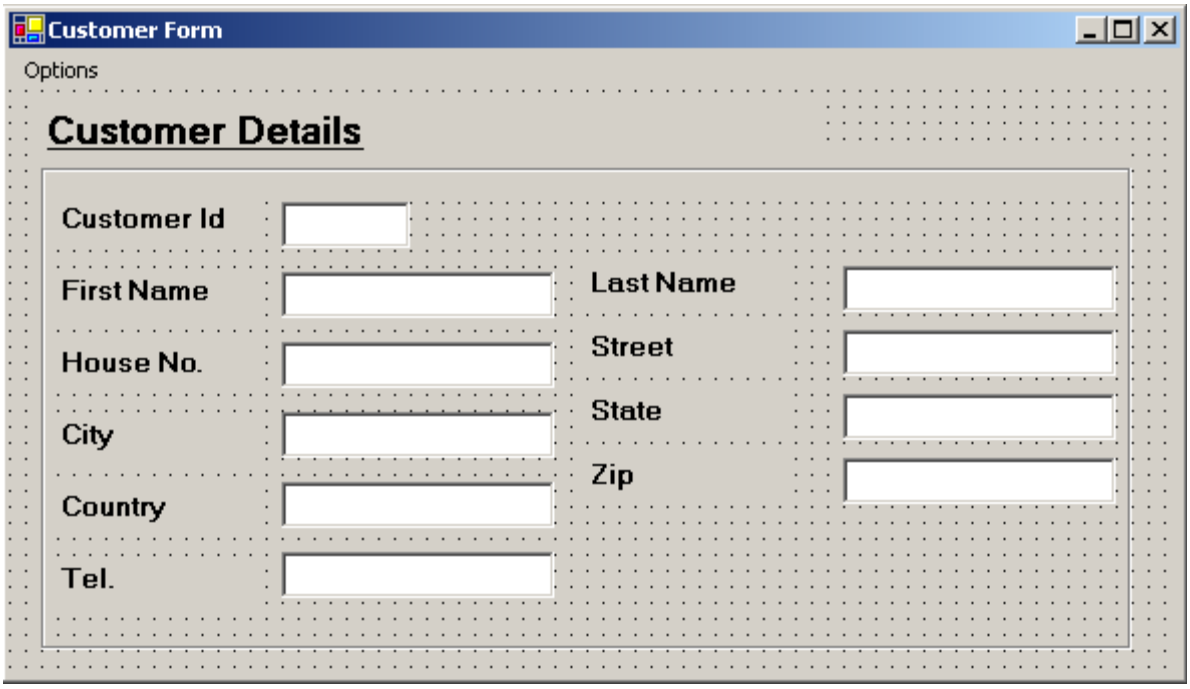
7) Mở cửa sổ Properties cho dự án TransactionClient từ menu chuột phải bằng cách nhấp chuột phải vào tên dự án trong Solution Explorer và thiết lập Startup Object as CustomerForm.

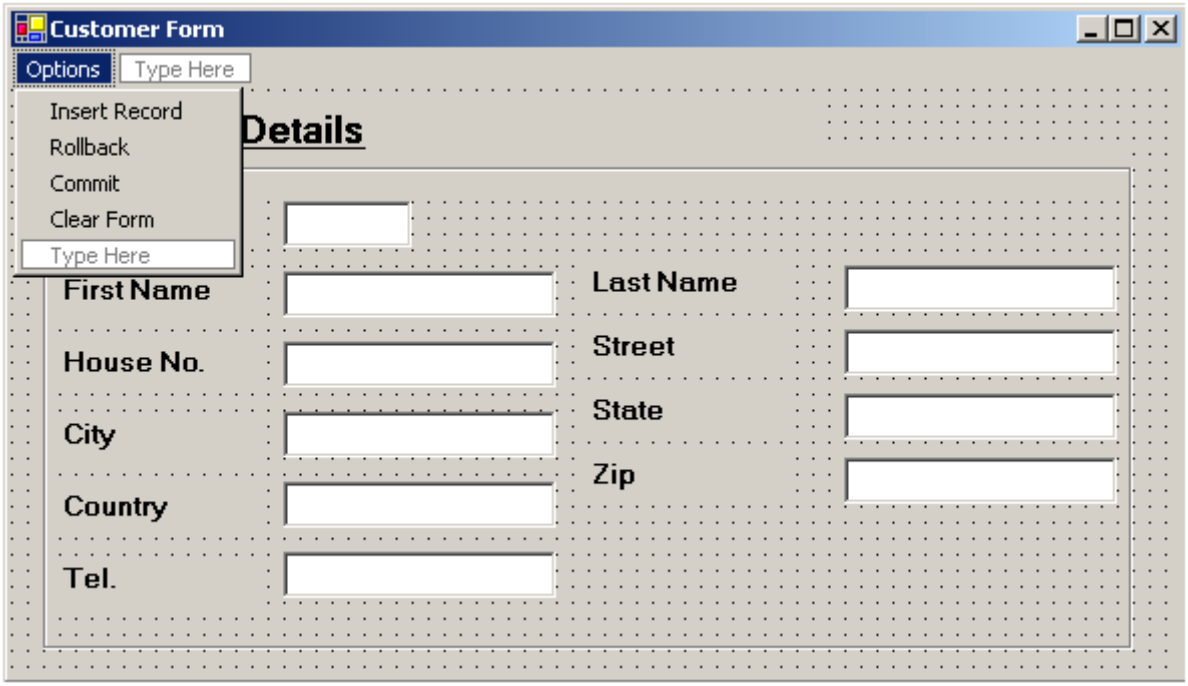
Lưu ý

Yêu cầu này chỉ yêu cầu riêng cho Visual Basic.NET

8)Kích chuột phải vào Reference dưới dự án TransactionClient và chọn Add Reference. Trong cửa sổ Add Reference, di chuyển xuống để chọn System.ServiceProcess.dll trên tab NET. Nhấp vào nút Select. System.ServiceProcess.dll xuất hiện trong các danh sách Components lựa chọn. Click vào OK để thêm một tham chiếu đến nó.

9)Thêm Lable, Textbox, và main menu, và thêm các mục menu con trên CustomerForm, và thiết lập các thuộc tính Text và Font để giao diện xuất hiện như thể hiện trong hình sau đây.





10) Thiết lập thuộc tính Name của các textbox như sua CustId, FName, LName, HouseNo, Street, City, State, Country, Zip, and Tel. Thiết lập thuộc tính Text của các TextBox để trống.

11) Thiết lập thuộc tính Name của các mục trình đơn menu thành InsertRecordMenuItem, RollbackMenuItem, CommitMenuItem, và ClearFormMenuItem.

12) Thêm mã sau đây lớp CustomerForm.

Visual Basic .NET

```
Imports System.IO

Imports System.ServiceProcess



Public Class CustomerForm
```

```
    Inherits System.Windows.Forms.Form

' Component Designer Code goes here
.
.
.

Private Sub InsertRecordMenuItem_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles InsertRecordMenuItem.Click
    Dim record As String
    record = "CustomerId: " + CustId.Text + "  FName: " + FName.Text + _
             "  LName: " + LName.Text + "  House No.: " + HouseNo.Text + _
             "  Street: " + Street.Text + "  City: " + City.Text + _
             "  State: " + State.Text + "  Country: " + Country.Text + _
             "  Zip: " + Zip.Text + "  Tel.: " + Tel.Text

    'Write the record in the C:\Transaction.tmp file
    Dim sw As New StreamWriter(New FileStream("C:\Transaction.tmp", _
       FileMode.Append, FileAccess.Write))
    sw.WriteLine(record)
    sw.Flush()
```

```
    sw.Close()
End Sub


Private Sub ClearFormMenuItem_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ClearFormMenuItem.Click
    clear()
End Sub



Private Sub RollbackMenuItem_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles RollbackMenuItem.Click
    ' Roll back all the changes made in the Transaction.tmp file
    Dim sc As New ServiceController("DBWriter")
    sc.ExecuteCommand(200)
    clear()
End Sub



Private Sub CommitMenuItem_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles CommitMenuItem.Click
    ' Commit all the changes made in the Transaction.tmp file
```

```
    Dim sc As New ServiceController("DBWriter")
    sc.ExecuteCommand(201)
End Sub

Private Sub clear()
    CustId.Text = ""
    FName.Text = ""
    LName.Text = ""
    HouseNo.Text = ""
    Street.Text = ""
    City.Text = ""
    State.Text = ""
    Country.Text = ""
    Zip.Text = ""
    Tel.Text = ""
End Sub

End Class
```

Visual C#

```
using System.Drawing;
using System.Collections;
```

```
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.ServiceProcess;


public class CustomerForm : System.Windows.Forms.Form
{


   // Component Designer Code goes here
   .
   .
   .


   private void InsertMenuItem_Click(object sender,
      System.EventArgs e)
   {
      String record;
      record = "CustomerId: " + CustId.Text + "  FName: " +
         FName.Text + "  LName: " + LName.Text +
         "  House No.: " + HouseNo.Text + "  Street: " +
```

```
        Street.Text + "  City: " + City.Text + "  State: " +
        State.Text + "  Country: " + Country.Text + "  Zip: " +
        Zip.Text + "  Tel.: " + Tel.Text;

    //Write the record in the C:\Tranasaction.tmp file
    StreamWriter sw =new StreamWriter(new FileStream(
    "C:\\Transaction.tmp", FileMode.Append, FileAccess.Write));
    sw.WriteLine(record);
    sw.Flush();
    sw.Close();
}

private void RollbackMenuItem_Click(object sender,
     System.EventArgs e)
{
    // Roll back all the changes made in the Transaction.tmp file
    ServiceController sc = new ServiceController("DBWriterCS");
    sc.ExecuteCommand(200);
    clear();
}
```

```
private void CommitMenuItem_Click(object sender,
     System.EventArgs e)
{
    // Commit all the changes made in the Transaction.tmp file
    ServiceController sc = new ServiceController("DBWriterCS");
    sc.ExecuteCommand(201);
}

private void ClearFormMenuItem_Click(object sender,
     System.EventArgs e)
{
    clear();
}

private void clear()
{
    CustId.Text = "";
    FName.Text = "";
    LName.Text = "";
    HouseNo.Text = "";
    Street.Text = "";
```

```
        City.Text = "";
        State.Text = "";
        Country.Text = "";
        Zip.Text = "";
        Tel.Text = "";
    }
}
```

13) Chọn Buid Solution từ menu Build

14) Chạy CustomerForm. Hãy chắc chắn rằng bạn chạy các dịch vụ DBWriter trước khi chạy CustomerForm.

15. Tạo hồ sơ Client và chèn chúng trong các tập tin Transaction.tmp bằng cách chọn Insert từ menu Record. Xem kết quả trong các tập tin Transaction.tmp. Xác minh rằng khi commit văn bản sẽ ghi vào tập tin Customers.db.

16. Mở Performance Monitor. Thêm Ctr1 Counter từ MyCounters Category. Ctr1sẽ tăng với số lượng giao dịch.

17. Mở Event Viewer để xem dịch vụ DBWriter viết trong nhật ký giao dịch vào mục Transaction Log.

C.Quản lý dịch vụ windows

Trong bài tập này, bạn sẽ tạo một ứng dụng Windows để quản lý các dịch vụ Windows trên một máy tính.

1)Mở cửa sổ New Project từ trình đơn File.

2)Chọn Windows ứng dụng từ cửa sổ Templates.

3)Đặt tên ServiceController trong trường Name.

4)Thay đổi Form1.vb hoặc Form1.cs đến ServiceControllerForm.vb ServiceControllerForm.cs trong Solution Explorer cho ServiceController project.

5)Mở ServiceControllerForm trong phần thiết kế.

6)Thay đổi thuộc tính Name thành ServiceControllerForm và thuộc tính Text thành Service Controller.

7) Mở cửa sổ Properties cho dự án ServiceController từ menu chuột phải trong Solution Explorer và thiết lập Startup Object ServiceControllerForm.

Lưu ý

Yêu cầu này chỉ áp dụng cho úng dụng Visual Basic.NET

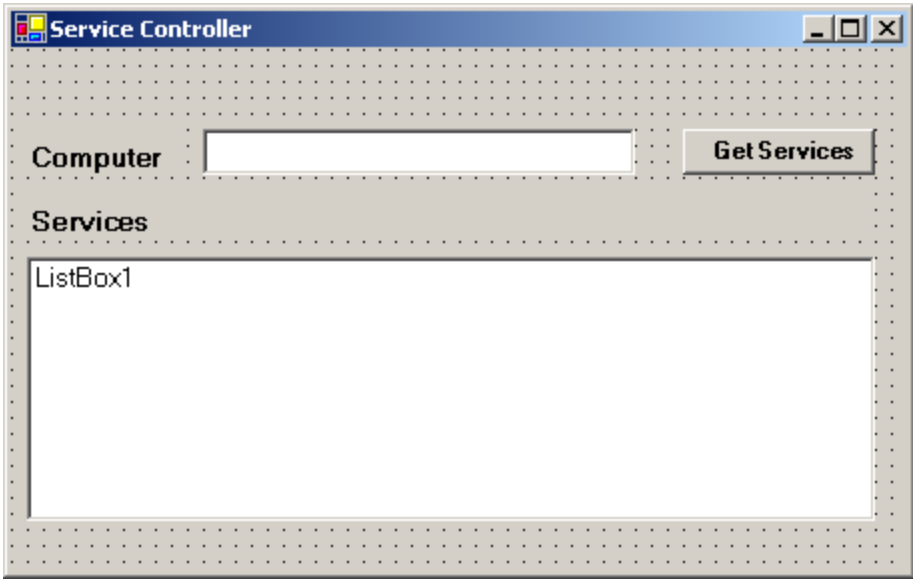
8)Kích chuột phải vào Reference ngay dưới ServiceController project và chọn Add Reference.

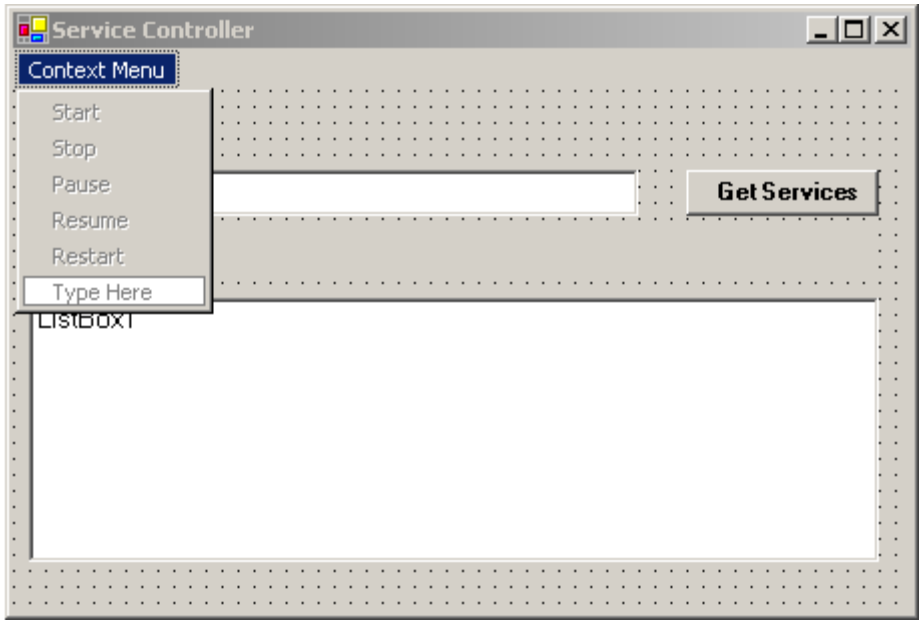
9)Trong cửa sổ Add Reference, chọn System.ServiceProcess.dll vào tab NET.

10)Nhấp vào Chọn. System.ServiceProcess.dll xuất hiện trong các danh sách Selected Components.

11)Click vào OK để thêm một tham chiếu đến nó.

12)Thêm Lable, TextBox, Button, ListBox, Context Menu, và các mục menu vào ServiceControllerForm, và thiết lập các thuộc tính Text và Font để giao diện xuất hiện như hiển thị trong hình sau đây





13)Thiết lập thuộc tính Name của các mục trong trình đơn menu như sau StartMenu, StopMenu, PauseMenu, ResumeMenu, và RestartMenu, tương ứng với

nhãn của menu. Thiết lập thuộc tính Enabled của tất cả các mục trình đơn thành False.

14)Thiết lập thuộc tính Text của nút thành Get Services và thuộc tính Name thành GetServices_Button.

15)Viết đoạn code sau đây cho lớp ServiceControllerForm.

Visual Basic .NET

```
Imports System.ServiceProcess
Public Class ServiceControllerForm
    Inherits System.Windows.Forms.Form

    Private svcs As ServiceProcess.ServiceController()
    Private svcs_enum As Ienumerator

    ' Component Designer generated Code
    .
    .
    .

    Private Sub ServiceControllerForm_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        ListBox1.ContextMenu = ContextMenu1
```

```
End Sub


Private Sub GetServices_Button_Click(ByVal sender As _
    System.Object, ByVal e As System.EventArgs) Handles _
    GetServices_Button.Click
    ListBox1.Items.Clear()
    If TextBox1.Text.Equals("") Then
        svcs = ServiceProcess.ServiceController.GetServices()
        Label2.Text = "Services on the local machine"
    Else
        Try
            svcs = ServiceProcess.ServiceController.GetServices _
                (TextBox1.Text)
            Label2.Text = "Services on " + TextBox1.Text
        Catch
            MsgBox("Unable to find the computer!")
            TextBox1.Text = ""
            Exit Sub
        End Try
    End If
```

```
    svcs_enum = svcs.GetEnumerator
    Dim sc As ServiceProcess.ServiceController

    While svcs_enum.MoveNext
        sc = CType(svcs_enum.Current, ServiceProcess.ServiceController)
        ListBox1.Items.Add(sc.ServiceName)
    End While
End Sub

Private Sub StartMenu_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles StartMenu.Click
    Dim sc As ServiceProcess.ServiceController
    sc = svcs.GetValue(ListBox1.SelectedIndex)
    enable_disable_MenuItems(sc)
    Try
        sc.Start()
        sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Running)
        MsgBox("Service " + sc.ServiceName + " started on " + _
            sc.MachineName + " !")
    Catch
        MsgBox("Unable to start " + sc.ServiceName + " service!")
```

```
    End Try
End Sub

Private Sub StopMenu_Click(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles StopMenu.Click
    Dim sc As ServiceProcess.ServiceController
    sc = svcs.GetValue(ListBox1.SelectedIndex)
    enable_disable_MenuItems(sc)
    Try
        sc.Stop()
        sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Stopped)
        MsgBox("Service " + sc.ServiceName + " stopped on " + _
            sc.MachineName + " !")
    Catch
        MsgBox("Unable to stop " + sc.ServiceName + " service!")
    End Try
End Sub

Private Sub PauseMenu_Click(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles PauseMenu.Click
    Dim sc As ServiceProcess.ServiceController
```

```
    sc = svcs.GetValue(ListBox1.SelectedIndex)
    enable_disable_MenuItems(sc)
    Try
        sc.Pause()
        sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Paused)
        MsgBox("Service " + sc.ServiceName + " paused on " + _
            sc.MachineName + " !")
    Catch
        MsgBox("Unable to pause " + sc.ServiceName + " service!")
    End Try
End Sub

Private Sub ResumeMenu_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ResumeMenu.Click
    Dim sc As ServiceProcess.ServiceController
    sc = svcs.GetValue(ListBox1.SelectedIndex)
    enable_disable_MenuItems(sc)
    Try
        sc.Continue()
        sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Running)
        MsgBox("Service " + sc.ServiceName + " resumed on " + _
```

```
            sc.MachineName + " !")
        Catch
            MsgBox("Unable to resume " + sc.ServiceName + " service!")
        End Try
    End Sub

    Private Sub RestartMenu_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles RestartMenu.Click
        Dim sc As ServiceProcess.ServiceController
        sc = svcs.GetValue(ListBox1.SelectedIndex)
        enable_disable_MenuItems(sc)
        Try
            sc.Stop()
            sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Stopped)
            sc.Start()
            sc.WaitForStatus(ServiceProcess.ServiceControllerStatus.Running)
            MsgBox("Service " + sc.ServiceName + " restarted on " + _
                sc.MachineName + " !")
        Catch
            MsgBox("Unable to restart " + sc.ServiceName + " service!")
        End Try
```

```
End Sub


Private Sub ListBox1_MouseDown(ByVal sender As Object, ByVal e As _
    System.Windows.Forms.MouseEventArgs) Handles ListBox1.MouseDown
    If ListBox1.Items.Count <> 0 Then
        Dim top_index As Integer = ListBox1.TopIndex()
    Dim sel_index As Integer = top_index + System.Math.Floor(e.Y / _
        ListBox1.ItemHeight)
        ListBox1.SetSelected(sel_index, True)
    End If
End Sub


Private Sub enable_disable_MenuItems(ByVal sc As _
    ServiceProcess.ServiceController)
    If sc.Status = ServiceProcess.ServiceControllerStatus.Stopped Then
        Me.StartMenu.Enabled = True
        Me.StopMenu.Enabled = False
        Me.PauseMenu.Enabled = False
        Me.ResumeMenu.Enabled = False
        Me.RestartMenu.Enabled = False
    ElseIf sc.Status = _
```

```
        ServiceProcess.ServiceControllerStatus.Paused Then
        Me.StartMenu.Enabled = False
        Me.StopMenu.Enabled = True
        Me.PauseMenu.Enabled = False
        Me.ResumeMenu.Enabled = True
        Me.RestartMenu.Enabled = True
    ElseIf sc.Status = _
        ServiceProcess.ServiceControllerStatus.Running Then
        Me.StartMenu.Enabled = False
        Me.StopMenu.Enabled = True
        If sc.CanPauseAndContinue Then
            Me.PauseMenu.Enabled = True
            Me.ResumeMenu.Enabled = False
        End If
        Me.RestartMenu.Enabled = True
    End If
End Sub

Private Sub ContextMenu1_Popup(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles ContextMenu1.Popup
    Dim sc As ServiceProcess.ServiceController
```

```
        sc = svcs.GetValue(ListBox1.SelectedIndex)
        enable_disable_MenuItems(sc)
    End Sub
End Class
```

Visual C#

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.ServiceProcess;


namespace ServiceController
public class ServiceControllerForm : System.Windows.Forms.Form
{
    private void GetServices_Button_Click(object sender,
        System.EventArgs e)
    {
        ListBox1.Items.Clear();
        if (TextBox1.Text.Equals(""))
```

```
{
    svcs = ServiceController.GetServices();
    Label2.Text = "Services on the local machine";
}
else
{
    try
    {
        svcs = ServiceController.GetServices(TextBox1.Text);
        Label2.Text = "Services on " + TextBox1.Text;
    }
    catch
    {
        MessageBox.Show("Unable to find the computer!");
        TextBox1.Text = "";
        return;
    }
}
svcs_enum = svcs.GetEnumerator();
ServiceController sc ;
```

```
    while (svcs_enum.MoveNext())
    {
        sc = (ServiceController)svcs_enum.Current;
        ListBox1.Items.Add(sc.ServiceName);
    }
}

private void StartMenu_Click(object sender, System.EventArgs e)
{
    ServiceController sc;
    sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
    enable_disable_MenuItems(sc);
    try
    {
        sc.Start();
        sc.WaitForStatus(ServiceControllerStatus.Running);
        MessageBox.Show("Service " + sc.ServiceName + " started on "
            + sc.MachineName + " !");
    }
    catch
    {
```

```
        MessageBox.Show("Unable to start " + sc.ServiceName +
        " service!");
    }
}

private void StopMenu_Click(object sender, System.EventArgs e)
{
    ServiceController sc;
    sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
    enable_disable_MenuItems(sc);
    try
    {
        sc.Stop();
        sc.WaitForStatus(ServiceControllerStatus.Stopped);
        MessageBox.Show("Service " + sc.ServiceName + " stopped on " +
            sc.MachineName + " !");
    }
    catch
    {
        MessageBox.Show("Unable to stop " + sc.ServiceName +
            " service!");
```

```
    }
}

private void PauseMenu_Click(object sender, System.EventArgs e)
{
    ServiceController sc ;
    sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
    enable_disable_MenuItems(sc);
    try
    {
        sc.Pause();
        sc.WaitForStatus(ServiceControllerStatus.Paused);
        MessageBox.Show("Service " + sc.ServiceName + " paused on " +
            sc.MachineName + " !");
    }
    catch
    {
        MessageBox.Show("Unable to pause " + sc.ServiceName +
            " service!");
    }
}
```

```
private void ResumeMenu_Click(object sender, System.EventArgs e)
{
    ServiceController sc ;
    sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
    enable_disable_MenuItems(sc);
    try
    {
        sc.Continue();
        sc.WaitForStatus(ServiceControllerStatus.Running);
        MessageBox.Show("Service " + sc.ServiceName + " resumed on " +
            sc.MachineName + " !");
    }
    catch
    {
        MessageBox.Show("Unable to resume " + sc.ServiceName +
            " service!");
    }
}

private void RestartMenu_Click(object sender, System.EventArgs e)
```

```
{
    ServiceController sc ;
    sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
    enable_disable_MenuItems(sc);
    try
    {
        sc.Stop();
        sc.WaitForStatus(ServiceControllerStatus.Stopped);
        sc.Start();
        sc.WaitForStatus(ServiceControllerStatus.Running);
        MessageBox.Show("Service " + sc.ServiceName +
            " restarted on " + sc.MachineName + " !");
    }
    catch
    {
        MessageBox.Show("Unable to restart " + sc.ServiceName +
            " service!");
    }
}

private void ListBox1_MouseDown(object sender,
```

```
    System.Windows.Forms.MouseEventArgs  e)
{
    if (ListBox1.Items.Count != 0)
    {
        int top_index = ListBox1.TopIndex ;
        int sel_index = top_index + (int)System.Math.Floor(e.Y /
            ListBox1.ItemHeight);
        ListBox1.SetSelected(sel_index, true);
    }
}

private void enable_disable_MenuItems( ServiceController sc  )
{
    if (sc.Status == ServiceControllerStatus.Stopped)
    {
        this.StartMenu.Enabled = true;
        this.StopMenu.Enabled = false;
        this.PauseMenu.Enabled = false;
        this.ResumeMenu.Enabled = false;
        this.RestartMenu.Enabled = false;
    }
```

```
    else if (sc.Status == ServiceControllerStatus.Paused)
    {
        this.StartMenu.Enabled = false;
        this.StopMenu.Enabled = true;
        this.PauseMenu.Enabled = false;
        this.ResumeMenu.Enabled = true;
        this.RestartMenu.Enabled = true;
    }
    else if (sc.Status == ServiceControllerStatus.Running)
    {
        this.StartMenu.Enabled = false;
        this.StopMenu.Enabled = true;
        if (sc.CanPauseAndContinue)
        {
            this.PauseMenu.Enabled = true;
            this.ResumeMenu.Enabled = false;
        }
        this.RestartMenu.Enabled = true;
    }
}
```

```
    private void ContextMenu1_Popup(object sender, System.EventArgs e)
    {
      ServiceController sc ;
      sc = (ServiceController)svcs.GetValue(ListBox1.SelectedIndex);
      enable_disable_MenuItems(sc);
    }

    private void ServiceControllerForm_Load(object sender,
        System.EventArgs e)
    {
        this.ContextMenu=this.ContextMenu1;
    }
}
```

16)Chọn Build Solution trong menu Build.

17)Chạy ứng dụng. Nhấp vào Get Services để có được danh sách các dịch vụ trên máy tính local. Sử dụng trình đơn phím tắt của ứng dụng để quản lý các dịch vụ.

18)Gõ tên (không phải là địa chỉ IP) của một Network Computer và nhấp vào Get Services để có được danh sách các dịch vụ trên máy tính từ xa đó. Quản lý các dịch vụ bằng cách sử dụng menu chuột phải trên ListBox.

III. Tóm tắt trình tự thực hiện hoặc quy trình công nghệ

STT	*Tên các bước công việc*	*Dụng cụ, thiết bị, vật tư*	*Yêu cầu kỹ thuật*	*Các chú ý về an toàn lao động*
1	Tạo ứng dụng dịch vụ	Visual Studio 2005 trở lên	-Lựa chọn ứng dụng dịch vụ windows server -Lựa chọn đúng ngôn ngữ lập trình - Làm đúng trình tự thực hiện - Giám sát các lỗi trong quá trình lập trình	
2	Cài đặt ứng dụng dịch vụ	Visual Studio 2005, đĩa CD windows.	-Cài đặt bằng command promt -Cài đặt bằng cách đóng gói project -Thiết lập các tham số khi cài đặt, tên server...	
3	Quản lý ứng dụng	Windows, visual studio.net	-Quản lý bằng SCM -Quản lý bằng một ứng dụng thứ 3	

Bài 2: Tạo và ứng dụng các dịch vụ Web XML

Dịch vụ Web XML mô hình lập trình cho phép bạn xây dựng khả năng mở rộng nâng ca, ứng dụng phân tán bằng cách sử dụng giao thức Web chuẩn như HTTP, XML, SOAP. Trong bài này bạn sẽ tìm hiểu về mô hình dịch vụ lập trình Web XML và tìm hiểu để tạo, triển khai, và ứng dụng các dịch vụ Web XML.

I.Mục tiêu

Học xong bài học này người đạt được những kỹ năng sau:

- Kỹ năng lập trình với ứng dụng dịch vụ Web XML
- Kỹ năng thao tác với các giao thức mạng HTTP,SOAP

II.Nội dung

2.1. Lý thuyết liên quan

2.1.1. Tìm hiểu XML Web Services

Tổng quan về các dịch vụ Web XML

Một dịch vụ Web XML là một thành phần thực hiện chương trình logic và cung cấp chức năng cho các ứng dụng khác nhau. Các ứng dụng này sử dụng các giao thức chuẩn, như HTTP, XML, SOAP, để truy cập vào các chức năng. Dịch vụ Web XML sử dụng dựa trên XML messaging để gửi và nhận dữ liệu, cho phép các ứng dụng không đồng nhất để tương thích với nhau. Bạn có thể sử dụng các dịch vụ Web XML để tích hợp các ứng dụng được viết bằng ngôn ngữ lập trình khác nhau và được triển khai trên các nền tảng khác nhau. Ngoài ra, bạn có thể triển khai các dịch vụ Web XML trong một mạng nội bộ cũng như trên Internet. Trong khi Internet mang lại cho người dùng gần gũi hơn với các tổ chức, các dịch vụ Web XML cho phép các tổ chức để tích hợp các ứng dụng của họ.

Một trong những tính năng quan trọng của mô hình tính toán Web XML servicesbased là một client không cần phải biết ngôn ngữ mà dịch vụ Web XML được thực hiện. Client chỉ cần biết vị trí của một dịch vụ Web XML và các phương thức mà Client có thể gọi vào dịch vụ.

Một số ví dụ sẽ giúp bạn hiểu được bối cảnh mà bạn có thể thực hiện các giải pháp XML Web servicesbased. Trong hình thức đơn giản của nó, một dịch vụ Web XML có thể bao gồm logic lập trình cụ thể để cung cấp chức năng, chẳng hạn như tính thuế thu nhập. Một dịch vụ Web XML tính thuế thu nhập đòi hỏi phải có một ứng dụng Client để cung cấp các thông tin như thu nhập, tiết kiệm, và các khoản đầu tư được thực hiện trong năm. ứng dụng Client có thể gọi một phương thức vào dịch vụ và cung cấp các thông tin cần thiết như các đối số cho phương thức gọi. Các dữ liệu liên quan đến các lời gọi phương thức và các đối số được gửi đến dịch vụ web ở định dạng XML bằng cách sử dụng giao thức SOAP trên HTTP vận chuyển. Một ví dụ khác mà bạn có thể sử dụng các dịch vụ Web XML là ứng dụng tích hợp. Bạn có thể kích hoạt một ứng dụng được viết trong một ngôn ngữ, như COBOL, gửi dữ liệu đến một thành phần được viết bằng một ngôn ngữ khác, chẳng hạn như Visual Basic, bằng cách sử dụng một dịch vụ Web XML.

Cơ sở hạ tầng dịch vụ Web XML

Một trong những tính năng quan trọng của mô hình tính toán Web XML servicesbased là cả Client và các dịch vụ Web XML là không biết gì về các chi tiết thực của nhau. XML Web servicesbased cung cấp một số thành phần cho

phép các ứng dụng của Client(Client) để định vị và lấy về các dịch vụ Web XML. Các thành phần này bao gồm những điều sau đây:

- ***XML Web Services directories.***

Những thư mục này cung cấp một vị trí trung tâm để lưu trữ thông tin về các dịch vụ Web XML. Những thư mục này cũng có thể là các dịch vụ Web XML cho phép bạn tìm kiếm các thông tin về các dịch vụ Web XML khác lập trình. Universal Description, Discovery, và Integration (UDDI) thông số kỹ thuật xác định các hướng dẫn về thông tin xuất bản về các dịch vụ Web XML. Các lược đồ XML kết hợp với UDDI định nghĩa bốn loại thông tin mà bạn phải xuất bản để làm cho dịch vụ Web XML của bạn có thể truy cập. Những thông tin này bao gồm thông tin kinh doanh, dịch vụ thông tin, thông tin ràng buộc, và thông số kỹ thuật dịch vụ. Microsoft cung cấp một dịch vụ thư mục như vậy, nằm ở http://uddi.microsoft.com.

- ***XML Web services discovery.***

Sử dụng quá trình này, Client tìm các tài liệu mô tả một dịch vụ Web XML bằng cách sử dụng WSDL. Quá trình phát hiện cho phép Client biết về sự hiện diện của một dịch vụ Web XML và về vị trí của một dịch vụ Web XML đặc biệt.

- ***XML Web services description.***

Thành phần này cung cấp thông tin cho phép bạn biết được hoạt động để thực hiện trên một dịch vụ Web XML. Mô tả dịch vụ Web XML một tài liệu

XML chỉ định định dạng của thông tin gửi mà một dịch vụ Web XML có thể hiểu được. Ví dụ, các tài liệu mô tả các lược đồ thông tin của SOAP mà bạn sử dụng khi gọi các phương thức trên một dịch vụ Web XML.

- ***XML Web service wire formats.***

Để cho phép giao tiếp giữa các hệ thống khác nhau, các dịch vụ Web XML sử dụng open wire format. ***Open wire format*** là các giao thức có thể được hiểu là bất kỳ hệ thống có khả năng hỗ trợ các tiêu chuẩn web phổ biến, chẳng hạn như HTTP và SOAP. HTTP-GET và giao thức HTTP-POST là các giao thức Web tiêu chuẩn cho phép bạn để gửi các thông số như các cặp tên-giá trị. Các giao thức HTTP-GET cho phép bạn gửi các thông số mã hóa URL như các cặp name-value với một dịch vụ Web XML. HTTP-GET giao thức đòi hỏi bạn phải gắn tham số cặp tên-giá trị (name-value) vào URL của dịch vụ Web XML. Bạn cũng có thể sử dụng giao thức HTTP POST URL mã hóa và truyền tham số cho các dịch vụ Web XML là cặp tên-giá trị(name-value). Tuy nhiên, các thông số được thông qua bên trong thông tin gửi đi yêu cầu tính thực tế và không gắn thêm vào URL của dịch vụ Web XML.

Các giao thức SOAP cho phép bạn trao đổi thông tin giữa các ứng dụng trên Internet có cấu trúc và đã ghi rõ các thông tin về nó. Các giao thức SOAP bao gồm bốn phần.

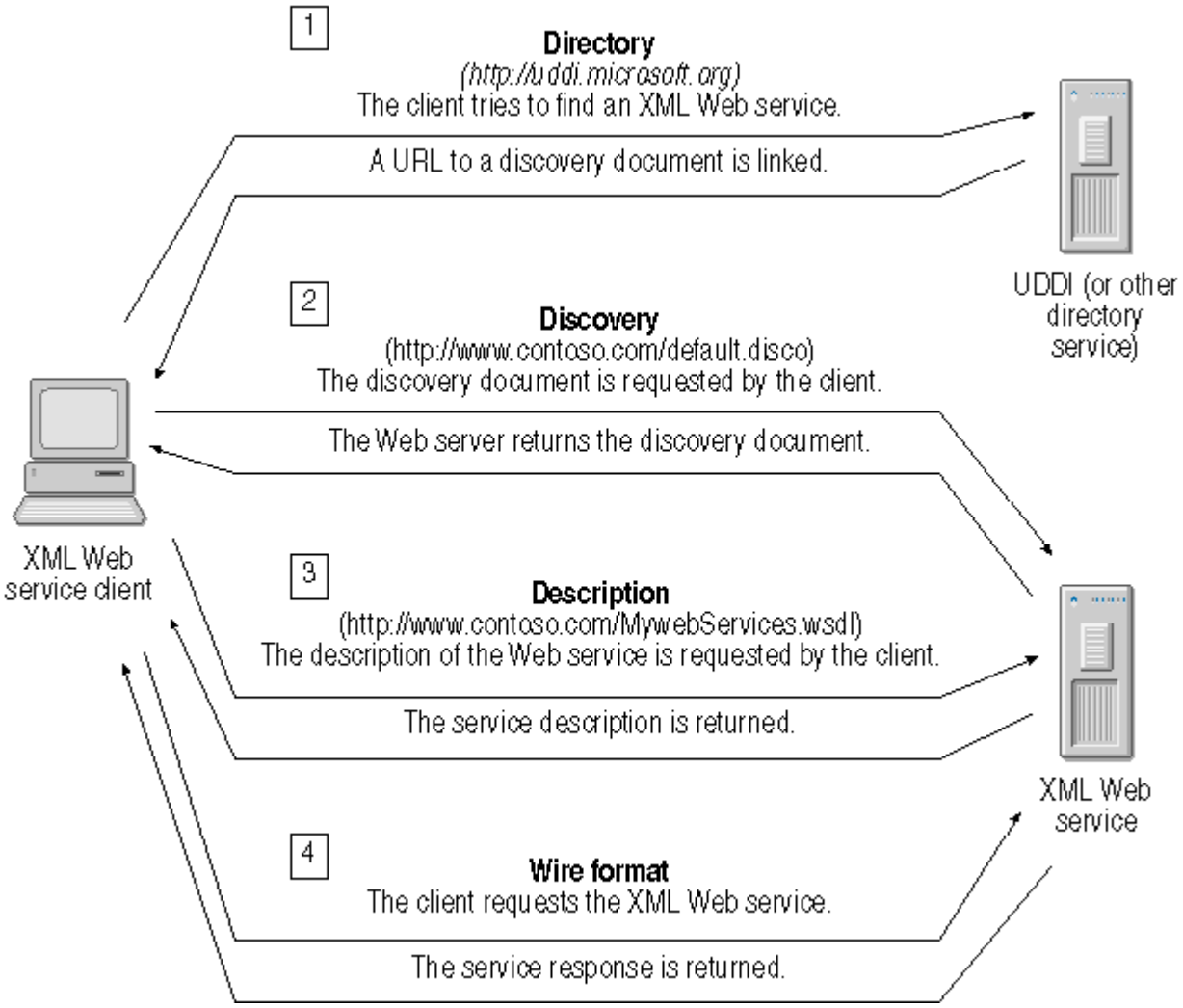
Phần đầu tiên là bắt buộc và định nghĩa envelope có chứa các message. Envelope SOAP là đơn vị cơ bản của trao đổi giữa các bộ vi xử lý của thông điệp SOAP.

Phần thứ hai định nghĩa các quy tắc mã hóa dữ liệu tùy chọn mà bạn sử dụng để mã hóa các loại dữ liệu ứng dụng cụ thể.

Phần thứ ba xác định các mô hình yêu cầu / đáp ứng trao đổi thông tin giữa các dịch vụ Web XML.

Phần thứ tư, đó là tùy chọn, xác định các cam kết ràng buộc giữa các giao thức SOAP và HTTP.

Hình 7.1 cho thấy làm thế nào các thành phần khác nhau của cơ sở hạ tầng dịch vụ Web XML cho phép Client xác định vị trí và gọi tới các phương thức của Web XML service.



Hình 7-1. The components of the XML Web services infrastructure

Khi một Client truy cập một dịch vụ UDDI để xác định vị trí một dịch vụ Web XML, các dịch vụ UDDI trả về một URL gói tin khám phá (discovery document) của dịch vụ Web XML. Một tài liệu được phát hiện là một tập tin khám phá (Disco file), trong đó có các liên kết đến các nguồn lực mà mô tả một dịch vụ Web XML. Một tập tin phát hiện là một tài liệu XML cho phép phát hiện ra chương trình của một dịch vụ Web XML. Sau khi Client nhận được các URL từ các gói tin tài liệu khám phá (discovery document), Client yêu cầu một máy chủ, trong đó trả về các tài liệu khám phá (discovery document) cho client. Các nội dung của một tài Dicos (discovery document) có dạng mẫu được thể hiện trong các mã sau đây.

```
XML

<?xml version="1.0" ?>
<disco:discovery xmlns:disco="http://schemas.xmlsoap.org/disco"
   xmlns:wsdl="http://schemas.xmlsoap.org/disco/wsdl">
   <wsdl:contractRef ref="http://www.contoso.com/MyWebService.asmx?
WSDL"/>
</disco:discovery>
```

Client sử dụng các thông tin trong các tài liệu khám phá (discovery document) và yêu cầu một máy chủ về một mô tả dịch vụ của dịch vụ Web XML do client yêu cầu. Mô tả dịch vụ là một tập tin WSDL và cho phép một client tương tác với một dịch vụ Web XML.

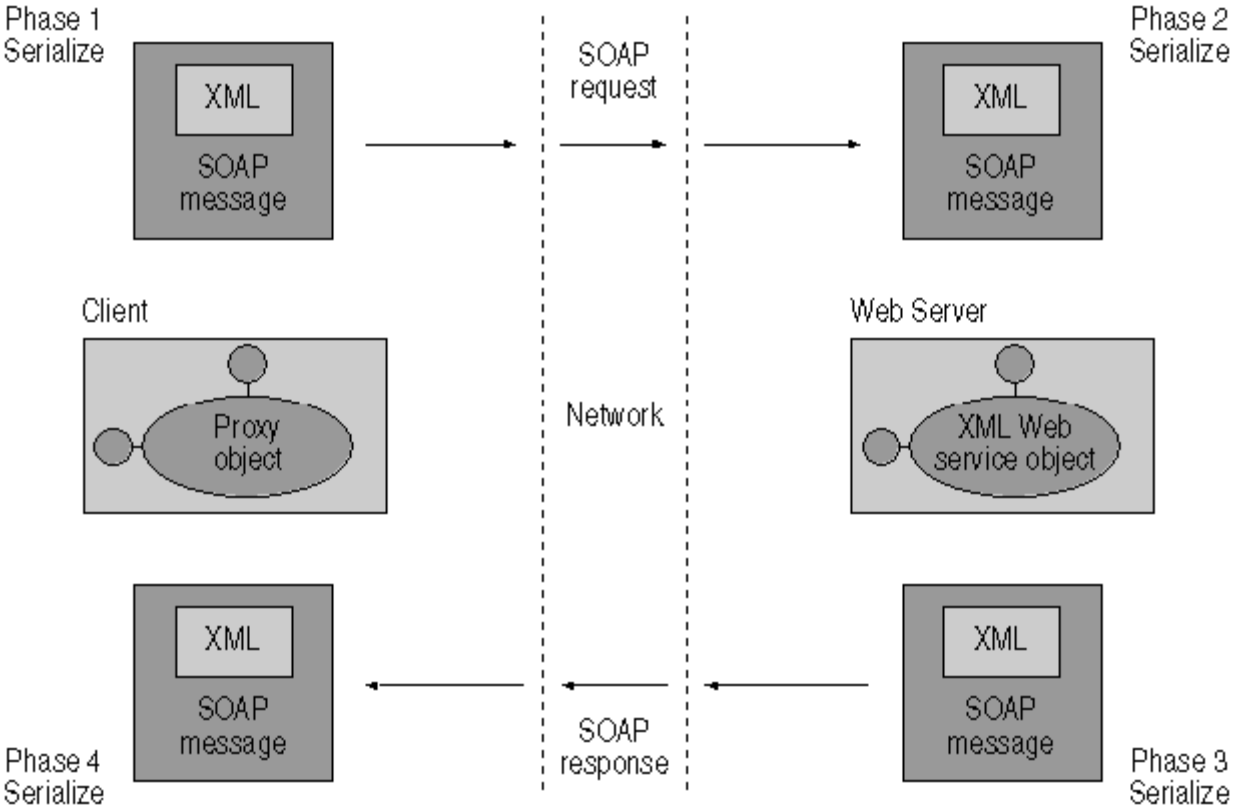
Bây giờ bạn sẽ xem xét quá trình tham gia khi một Client gọi tới phương thức trên một dịch vụ Web XML.

Giao tiếp giữa Client và các dịch vụ Web XML

Quá trình giao tiếp giữa một Client và một dịch vụ Web XML tương tự như một cuộc gọi thủ tục từ xa (RPC). Client sử dụng một đối tượng proxy của dịch vụ Web XML trên máy tính địa phương(local computer) để gọi các phương thức trên các dịch vụ Web XML.

Hình 7.2. hiển thị quá trình giao tiếp giữa một Client và một dịch vụ Web XML.

-



Hình 7.2. Client and XML Web service communication

Như thể hiện trong hình 7.2, sự tương tác giữa một Client và một dịch vụ Web XML bao gồm nhiều giai đoạn. Các nhiệm vụ sau đây được thực hiện trong các giai đoạn:

1) Client tạo ra một đối tượng của lớp XML Web service proxy trên cùng một máy tính mà Client cư trú.
2) Client gọi tới một phương trong Proxy object
3) Hệ thống và cấu trúc dịch vụ Web XML trên hệ thống Client đóng gói lời gọi phương thức và đối số vào một tin nhắn SOAP và gửi nó đến các dịch vụ Web XML qua mạng.

4) Hệ thống và cấu trúc trên máy chủ mà trên đó các dịch vụ Web XML thường trú cùng thông điệp SOAP và tạo ra một thể hiện của các dịch vụ Web XML. Cơ sở hạ tầng sau đó gọi phương thức với các đối số vào dịch vụ Web XML.

5) Các dịch vụ Web XML thực hiện các phương thức và trả về giá trị với bất kỳ truyền cho thông số tới hệ thống ứng dụng.

6) Hệ thống trả về và bất kỳ giá trị cho bất ký một thông số gửi vào một thông điệp tới SOAP và gửi chúng cho Client qua mạng.

7) Hệ thống ứng dụng trên máy tính của Client gửi đi thông điệp SOAP chứa giá trị trả về và các thông số bất kỳ nào được tạo ra và gửi chúng đến các đối tượng proxy.

8) Các đối tượng proxy sẽ gửi giá trị trả về và bất kỳ thông số được tạo ra cho client.

Để xây dựng các dịch vụ Web XML các Client có thể sử dụng được, bạn sử dụng cơ sở hạ tầng ASP.NET, đó là một phần của NET Framework. Visual Studio. NET cung cấp các công cụ để xây dựng, triển khai, và xuất bản các dịch vụ Web XML của bạn bằng cách sử dụng ASP.NET. Trong bài học tiếp theo, bạn sẽ tìm hiểu làm thế nào để tạo ra các dịch vụ Web XML bằng cách sử dụng Visual Studio NET.

2.1.2. Tạo XML Web Services

Trong bài trước, bạn đã học về các dịch vụ Web XML và các thành phần khác nhau của cơ sở hạ tầng dịch vụ Web XML. Bạn cũng đã học về kiến trúc của dịch vụ Web XML. Trong bài học này, bạn sẽ tìm hiểu làm thế nào để tạo ra các dịch vụ Web XML bằng cách sử dụng Visual Studio NET. Ngoài ra, bạn

sẽ tìm hiểu về các tập tin khác nhau được tạo ra khi bạn tạo ra một dịch vụ Web XML.

Tạo một dịch vụ Web XML

Tạo một dịch vụ Web XML tương tự như việc tạo ra một Component Object Model (COM) thành phần cung cấp logic ứng dụng cho một ứng dụng của Client. Bạn tạo ra một dịch vụ Web XML cung cấp chức năng cụ thể cho các ứng dụng của Client thông qua web. Để tạo ra một dịch vụ Web XML, trước tiên bạn chọn một ngôn ngữ lập trình mà bạn muốn để tạo ra các dịch vụ và tạo ra một lớp kế thừa từ System.Web.Services.WebService. Tiếp theo, bạn cần một cơ sở hạ tầng làm cho các dịch vụ Web XML có sẵn cho các ứng Client sử dụng trên Internet. Visual Studio. NET cung cấp cho bạn với các công cụ giúp bạn xây dựng, triển khai, và sử dụng một dịch vụ Web XML.

Bây giờ bạn sẽ tìm hiểu làm thế nào để tạo ra một dịch vụ Web XML. Trong bài học này, bạn sẽ có một dịch vụ du lịch mẫu và xử lý nó thông qua các giai đoạn khác nhau trong vòng đời của một dịch vụ Web XML. Dịch vụ Web XML mẫu này cho phép người dùng tạo các cuộc hẹn của họ trong một ứng dụng Scheduler và gửi e-mail thông báo cho những người mà họ cần phục vụ. Để cung cấp chức năng này vào các ứng dụng của Client, bạn cần phải tạo ra một dịch vụ Web XML để tạo ra các cuộc hẹn và gửi thông báo bằng e-mail.

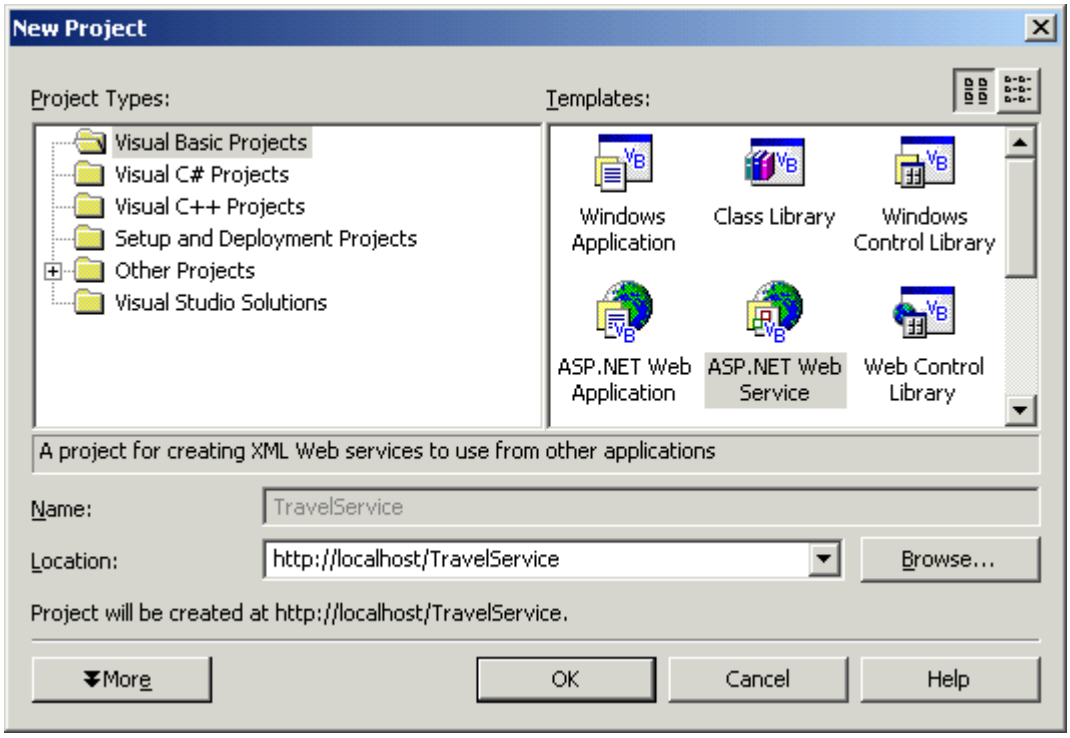
Dịch vụ Web XML TravelService, tuy nhiên, không phải là giao diện người dùng điều khiển hay tương tác. Các dịch vụ Web XML cung cấp ứng dụng logic và được tích hợp với một ứng dụng Scheduler. ứng dụng Scheduler cho phép người sử dụng để tạo ra và duy trì các cuộc hẹn. Lịch trình ứng dụng đi du lịch đặt phòng dựa trên các chi tiết bổ nhiệm được cung cấp bởi người sử dụng. Các ứng dụng Scheduler hoạt động như các ứng dụng cho các dịch vụ Web XML và có thể là một ứng dụng Windows hoặc một ứng dụng Web

ASP.NET. Bạn sẽ học cách tạo ra một ứng dụng Client trong các bài học tiếp theo.

Studio Visual IDE cung cấp cho bạn với môi trường cần thiết để tạo ra các dự án để phân phối các ứng dụng máy tính để bàn, các ứng dụng Web, và các dịch vụ Web XML. Để tạo ra một dịch vụ Web XML, thực hiện theo các bước sau:

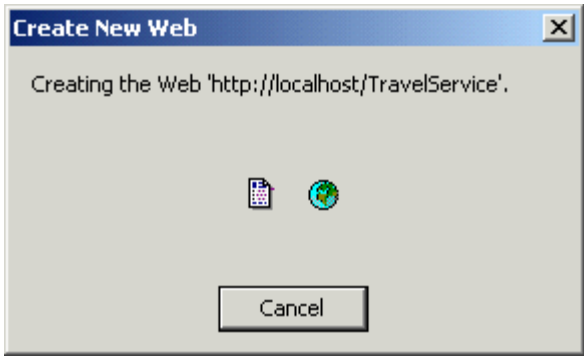
1) Mở Visual Studio IDE. Bắt đầu một dự án để tạo ra một ứng dụng hoặc một dịch vụ bằng cách chọn đến New trên menu File và chọn dự án để mở trong hộp thoại New Project.

2) Trong hộp thoại New Project, chọn Visual Basic Project trong cửa sổ Project type pane. Trong cửa sổ Templates, chọn ASP.NET Web Service. Trong mục Location, gõ tên và vị trí của dịch vụ Web XML như http://localhost/TravelService, và nhấn OK. Hình 7.3 cho thấy hộp thoại New Project.



Hình 7-3. The New Project dialog box

3) hộp thông báo" Create New Web" xuất hiện trong khi một thư mục web với tên được chỉ định đang được tạo ra. Hình 7.4 cho thấy hộp tin nhắn Create New Web.



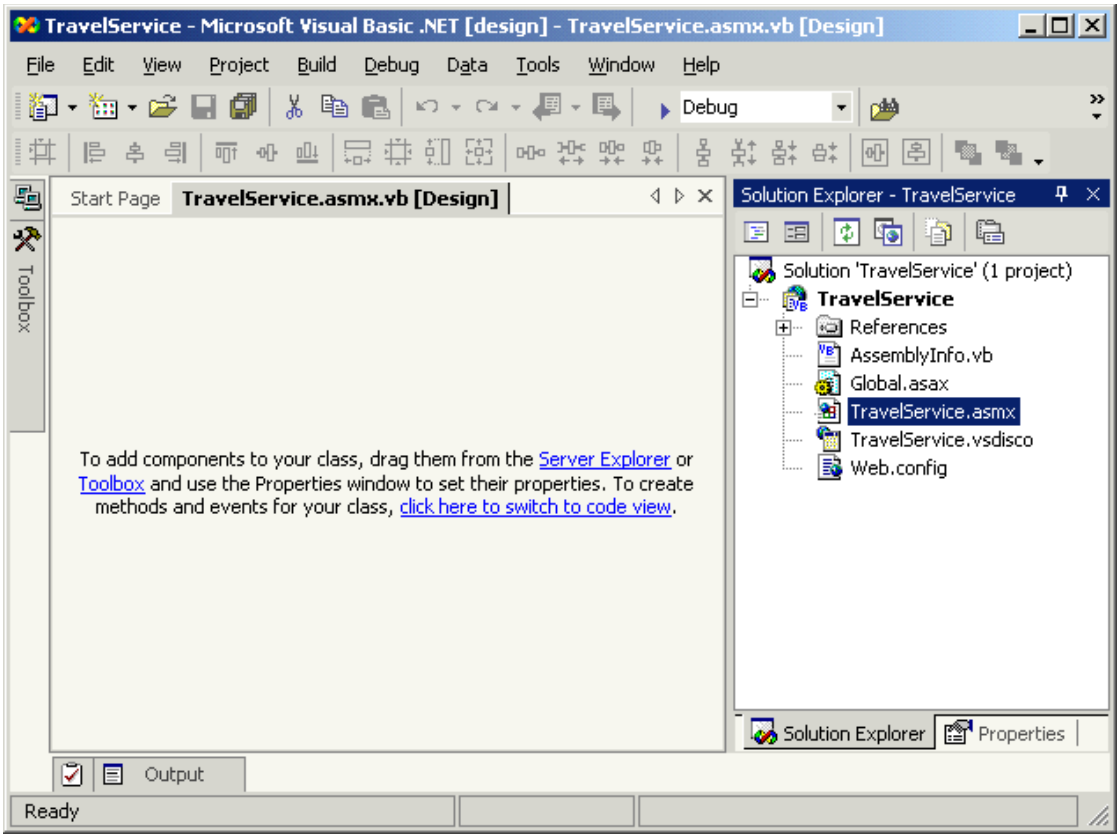
Hình 7-4. The Create New Web message box

Khi bạn tạo dự án ASP.NET Web dịch vụ, trang mặc định được hiển thị trong chế độ thiết kế, và được tạo ra một tập tin asmx. Sau khi tạo một dự án dịch vụ Web XML, bạn viết mã lệnh để cung cấp chức năng cho các dịch vụ

Web XML. Mã này được lưu trữ trong một tập tin liên kết với asmx.vb hoặc tập tin asmx.cs. Và được biết đến như là các tập tin mã phía sau(code-behind). Các tập tin mã phía sau phụ thuộc vào ngôn ngữ mà bạn sử dụng để tạo ra các dịch vụ Web XML. Đối với một dự án Visual C #, các tập tin mã phía sau là một tập tin .cs. Đối với một dự án Visual Basic, các tập tin mã phía sau là một tập tin .vb.

4) Đổi tên tập tin Service1.asmx trong Solution Explorer thành TravelService.asmx.

5) Để thêm chức năng cho các dịch vụ Web XML, chọn thẻ Solution Explorer, nhấp chuột phải vào file TravelService.asmx, và chọn View Code. Cửa sổ viết mã cho dịch vụ sẽ xuất hiện. Hình 7.5 cho thấy màn hình khởi động của dịch vụ Web XML TravelService.



Hình 7-5. The TravelService page in design mode

Đối với mỗi dịch vụ Web XML mà bạn tạo ra, các tập tin sau đây được tạo ra bởi Visual Studio. NET

-AssemblyInfo.cs hoặc AssemblyInfo.vb

-Global.asax

-Global.asax.cs hoặc Global.asax.vb

-Service1.asmx

-Service1.asmx.cs hoặc Service1.asmx.vb

-Service1.vsdisco

-Web.config

AssemblyInfo file

Một assembly là một đơn vị chức năng để chia sẻ và tái sử dụng trong thời gian chạy ngôn ngữ chung. Để biết thêm thông tin về assemblies và

configuration, hãy tham khảo phần tài liệu, " Understanding the .NET Framework.". Các tập tin AssemblyInfo bao gồm các thông tin chung về assemblies trong dự án. Để sửa đổi các thông tin, assemblies, chẳng hạn như thông tin về phiên bản, bạn thay đổi các thuộc tính trong các tập tin AssemblyInfo.

Web.config file

Tập tin này có chứa thông tin cấu hình, chẳng hạn như chế độ gỡ lỗi và phương thức xác thực cho một dự án Web. Nó cũng bao gồm thông tin về việc hiển thị các lỗi tùy chỉnh cho một dự án Web. Bạn cũng có thể sử dụng các tập tin Web.config để lưu trữ thông tin cấu hình tùy chỉnh cho dịch vụ Web XML của bạn.

Tập tin Global.asax và Global.asax.cs hoặc Global.asax.vb

Những tập tin này cho phép bạn quản lý sự kiện diễn ra application-level và session-level . Những tập tin này nằm trong thư mục gốc của một ứng dụng Web ASP.NET hoặc ASP.NET Web Service. Tập tin Global.asax.cs hoặc Global.asax.vb là một tập tin ẩn , trong đó có chứa mã để xử lý các sự kiện ứng dụng chẳng hạn như sự kiện Application_OnError.

Tập tin Service1.asmx và các tập tin Service1.asmx.cs hoặc Service1.asmx.vb

Hai tập tin này tạo nên một dịch vụ Web XML. Các tập tin Service1.asmx chứa các chỉ thị serviceprocessing XML Web và phục vụ điều khiển và chỉ dẫn dịch vụ Web XML. Các tập tin lớp Service1.asmx.cs hoặc Service1.asmx.vb là một tập tin ẩn WebService.asmx và chứa các lớp code-behind cho các dịch vụ Web XML.

WebService.vsdisco

Các tập tin. Vsdisco còn được gọi là các tập tin phát hiện hay khám phá. Tập tin này là một tập tin dựa trên XML có chứa các liên kết hoặc URL để các nguồn cung cấp thông tin khám phá hay thăm dò cho một dịch vụ Web XML.

Trước khi bạn tạo ra các dịch vụ Web XML TravelService, bạn cần phải tạo ra một cơ sở dữ liệu, BookingDetails, bao gồm các bảng sau đây:

* *AppointmentDetails.*

Bảng này lưu trữ chi tiết của tất cả các cuộc hẹn.

* *ContactPersonDetails.*

Bảng này lưu trữ các chi tiết của người được gặp nhau tại một địa điểm cụ thể.

**FlightDetails.*

Bảng này lưu trữ thông tin chi tiết của các hãng hàng không ưu đãi và các lớp học của người sử dụng.

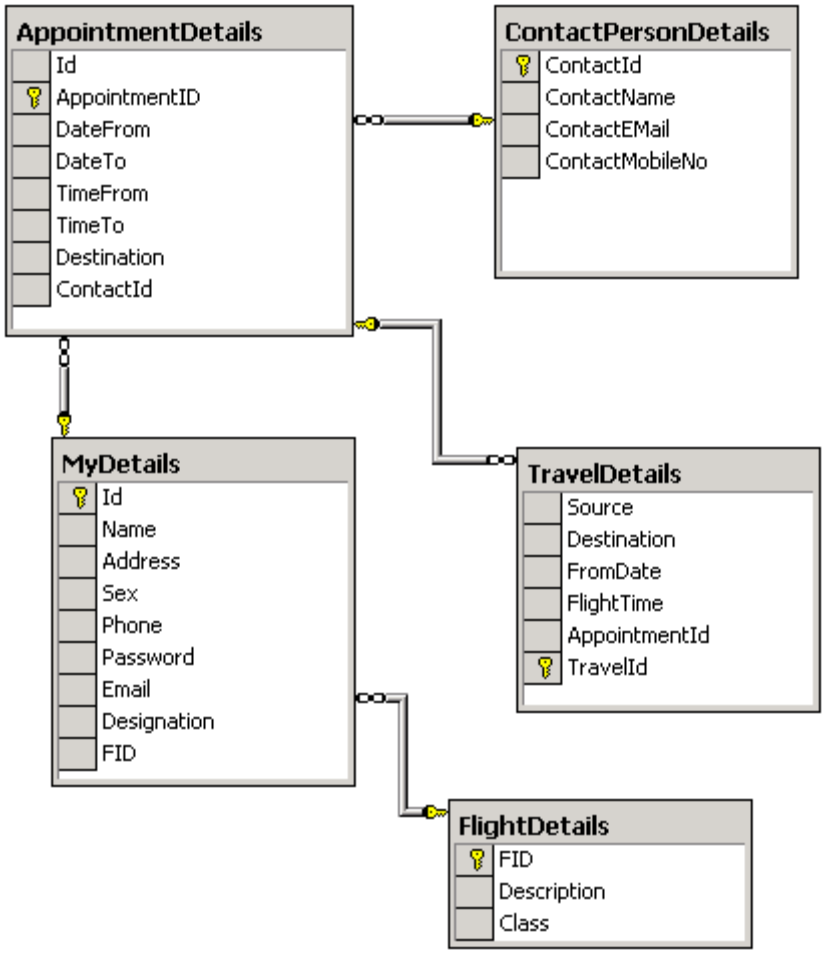
**MyDetails.*

Bảng này lưu trữ các chi tiết cá nhân và sở thích của người sử dụng.

**TravelDetails.*

Bảng này lưu trữ chi tiết du lịch, chẳng hạn như là điểm đến, thời gian chuyến bay, và ngày.

Hình 7,6 hiển thị các lược đồ cơ sở dữ liệu BookingDetails.



Hình 7-6. The BookingDetails database schema

Để tạo cơ sở dữ liệu BookingDetails và các bảng, thực hiện các bước sau:

1)Từ menu Start, xác định vị trí thư mục Microsoft SQL Server và mở Enterprise Manager.

2)Mở rộng nút SQL Server của bạn và xác định vị trí các nút Databases. Kích chuột phải vào nút Databases và chọn New Database từ menu chuột phải.

3)Trong hộp thoại Databases Properties, đặt tên BookingDetails trong mục Name, và click OK để tạo ra các cơ sở dữ liệu BookingDetails.

4)Mở Query Analyzer. Cung cấp thông tin xác thực cần thiết để kết nối với máy chủ cơ sở dữ liệu của bạn.

5)Chọn cơ sở dữ liệu BookingDetails từ danh sách các cơ sở dữ liệu trong Query Analyzer.

6)Thực thi Script sau đây.

SQL Script

```
if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[FK_TravelDetails_AppointmentDetails]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[TravelDetails] DROP CONSTRAINT
FK_TravelDetails_AppointmentDetails
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[FK_AppointmentDetails_ContactPersonDetails]') and OBJECTPROPERTY(
id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AppointmentDetails] DROP CONSTRAINT
FK_AppointmentDetails_ContactPersonDetails
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[FK_AppointmentDetails_MyPersonalDetails]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AppointmentDetails] DROP CONSTRAINT
FK_AppointmentDetails_MyPersonalDetails
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[AppointmentDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[AppointmentDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[ContactPersonDetails]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[ContactPersonDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[FlightDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[FlightDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[MyPersonalDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[MyPersonalDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[TravelDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[TravelDetails]
GO

if not exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[AppointmentDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
 BEGIN
```

```
CREATE TABLE [dbo].[AppointmentDetails] (
   [ID] [int] NOT NULL ,
   [AppointmentID] [int] NOT NULL ,
   [DateFrom] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [DateTo] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [TimeFrom] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [TimeTo] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Destination] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
   NOT NULL ,
   [ContactID] [int] NOT NULL
) ON [PRIMARY]
END

GO

if not exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[ContactPersonDetails]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
 BEGIN
CREATE TABLE [dbo].[ContactPersonDetails] (
   [ContactID] [int] NOT NULL ,
   [ContactName] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL ,
   [ContactEmail] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL ,
   [ContactMobileNo] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL
) ON [PRIMARY]
END

GO

if not exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[FlightDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
 BEGIN
CREATE TABLE [dbo].[FlightDetails] (
   [FlightID] [int] NOT NULL ,
   [Description] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL ,
   [Class] [char] (5) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
END

GO

if not exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[MyPersonalDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
 BEGIN
CREATE TABLE [dbo].[MyPersonalDetails] (
   [ID] [int] NOT NULL ,
   [Name] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Address] [char] (40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Sex] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Phone] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Password] [char] (8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Email] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
```

```
   [Designation] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL,
   [FlightID] [int] NOT NULL
) ON [PRIMARY]
END

GO

if not exists (select * from dbo.sysobjects where id = object_id(
N'[dbo].[TravelDetails]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
 BEGIN
CREATE TABLE [dbo].[TravelDetails] (
   [TravelID] [int] NOT NULL ,
   [Source] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [Destination] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL,
   [FromDate] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
   [FlightTime] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS
      NOT NULL,
   [AppointmentID] [int] NOT NULL
) ON [PRIMARY]
END

GO

ALTER TABLE [dbo].[AppointmentDetails] WITH NOCHECK ADD
   CONSTRAINT [PK_AppointmentDetails] PRIMARY KEY CLUSTERED
   (
      [AppointmentID]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[ContactPersonDetails] WITH NOCHECK ADD
   CONSTRAINT [PK_ContactPersonDetails] PRIMARY KEY CLUSTERED
   (
      [ContactID]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[FlightDetails] WITH NOCHECK ADD
   CONSTRAINT [PK_FlightDetails] PRIMARY KEY CLUSTERED
   (
      [FlightID]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[MyPersonalDetails] WITH NOCHECK ADD
   CONSTRAINT [PK_MyPersonalDetails] PRIMARY KEY CLUSTERED
   (
      [ID]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[TravelDetails] WITH NOCHECK ADD
   CONSTRAINT [PK_TravelDetails] PRIMARY KEY CLUSTERED
   (
```

```
      [TravelID]
   )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[AppointmentDetails] ADD
   CONSTRAINT [FK_AppointmentDetails_ContactPersonDetails] FOREIGN KEY
   (
      [ContactID]
   ) REFERENCES [dbo].[ContactPersonDetails] (
      [ContactID]
   ),
   CONSTRAINT [FK_AppointmentDetails_MyPersonalDetails] FOREIGN KEY
   (
      [ID]
   ) REFERENCES [dbo].[MyPersonalDetails] (
      [ID]
   )
GO

ALTER TABLE [dbo].[TravelDetails] ADD
   CONSTRAINT [FK_TravelDetails_AppointmentDetails] FOREIGN KEY
   (
      [AppointmentID]
   ) REFERENCES [dbo].[AppointmentDetails] (
      [AppointmentID]
   )
GO
```

Sau khi bạn tạo cơ sở dữ liệu và các bảng, bạn cung cấp các chức năng liên quan đến các dịch vụ Web XML. Để kích hoạt các ứng dụng của Client để truy cập và sử dụng dịch vụ Web XML, bạn giao tiếp với các chức năng của dịch vụ Web XML bằng cách tạo ra các phương thức web. Phương thức web là những đơn vị độc lập mà chức năng thực hiện một số hoạt động được xác định trước trong một dịch vụ Web XML. Bạn sử dụng các thuộc tính <WebMethod> trong Visual Basic. NET hoặc [WebMethod] thuộc tính trong Visual C # để khai báo một phương thức web, tùy thuộc vào ngôn ngữ mà bạn chọn để phát triển dịch vụ Web XML của bạn. Đoạn mã sau đây cho thấy một phương thức web mẫu:

Visual Basic .NET

```
<WebMethod()> Public Function HelloWorld() As String
   HelloWorld = "Hello World"
```

```
End Function
```

Visual C#

```
[WebMethod]
public string HelloWorld()
{
   return "Hello World";
}
```

Dịch vụ Web XML TravelService bao gồm hai phương thức Web, MakeReservations và SendMessage. Phương thức MakeReservations web được giao tiếp với các ứng dụng của Client cho các dịch vụ Web XML TravelService. Phương thức SendMessage Web được sử dụng để gửi e-mail thông báo đến địa chỉ e-mail được chỉ định.

Mã lệnh phương thức MakeReservations viết như sau:

Visual Basic .NET

```
<WebMethod()> _
Public Function MakeReservations(ByVal DateFrom As String, _
   ByVal TimeFrom As String, ByVal TimeTo As String, _
   ByVal Destination As String, ByVal ContactId As String) As String

End Function
```

Visual C#

```
[WebMethod]
public string MakeReservations(string DateFrom, string
TimeFrom,
    string TimeTo, string Destination, string
ContactId)
{

}
```

Phương thức Web MakeReservations thực hiện ba nhiệm vụ chính:

-Populates DataSet

-Tạo đặt phòng

-Gửi một e-mail thông báo

Populating các DataSet

Mã lệnh cho populating DataSet khởi tạo các đối tượng OleDbConnection và tạo ra một kết nối với máy chủ thông qua đối tượng. Để thực hiện nhiệm vụ này, chuỗi kết nối được kê khai theo thông số kỹ thuật của máy chủ. Một đối tượng của lớp DataSet được tạo ra để lấy nội dung từ bảng AppointmentDetails. Cuối cùng, một đối tượng OleDbDataAdapter được tạo ra để thực thi một truy vấn và cư trú trong DataSet.

Visual Basic .NET

```
Imports System.Data
Imports System.Data.OleDB

Public Class MyWebService

   <WebMethod()> _
   Public Function MakeReservations(ByVal ID As Integer, ByVal _
      DateFrom As String, ByVal DateTo As String, ByVal TimeFrom As _
      String, ByVal TimeTo As String, ByVal Destination As String, _
      ByVal ContactId As String) As String
      Dim sqlStatement As String = ""
      Try
         Dim strConnectionString As String
         strConnectionString = "Provider=SQLOLEDB.1; Data Source=" & _
            "localhost;uid=sa;pwd=;Initial Catalog=BookingDetails;"
         Dim conADOConnect As New OleDbConnection(strConnectionString)
         conADOConnect.Open()
         Dim dsID As New DataSet()
         Dim comADOCommand As New OleDbCommand()
         comADOCommand.Connection = conADOConnect
         comADOCommand.CommandText = "Select * from AppointmentDetails"
         Dim adosc As New OleDbDataAdapter()
         adosc.SelectCommand = comADOCommand
         adosc.Fill(dsID, "AppointmentDetails")
         Dim dview As New DataView(dsID.Tables("AppointmentDetails"))
```

```
            dview.RowStateFilter = DataViewRowState.CurrentRows
            ' Code for creating the reservation
            .
            .
            .

        Catch e As Exception
            MakeReservations = e.Message
        End Try
    End Function



End Class
```

Visual C#

```
using System.Data.OleDb;

public class MyWebService
{
    [WebMethod]
    public string MakeReservations(int ID, string DateFrom, string
    DateTo, string TimeFrom, string TimeTo, string Destination, int
    ContactId)
    {
        String sqlStatement = "";
        try
        {
            String strConnectionString = "";
            strConnectionString = "Provider= SQLOLEDB.1; Data Source=" +
                "localhost; uid=sa; pwd=; Initial Catalog=Booker;";
            OleDbConnection conADOConnect = new OleDbConnection(
                strConnectionString);
            conADOConnect.Open();
            DataSet dsID = new DataSet();
            OleDbCommand comAdoCommand = new OleDbCommand();
            comAdoCommand.Connection = conADOConnect;
            comAdoCommand.CommandText =
                "Select * from AppointmentDetails";
            OleDbDataAdapter adosc = new OleDbDataAdapter();
```

```
            adosc.SelectCommand = comAdoCommand;
            adosc.Fill(dsID,"AppointmentDetails");
            DataView dview = new
               DataView(dsID.Tables["AppointmentDetails"]);
            dview.RowStateFilter = DataViewRowState.CurrentRows;


            // Code for creating the reservation
            .
            .
            .
        }
        catch(Exception e)
        {
            return e.ToString ();
        }
    }
}
```

Tạo chức năng Đặt phòng

Sau khi populating DataSet, bước tiếp theo là để tạo ra một chức năng đặt phòng. Mã lệnh này tính toán giá trị cuối cùng của trường AppointmentID và tăng nó 1 đơn vị. Trường(field) AppointmentID là một trường khóa chính trong bảng, và bạn không thể chèn các giá trị hoặc để null vào trường này. Sau đây cho thấy mã lệnh cho việc tạo ra cách đặt phòng.

Visual Basic .NET

```
' Code for creating the reservation
```

```
Dim conid As Integer = 0
Dim drview As DataRowView
For Each drview In dview
   conid = Int32.Parse(drview("AppointmentId").ToString())
Next
If conid > 0 Then
   conid = conid + 1
Else
   conid = 1
End If
Dim adcConnection As New OleDbConnection(strConnectionString)
adcConnection.Open()
sqlStatement = "Insert into AppointmentDetails values(ID," + _
   conid + ",'" + DateFrom + "','" + DateTo + "','" + TimeFrom + _
   "','" + TimeTo + "','" + Destination + "'," + ContactId + ")"
conid = conid + 1
Dim cSQLStatement As New OleDbCommand(sqlStatement, adcConnection)
cSQLStatement.CommandText = sqlStatement
cSQLStatement.ExecuteNonQuery()
```

Visual C#

```
// Code for creating the reservation
//

int conid=0;
foreach(DataRowView drview in dview)
{
   conid = Int32.Parse(drview["AppointmentId"].ToString());
}
if(conid>0)
{
   conid++;
}
else
{
   conid = 1;
}
OleDbConnection adcConnection = new OleDbConnection(strConnectionString);
adcConnection.Open();
sqlStatement = "Insert into AppointmentDetails values(" + ID + "," +
```

```
   conid + ",'" + DateFrom + "','" + DateTo + "','" + TimeFrom + "','" +
   TimeTo + "','" + Destination + "'," + ContactId + ")";
conid++;
OleDbCommand cSQLStatement = new
OleDbCommand(sqlStatement,adcConnection);
cSQLStatement.CommandText = sqlStatement;
cSQLStatement.ExecuteNonQuery();
```

Gửi một thông báo E-Mail

Nhiệm vụ cuối cùng của phương thức MakeReservations web là để gửi một thông báo qua e-mail cho người sử dụng trong danh sách liên lạc về việc hẹn gặp và các chi tiết về lịch trình du lịch. Trong đoạn code để thực hiện việc đặt chỗ, một tham chiếu tới XML Web Service được tạo ra. Bạn sẽ tìm hiểu làm thế nào để thêm một tham chiếu web trong bài sau. Một XML Web Service khác là MailSender sẽ gửi e-mail thông báo. Cuối cùng, một đối tượng của XML Web Service được tạo ra, và sau đó phương thức web SendMessage, sẽ giao tiếp với các dịch vụ Web XML, khi nó được gọi.

Lưu ý

Hãy chắc chắn rằng bạn đã thêm các yêu cầu về namespace cho các dịch vụ MailSender để thực hiện. Bạn nên thêm System.Text.RegularExpressions và không gian tên System.Web.Mail. Ngoài ra, xin lưu ý rằng toàn bộ khối mã lệnh sau là một phần của phương thức MakeReservations.

Visual Basic .NET

```
'Sending mail
SendMessage("johnd@mymail.com", "susanj@mail.com", _
   "Appointment Booking Confirmation", _
   "Your appointment has been confirmed.", 1)
MakeReservations = "Appointment created successfully."
```

Visual C#

```
/* Sending mail */
SendMessage("johnd@mymail.com", "susanj@mail.com",
   "Appointment Booking Confirmation",
   "Your appointment has been confirmed.", 1);
return "Appointment created successfully.";
```

Tương tự như vậy, dịch vụ Web XML TravelService chứa một phương thức web sẽ gửi e-mail thông báo cho người sử dụng bất cứ khi nào bạn thực hiện đặt phòng cho họ.

Phương thức web SendMessage sẽ gửi một thông điệp tới mỗi người nhận trong danh sách mà nó được cung cấp. Phương thức này chấp nhận địa chỉ e-mail của một người gửi, và của một danh sách người nhận, chủ đề, nội dung, và định dạng của tin nhắn. Mỗi địa chỉ người nhận được giới hạn bằng cách sử dụng một dấu phẩy, dấu chấm phẩy, hoặc một tab. Thông điệp có thể là văn bản gốc hoặc là các định dạng HTML, như thể hiện trong các mã sau đây.

Visual Basic .NET

```
<WebMethod()> Public Function SendMessage(ByVal from As String, _
   ByVal recipients As String, ByVal subject As String, _
   ByVal body As String, ByVal format As Integer) As String
   Dim retValue As String = ""
      Try
         Dim aryRecipients As String()
         Dim Separators As Char() = New Char(2) {",", ";", "\t"}
         aryRecipients = recipients.Split(Separators)
         Dim recipient As String
         For Each recipient In aryRecipients
            If recipient <> "" And recipient <> Nothing Then
            retValue = SendMail(from, recipient, subject, body, _
               format)
            End If
         Next
```

```
        Catch e As Exception
            retValue = "Error"
            Throw New Exception(e.ToString())
        End Try
    SendMessage = retValue
End Function
```

Visual C#

```
public string SendMessage(string from, string recipients, string
    subject,string body,int format)
{
    string retValue = "";
    try
    {
        string[] aryRecipients;
        char[] Separators = new Char[] {',', ';', '\t'};
        aryRecipients = recipients.Split(Separators);
        foreach (string recipient in aryRecipients)
        {
            if ((recipient != "") && (recipient != null))
            {
                retValue = SendMail(from, recipient, subject, body,
                    format);
            }
        }
    }
    catch(Exception e)
    {
        retValue = "Error";
        throw new Exception(e.ToString());
    }
    return retValue;
}
```

<WebMethod> chỉ thị thông báo cho các ứng dụng khác mà họ có thể truy cập vào các phương thức SendMessage. Thuộc tính WebMethod được trình bày chi tiết trong phần" Advanced XML Web Services Programming" của tài liệu MCAD, Phương thức SendMessage bao gồm các thông số sau:

-***Form*** là tham số đại diện cho ID e-mail của người gửi. Kiểu dữ liệu là String

- ***Recipients*** là tham số đại diện cho ID e-mail của người nhận. Kiểu dữ liệu là String

- ***Subject*** là tham số đại diện cho chủ đề của thư. Kiểu dữ liệu là String.

- ***Body*** là tham số đại diện cho nội dung tin nhắn. Kiểu dữ liệu là String.

- ***Format*** là tham số đại diện cho các định dạng của các tin nhắn được soạn.

Nó là một kiểu dữ liệu số nguyên có thể nhận hai giá trị 0 cho văn bản Text và 1 cho HTML.

Phương thức SendMessage chia tách các danh sách địa chỉ người nhận vào các địa chỉ cá nhân và gọi phương thức SendMail để gửi tin nhắn cho mỗi địa chỉ. Xem các mã lệnh sau đây để xem cách sử dụng phương thức SendMail để gửi tin nhắn e-mail.

Visual Basic .NET

```
Protected Function SendMail(ByVal fromID As String, ByVal toID As _
   String, ByVal subject As String, ByVal body As String, ByVal format _
   As Integer) As String
   Dim strErrorMessage As String
   If ValidateEmail(toID) Then
      Try
         Dim objMessage As New MailMessage()
         objMessage.From = fromID
         objMessage.To = toID
         objMessage.Subject = subject
         objMessage.Body = body
         If format = 0 Then
            objMessage.BodyFormat = MailFormat.Text
         Else
            objMessage.BodyFormat = MailFormat.Html
```

```
        End If
        SmtpMail.Send(objMessage)
        SendMail = "Sent"
      Catch e As Exception
      If e.Message.ToString = "Could_not_access_cdo_newmail_object" Then
        strErrorMessage = "Failed. Mail server was not " & _
        "available. Please try again later."
      Else
        strErrorMessage = "Failed.  " + e.ToString() + "."
        SendMail = strErrorMessage
      End If
      End Try
   Else
      strErrorMessage = "Failed. The address was poorly formed."
      SendMail = strErrorMessage
   End If
End Function
```

Visual C#

```
protected string SendMail(string from, string to, string subject,
   string body, int format)
{
   string strErrorMessage = null;
   if (ValidateEmail(to))
   {
      try
      {
         MailMessage objMessage = new MailMessage();
         objMessage.From = from;
         objMessage.To = to;
         objMessage.Subject = subject;
         objMessage.Body = body;
         if (format == 0)
         {
           objMessage.BodyFormat = MailFormat.Text;
         }
         else
         {
           objMessage.BodyFormat = MailFormat.Html;
         }
```

```
            SmtpMail.Send(objMessage);
            return "Sent";
        }
        catch (Exception e)
        {
            if (e.Message== "Could_not_access_cdo_newmail_object")
            {
                strErrorMessage = "Failed. Mail server was not " +
                    "available. Please try again later.";
            }
            else
            {
                strErrorMessage = "Failed.  " + e.ToString() + ".";
            }
            return strErrorMessage;
        }
    }
    else
    {
        strErrorMessage = "Failed. The address was poorly formed.";
        return strErrorMessage;
    }
}
```

Phương thức SendMail sử dụng lớp MailMessage để soạn và gửi thư. Để sử dụng các thuộc tính và phương thức của lớp MailMessage, bạn cần tạo một đối tượng của lớp MailMessage. Trong đoạn mã trên, các đối tượng của lớp MailMessage được đặt tên là objMessage.

Visual Basic .NET

```
Public Function ValidateEmail(ByVal email As String) As Boolean
    Dim blnValidated As Boolean = False
    If Regex.IsMatch(email, "@") Then
        blnValidated = True
    End If
    Return blnValidated
```

```
End Function
```

Visual C#

```
public bool ValidateEmail(string email)
{
   bool blnValidated = false;
   if (Regex.IsMatch(email,"@"))
   {
      blnValidated = true;
   }
   return blnValidated;
}
```

Trước khi tạo một đối tượng của lớp MailMessage, bạn cần kiểm tra có hoặc không có địa chỉ e-mail của người nhận ở trong các định dạng gửi đi. Trong đoạn mã trên, phương pháp ValidateEmail xác minh các địa chỉ cho các định dạng mẫu gửi đi.

Phương thức ValidateEmail kiểm tra xem địa chỉ e-mail của người nhận là hợp lệ, bằng cách kiểm tra nếu nó có chứa ký hiệu @. Xác minh này được thực hiện bằng cách sử dụng phương pháp IsMatch của lớp Regex. Phương thức ValidateEmail trả về True nếu địa chỉ e-mail của người nhận có chứa các ký hiệu @, nếu không, nó sẽ trả về False.

Phương thức SendMail là một thể hiện của lớp MailMessage bao gồm các thông số sau đây:

- Thuộc tính From, To, Subject, và Body của objMessage được thiết lập với các giá trị và truyền cho phương thức SendMail theo phương thức Web SendMessage.

-Các thuộc tính BodyFormat của objMessage sẽ thiết đặt MailFormat.Text hoặc MailFormat.HTML tùy thuộc vào giá trị được thông qua với phương thức SendMail.

-Các tin nhắn được soạn và đi gửi bằng cách sử dụng phương thức Send của lớp SmtpMail.

-Phương thức Send của lớp SmtpMail chấp nhận một đối tượng của lớp MailMessage như một tham số.

2.1.3. Phát triển và công bố một XML Web Service

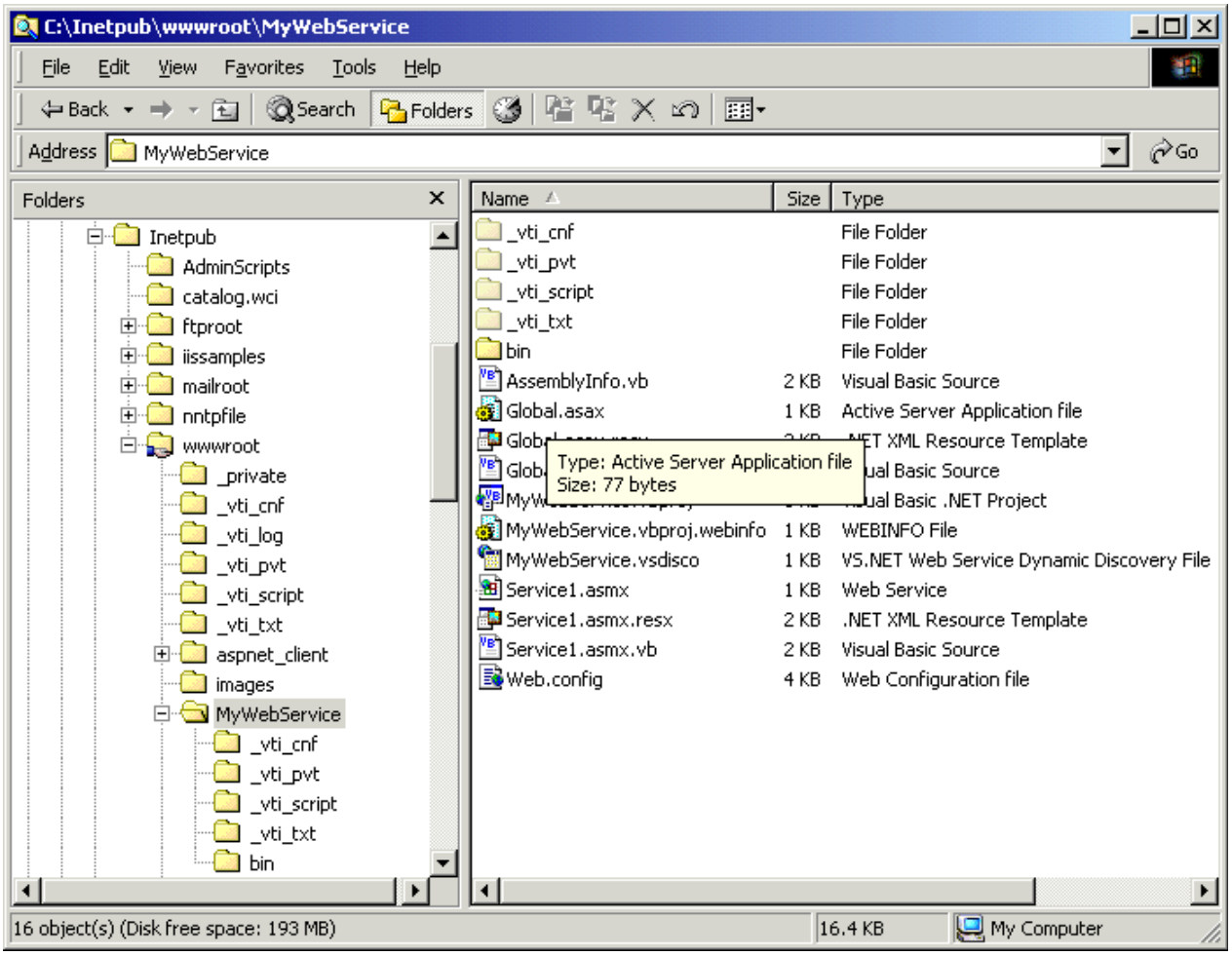
Sau khi bạn tạo ra một dịch vụ Web XML, bạn cần để triển khai các dịch vụ Web XML đến một máy chủ Web để cho phép nó chạy thường trú trên máy chủ và cho các ứng dụng làm việc với dịch vụ này. Khi bạn triển khai một dịch vụ Web XML, bạn tạo ra ra file Disco. File Disco trong một dịch vụ Web XML có chứa các thông tin như vị trí của các dịch vụ Web. Mỗi Client có nhu cầu truy cập dịch vụ Web XML của bạn sử dụng tập tin này để xác định vị trí các dịch vụ Web. Trong bài học này, bạn sẽ tìm hiểu về các công việc mà bạn cần phải thực hiện khi bạn triển khai và công bố một dịch vụ Web XML. Ngoài ra, bạn sẽ tìm hiểu về Web XML cơ chế dịch vụ phát hiện và làm thế nào để cấu hình thông tin khám phá về dịch vụ Web XML của bạn.

Công bố một dịch vụ Web XML

Khi bạn triển khai một dịch vụ Web XML trên một máy chủ Web, dịch vụ này được công bố trên web và trở nên truy cập vào các ứng dụng của Client. Bạn có thể triển khai một dịch vụ Web XML trên một máy chủ Web theo hai cách. Bạn có thể tạo một dự án thiết lập Web hoặc sao chép các tập tin dịch vụ Web XML đến máy chủ Web. Bạn sẽ tìm hiểu làm thế nào để tạo ra một dự án thiết lập Web trong bài, " Deploying XML Web Services and Server Components." trong giáo trình MCAD. Bây giờ bạn sẽ triển khai một dịch vụ Web XML bằng cách sao chép các tập tin dịch vụ Web XML đến máy chủ Web.

Để triển khai dịch vụ Web XML của bạn trên một máy chủ Web, thực hiện các bước sau:

1)Sao chép các tập tin XML ứng dụng dịch vụ Web, trong thư mục Inetpub \ wwwroot của bạn. Bạn chỉ cần sao chép các tập tin asmx, Web.config, và các tập tin Global.asax vào thư mục ứng dụng và các tập tin .dll vào thư mục \bin . Hình 7.7 hiển thị cấu trúc thư mục của một dịch vụ triển khai Web XML.

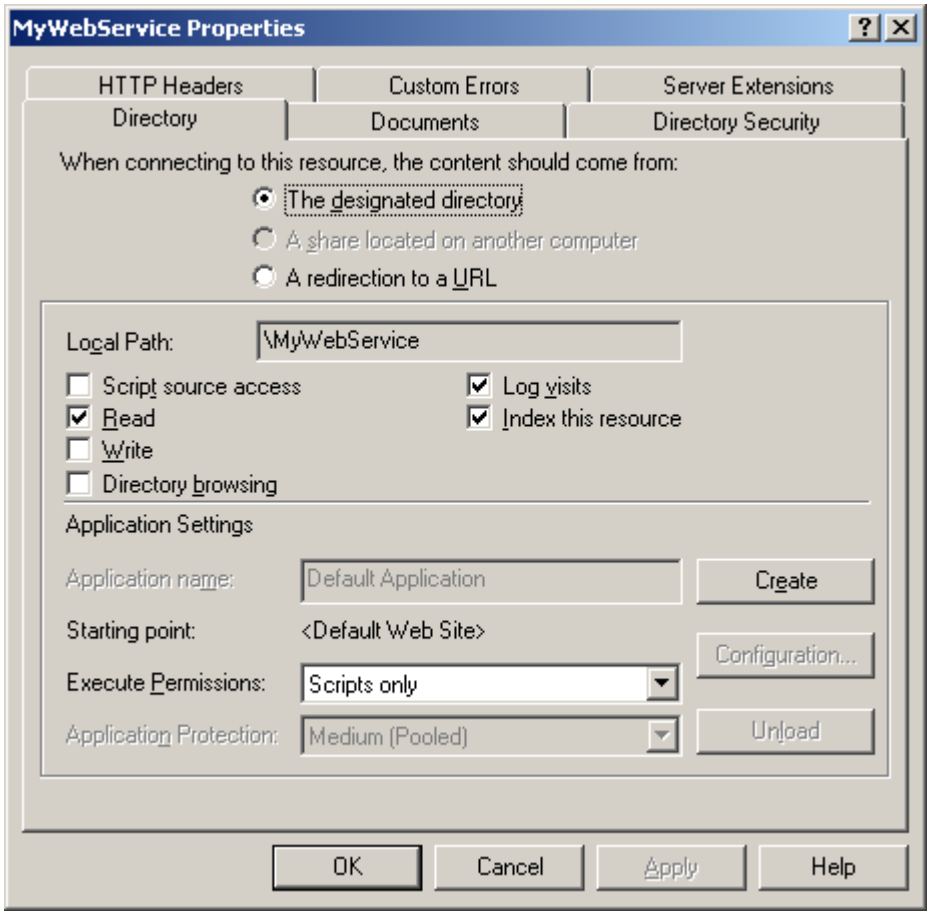


Hình 7-7. The directory structure of a virtual directory that contains an XML Web service

2)Mở Internet Services Manager từ thư mục Administrative Tools.

3)Mở nút Default Website.

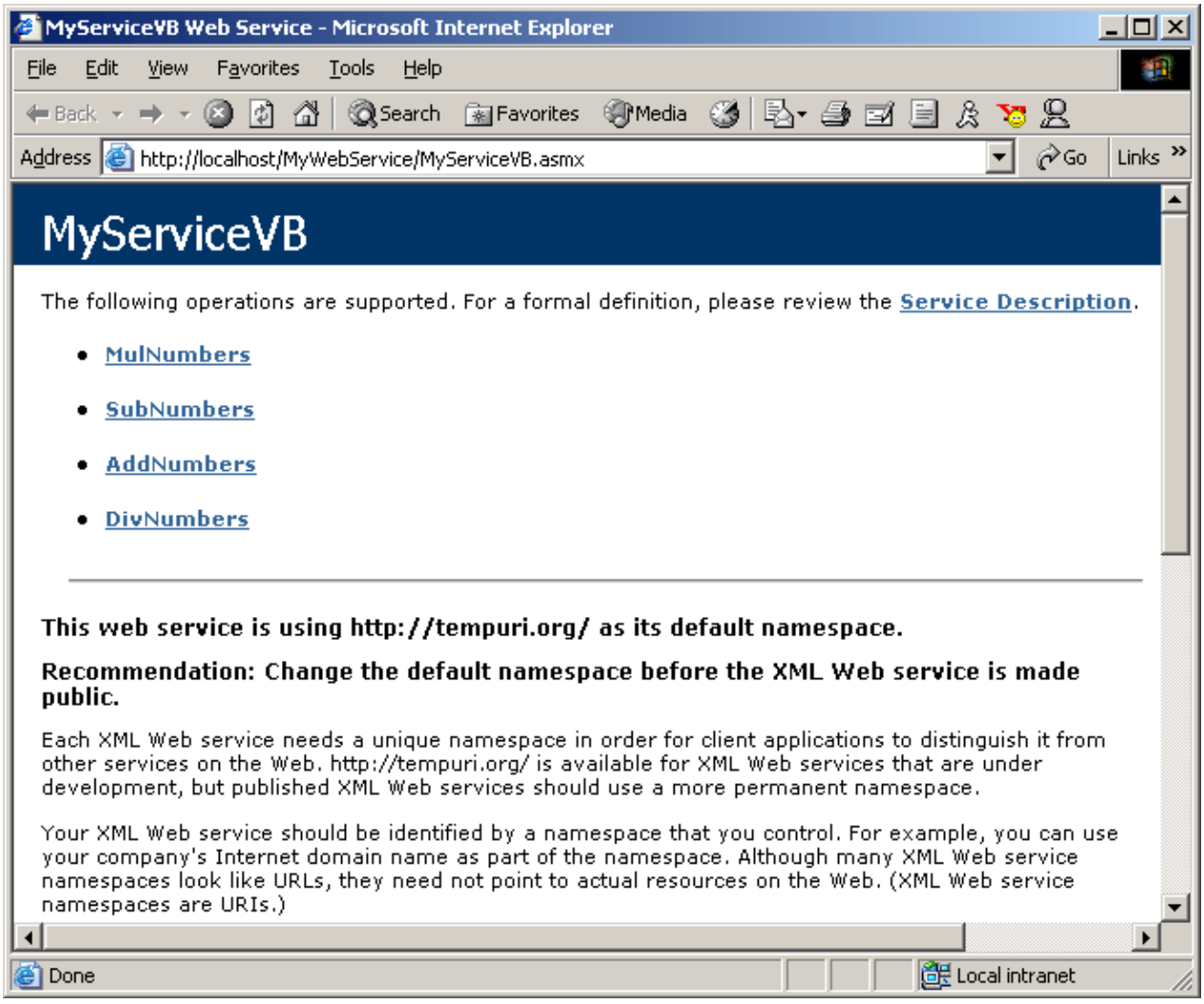
4)Kích chuột phải vào thư mục dịch vụ web mà bạn đã sao chép trong thư mục Inetpub \ wwwroot, và chọn Properties từ menu chuột phải để mở hộp thoại MyWebService Properties, như thể hiện trong hình 7.8.



Hình 7-8. The MyWebService Properties dialog box

5)Nhấp vào nút Create trong hộp thoại MyWebService Properties để cấu hình các thư mục ảo như là thư mục gốc của ứng dụng dịch vụ Web của bạn.

Sau khi bạn cấu hình ứng dụng dịch vụ Web của bạn, dịch vụ Web của bạn đã sẵn sàng để được sử dụng bởi các ứng dụng của Client. Bạn có thể sau đó nhấp chuột phải vào tập tin. Asmx trong khung bên phải của Internet Services Manager và chọn Browse từ menu chuột phải để xem các mô tả dịch vụ Web của bạn, thể hiện trong hình 7,9.



Hình 7-9. The description of a Web service

Những thành phần được công bố với một Web Service

Khi bạn triển khai một dịch vụ Web XML, các thành phần sau của dịch vụ Web được công bố trên web:

- Thư mục ứng dụng Web.

Thành phần này là thư mục gốc cho dịch vụ Web XML của bạn. Tất cả các tập tin dịch vụ Web XML có mặt trong thư mục này hoặc các thư mục con mà bạn có thể tạo ra. Bạn cần phải cấu hình các thư mục ứng dụng Web như một ứng dụng web IIS.

- Các tập tin <WebService>. asmx.

Tập tin này là URL cơ sở (*được đính kèm lời gọi vào URL*) cho các Client truy cập vào dịch vụ Web XML của bạn.

-Các tập tin <WebService>.disco.

Đây là một tập tin tùy chọn mô tả cơ chế phát hiện ra cho dịch vụ Web XML của bạn. Bạn sẽ tìm hiểu thêm về cơ chế phát hiện ra trong bài học kế tiếp.

-Các tập tin Web.config.

Tập tin này là tùy chọn. Thêm nó để ghi đè lên các thiết lập cấu hình mặc định của dịch vụ Web XML của bạn. Bạn có thể sử dụng file cấu hình để tùy chỉnh hệ thống hoặc lưu trữ thông tin cấu hình cho các ứng dụng của bạn.

-Thư mục \Bin

Thư mục này chứa các tập tin nhị phân cho các dịch vụ Web XML. Nếu bạn không xác định XML lớp dịch vụ Web của bạn trong cùng một tập tin như các tập tin asmx, assembly có chứa trong class sẽ được hiện diện trong thư mục \ Bin.

Sau khi triển khai dịch vụ Web XML đến một máy chủ Web, các ứng dụng có thể xác định vị trí các dịch vụ Web XML bằng cách sử dụng một cơ chế phát hiện.

Hiểu biết về cơ chế hoạt động XML Web Services Discovery

XML Web Services Discovery ra cơ chế cho phép một ứng dụng Client (Client) để xác định vị trí hoặc khám phá những tài liệu mô tả một dịch vụ Web XML. XML Web Services Discovery trả về một tài liệu mô tả dịch vụ cho Client(Client). Các tài liệu mô tả của dịch vụ được viết bằng ngôn ngữ Web Services Description Language (WSDL) và chứa các thông tin về khả năng của một dịch vụ Web XML, vị trí của nó, và làm thế nào để tương tác với nó. Trong

quá trình phát hiện dịch vụ Web XML, các tập tin, chẳng hạn như mô tả dịch vụ, lược đồ XSD, hoặc văn bản Document Discovery, có chứa các chi tiết của một dịch vụ Web XML, được tải về máy tính của Client.

Một tập tin. Disco, hoặc Document Discovery, chứa các liên kết tới các nguồn khác mô tả dịch vụ Web XML của bạn và cho phép Client khám phá ra một dịch vụ Web XML. Các tài Document Discovery có chứa thông tin về các dịch vụ Web XML khác nằm trên cùng một máy chủ Web hoặc khác nhau. Các mã sau đây sẽ hiển thị các nội dung của một tài liệu khám phá Document Discovery.

```
XML
<?xml version="1.0" encoding="utf-8"?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://schemas.xmlsoap.org/disco/">
<discoveryRef ref="/Folder/Default.disco"/>
<contractRef ref="http://Nancyd/mywebservice/Service1.asmx?wsdl"
   docRef="http://Nancyd/mywebservice/Service1.asmx"
   xmlns="http://schemas.xmlsoap.org/disco/scl/" />
<soap address="http://Nancyd/mywebservice/Service1.asmx"
   xmlns:q1="http://tempuri.org/" binding="q1:Service1Soap"
   xmlns="http://schemas.xmlsoap.org/disco/soap/" />
</discovery>
```

Bạn có thể thêm nhiều tham chiếu của những mô tả dịch vụ bởi các (element) phần tử <discovery>. Bạn có thể chỉ định các tài liệu tham chiếu mô tả dịch vụ trong một tài liệu discovery bằng cách thêm vào một phần tử

<contractRef> với các không gian tên XML http://schemas.xmlsoap.org/disco/scl/. Tương tự như vậy, bạn có thể chỉ định các tài liệu tham chiếu các tài liệu discovery và lược đồ XSD khác bằng cách thêm phần tử <discoveryRef> và <schemaRef>, tương ứng. Nếu bạn thêm một tài liệu tham chiếu là giản đồ XSD, bạn cần phải xác định không gian tên XML http://schemas.xmlsoap.org/disco/schema. Đối với tất cả các tài liệu tham chiếu, bạn xác định vị trí của tài liệu bằng cách sử dụng các thuộc tính ref.

Cấu hình thông tin Discovery cho một Dịch vụ Web

Trong lúc thiết kế, quá trình phát hiện giúp các ứng dụng Client của một dịch vụ Web tìm hiểu về chi tiết, chẳng hạn như vị trí và khả năng của các dịch vụ Web. Cơ chế phát hiện cũng cho phép các Client để tìm hiểu làm thế nào để tương tác với dịch vụ Web XML.

Một ứng dụng Client có thể phát hiện ra một dịch vụ Web XML nếu bạn xuất bản một tập tin. Disco cho dịch vụ Web XML của bạn. Các dịch vụ Web XML bạn sử dụng ASP.NET để tự động tạo ra một tài liệu phát hiện. Ngoài ra, phát hiện một tài liệu được tự động tạo ra cho một dịch vụ Web XML khi bạn truy cập nó bằng cách sử dụng một URL với DISCO trong chuỗi truy vấn Ví dụ, nếu URL cho một dịch vụ Web XML là http://www.contoso.com/getprice.asmx, phát hiện một tài liệu được tự động tạo ra với http://www.contoso.com/getprice.asmx DISCO như URL. Để cấu hình các thông tin phát hiện ra cho một dịch vụ Web XML, thực hiện các bước sau:

1) Tạo một tài liệu XML và chèn <phiên bản xml = "1.0" > Tag trong dòng đầu tiên.

2)Thêm một phần tử <discovery>, chẳng hạn như:

```
<discovery xmlns="http://schemas.xmlsoap.org/disco/">
```

```
</discovery>
```

3) Thêm các tham chiếu cho mô tả dịch vụ, lược đồ XSD, và các văn bản discovery khác ra bên trong phần tử <discovery> như hiển thị trong các mã sau đây:

```
<?xml version="1.0"?>
<discovery xmlns="http://schemas.xmlsoap.org/disco/">
<discoveryRef ref="/Folder/Default.disco"/>
<contractRef ref="http://NancyD/MyWebService.asmx?WSDL"
   docRef="Service.htm"
   xmlns="http://schemas.xmlsoap.org/disco/scl/"/>
<schemaRef ref="Schema.xsd"
   xmlns="http://schemas.xmlsoap.org/disco/schema/"/>
</discovery>
```

4)Triển khai các tài liệu discovery trên một máy chủ Web.

Sau khi xác định các thông tin discovery ra cho dịch vụ Web XML của bạn và xuất bản các tập tin Dico, người dùng có thể duyệt các tập tin discovery để xác định vị trí dịch vụ Web XML của bạn.

Ngoài ra các tập tin disco., Studio Visual NET tạo ra một tập tin vsdisco, cho phép phát hiện tự động của các dịch vụ Web. Tự động phát hiện cho phép một ứng dụng Client lặp đi lặp lại tìm kiếm thông qua các thư mục trên một máy chủ Web để xác định vị trí tất cả các dịch vụ Web XML có sẵn trên máy chủ Web. Vsdisco là một tập tin dựa trên XML với <dynamicDiscovery> như là nút gốc thay vì nút <discovery> trong một tài liệu disco. Nút này có chứa một hay nhiều nốt <exclude>. <exclude> Nút có chứa một thuộc tính đường dẫn, trong đó

có các đường dẫn tương đối tới một thư mục con mà quá trình phát hiện động sẽ loại trừ. Các mã sau đây sẽ hiển thị một mẫu. Tập tin Vsdisco.

```
XML
<?xml version="1.0" encoding="utf-8" ?>
<dynamicDiscovery xmlns="urn:schemas-dynamicdiscovery:disco.2000-03-17">
   <exclude path="_vti_cnf" />
   <exclude path="_vti_pvt" />
   <exclude path="_vti_log" />
   <exclude path="_vti_script" />
   <exclude path="_vti_txt" />
   <exclude path="Web References" />
</dynamicDiscovery>
```

Sau khi một ứng dụng Client phát hiện ra một dịch vụ Web, các ứng dụng Client có thể truy cập và sử dụng tất cả các phương thức giao tiếp với một dịch vụ Web. Quá trình hoạt động bằng cách sử dụng một phương pháp giao tiếp của một dịch vụ web được gọi là tiêu thụ một dịch vụ Web. Bạn sẽ tìm hiểu về tiêu thụ một dịch vụ Web trong bài học kế tiếp.

2.1.4. Tiêu Thụ a service Web XML(Consuming an XML Web Service)

Trong bài trước, bạn đã học về cơ chế dịch vụ Web XML phát hiện và làm thế nào để cấu hình thông tin phát hiện ra cho một dịch vụ Web XML. Sau khi một ứng dụng Client phát hiện ra một dịch vụ Web XML, ứng dụng Client có thể truy cập các dịch vụ được cung cấp bởi nó. Này được gọi là tiêu thụ một dịch vụ Web. Trong bài học này, bạn sẽ tìm hiểu về tiêu thụ một dịch vụ Web XML. Ngoài ra, bạn sẽ tìm hiểu làm thế nào để truy cập vào các dịch vụ được cung cấp bởi một dịch vụ Web XML.

-

Tiêu thụ một dịch vụ Web XML

Sau khi bạn tạo ra một dịch vụ Web XML và xuất bản nó, bất kỳ ứng dụng có quyền truy cập các dịch vụ Web có thể truy cập dịch vụ Web XML của bạn và tiêu thụ dịch vụ của mình. Các ứng dụng mà tiêu thụ một dịch vụ web được biết đến như là Web Service trên Client. Một dịch vụ web trên Client có thể là một thành phần, dịch vụ, hoặc các ứng dụng máy tính để bàn. Trong thực tế, các dịch vụ web Client phổ biến nhất có thể tiêu thụ các dịch vụ Web là các ứng dụng Web và dịch vụ Web.

Để truy cập một dịch vụ Web XML từ một ứng dụng của Client, bạn cần phải thực hiện các nhiệm vụ sau đây:

-Thêm một tài liệu tham khảo web để các dịch vụ Web XML trong các ứng dụng của Client bằng cách khám phá các dịch vụ Web XML mà bạn muốn tiêu thụ.

-Tạo ra một lớp proxy cho các dịch vụ Web XML.

-Tạo một đối tượng của lớp XML Web proxy trong các ứng dụng của Client.

-Truy cập các dịch vụ Web XML bằng cách sử dụng một đối tượng proxy.

Bây giờ bạn sẽ nhìn vào những công việc chi tiết.

Thêm một Web Reference

Để sử dụng các dịch vụ của một dịch vụ Web XML, trước tiên bạn xác định vị trí các dịch vụ Web XML bằng cách sử dụng cơ chế phát hiện. Để sử dụng một dịch vụ Web XML được tạo ra bởi lập trình viên khác, bạn nên biết vị trí của các dịch vụ Web XML và xem các dịch vụ Web có đáp ứng yêu cầu của

bạn hay không. Nếu bạn không biết nơi dịch vụ, bạn có thể tìm kiếm cho nó thông qua UDDI. Để làm cho nó dễ dàng để phát hiện ra một dịch vụ Web XML, NET Studio Visual cung cấp một tài liệu tham khảo Web cho mỗi dịch vụ Web XML mà bạn sử dụng trong một dự án. Một tài liệu tham khảo Web là các đại diện địa phương (Local) của một dịch vụ Web XML trong dự án của bạn.

Bạn có thể thêm một tài liệu tham khảo web để dự án của bạn bằng cách sử dụng hộp thoại Add Web Reference. Hộp thoại này cho phép bạn duyệt máy chủ địa phương của bạn, mục Microsoft UDDI, và Web để xác định vị trí một dịch vụ Web. Web Thêm hộp thoại tham khảo sử dụng cơ chế phát hiện dịch vụ Web để xác định vị trí các dịch vụ Web XML trên các trang web mà bạn cung cấp. Web Add Reference hộp thoại yêu cầu hộp URL cho các mô tả các dịch vụ Web XML trên trang web và hiển thị nó. Sau khi bạn chọn một dịch vụ Web XML, bạn nhấp vào Add Reference để thêm một tài liệu tham khảo web trong dự án của bạn.

Tạo một lớp proxy

Khi bạn nhấn vào nút Add Reference trong hộp thoại Add Web Reference, Visual Studio. NET tải mô tả dịch vụ máy tính địa phương và tạo ra một lớp proxy cho các dịch vụ Web. Lớp proxy của một dịch vụ Web XML có chứa các hướng dẫn để kêu gọi mỗi phương pháp dịch vụ Web XML. Ngoài ra, lớp proxy thống chế tranh cãi giữa một dịch vụ Web XML và ứng dụng của Client. NET Studio Visual sử dụng WSDL để tạo ra các lớp proxy. Lớp proxy được mô tả trong các tập tin WSDL. Sau khi bạn thêm một tài liệu tham khảo Web với một dịch vụ Web XML và tạo ra một lớp proxy, bạn thêm một tham chiếu đến các lớp proxy và tạo một đối tượng của lớp proxy trong ứng dụng của Client.

Tuy nhiên, nếu bạn không thể để xác định vị trí một dịch vụ Web XML bằng cách sử dụng hộp thoại Add Web Reference, bạn có thể sử dụng công cụ WSDL, wsdl.exe, để tạo ra một lớp proxy thủ công cho các dịch vụ Web XML.

Các công cụ WSDL có thể mất một hợp đồng WSDL, giản đồ XSD, hoặc một tài liệu DiscoMap. Như một tham số và tạo ra một lớp proxy.

Bạn thực hiện các công cụ WSDL từ dấu nhắc lệnh. Cú pháp để thực hiện các công cụ WSDL từ dấu nhắc lệnh

Wsdl [options] {URL Path}

Bảng 7.1 cho thấy một số tùy chọn mà bạn có thể sử dụng với lệnh WSDL.

Các lệnh sau đây tạo ra một tập tin WSDL và một lớp proxy client, WebService.cs trong Visual C # cho các dịch vụ Web đặt tại URL được chỉ định.

Wsdl http://hostserver/MyApp/WebService.asmx wsdl

Bạn có thể thay đổi ngôn ngữ của tập tin đầu ra WebService.cs bằng cách sử dụng chuyển đổi / ngôn ngữ. Cú pháp để tạo ra một lớp proxy trong NET Visual. Basic

Wsdl / ngôn ngữ: VB http://hostserver/MyApp/WebService.asmx wsdl

TIP

Sau khi bạn tạo mã nguồn cho lớp proxy, bạn biên dịch nó sang assembly của dự án hiện tại, hoặc bạn có thể biên dịch lớp proxy vào assembly riêng của mình và cài đặt nó trong bộ nhớ cache assembly tất cả sử dụng.

Hình.7-1. Một số tùy chọn của Wsdl Command

Option	Description
/baseurl:baseurl	Chỉ định URL cơ sở để sử dụng khi tính toán đoạn

	URL. Các công cụ tính toán đoạn URL bằng cách chuyển đổi các URL có liên quan từ đối số baseUrl tới URL trong tài liệu WSDL. Bạn phải chỉ định tùy chọn / appsettingurlkey với tùy chọn này.
/d[omain]:domain	Chỉ định tên miền để sử dụng khi kết nối với một máy chủ mà yêu cầu xác thực
/l[anguage]:language	Chỉ định ngôn ngữ để sử dụng cho lớp proxy được tạo ra. Bạn có thể chỉ định CS (C #; mặc định), VB (Visual Basic), hoặc JS (JScript) như là đối số ngôn ngữ.
/n[amespace]:namespace	Chỉ định không gian tên cho các proxy được tạo ra hoặc mẫu. Các không gian tên mặc định là không gian tên toàn cầu.
/o[ut]:filename	Xác định tập tin trong đó để lưu mã proxy được tạo ra. Công cụ này có nguồn gốc tên tập tin mặc định từ tên dịch vụ Web XML. Các công cụ tiết kiệm bộ dữ liệu được tạo ra trong các tập tin khác nhau.
/protocol:protocol	Chỉ định giao thức để thực hiện. Bạn có thể chỉ định SOAP (mặc định), HttpGet, HttpPost, hoặc giao thức tùy chỉnh quy định trong file cấu hình.
/?	Hiển thị cú pháp lệnh và các tùy chọn cho công cụ này.
/proxy:URL	Chỉ định URL của máy chủ proxy để sử dụng cho các yêu cầu HTTP. Mặc định là để sử dụng cài đặt proxy hệ thống.

Tạo một đối tượng của class Proxy

Lớp proxy được tạo ra bằng cách thêm một tham chiếu đến một dịch vụ Web XML được định nghĩa trong không gian tên riêng của mình. Bạn cần phải bao gồm các không gian tên lớp proxy trong ứng dụng khách hàng của bạn trước khi bạn có thể tạo một đối tượng của lớp proxy. Sau khi bạn bao gồm các không gian tên lớp proxy trong ứng dụng khách hàng của bạn, bạn tạo một đối tượng của lớp proxy bằng cách sử dụng các nhà điều hành mới. Các đối tượng của lớp proxy cho phép bạn gọi các phương thức tiếp xúc với một dịch vụ Web XML và truy cập kết quả giống như bất kỳ phương pháp khác của một thành phần.

Truy cập các dịch vụ Web sử dụng một đối tượng Proxy

Các đối tượng của lớp proxy chuyển đổi các phương pháp kêu gọi một dịch vụ Web XML để một tin nhắn yêu cầu và thông điệp phản ứng với một giá trị trả về, mà bạn có thể truy cập trong ứng dụng của khách hàng.

Sau khi bạn tạo một đối tượng của lớp proxy, bạn có thể dễ dàng viết mã chống lại một dịch vụ Web XML. NET Studio Visual cung cấp IntelliSense và an toàn kiểu khi bạn sử dụng một đối tượng của lớp proxy để truy cập vào các phương pháp của một dịch vụ Web.

Tiêu thụ các phương thức giao tiếp bởi một dịch vụ Web

Trước đây bạn đã học về quá trình tiêu thụ một dịch vụ Web XML. Bây giờ bạn sẽ nhìn vào tiêu thụ một dịch vụ Web XML bằng cách sử dụng một dịch vụ Web XML mẫu, trong đó có bốn phương pháp Web cho phép bạn thực hiện các phép tính số học đơn giản như cộng, trừ, nhân, và phân chia của hai số nguyên. Tất cả những phương pháp này Web lấy hai số nguyên như là các tham số. Các mã sau đây cho thấy một mẫu dịch vụ Web XML.

Visual Basic .NET

```
Imports System.Web.Services


<WebService(Namespace:="http://tempuri.org/")> _
Public Class MyServiceVB
    Inherits System.Web.Services.WebService


    <WebMethod()> Public Function AddNumbers(ByVal Num1 As Integer, _
        ByVal Num2 As Integer) As Integer
        AddNumbers = Num1 + Num2
    End Function
    <WebMethod()> Public Function SubNumbers(ByVal Num1 As Integer, _
        ByVal Num2 As Integer) As Integer
        SubNumbers = Num1 - Num2
    End Function
    <WebMethod()> Public Function MulNumbers(ByVal Num1 As Integer, _
        ByVal Num2 As Integer) As Integer
        MulNumbers = Num1 * Num2
    End Function
    <WebMethod()> Public Function DivNumbers(ByVal Num1 As Integer, _
        ByVal Num2 As Integer) As Integer
        DivNumbers = Num1 / Num2
    End Function
End Class
```

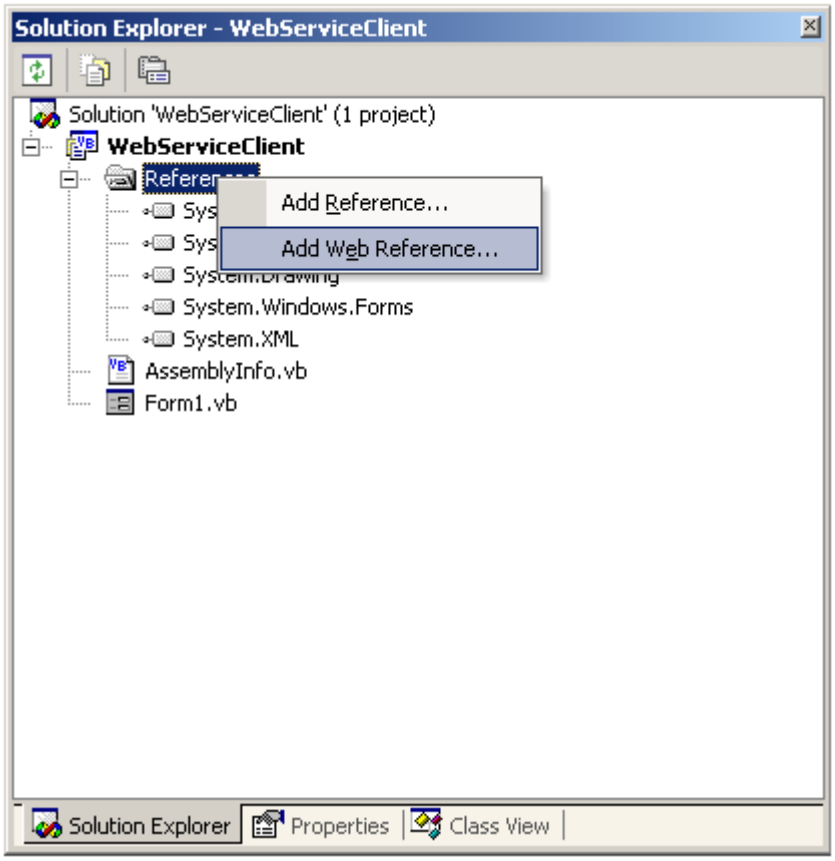
Visual C#

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace MyWebServiceCS
{
   public class MyServiceCS : System.Web.Services.WebService
   {
      public MyServiceCS()
      {
      }

      [WebMethod]
      public int AddNumbers(int num1, int num2)
      {
         return (num1+num2);
      }
      [WebMethod]
      public int SubNumbers(int num1, int num2)
      {
         return (num1-num2);
      }
```

```
    [WebMethod]
    public int MulNumbers(int num1, int num2)
    {
        return (num1*num2);
    }
    [WebMethod]
    public int DivNumbers(int num1, int num2)
    {
        return (num1/num2);
    }

  }
}
```

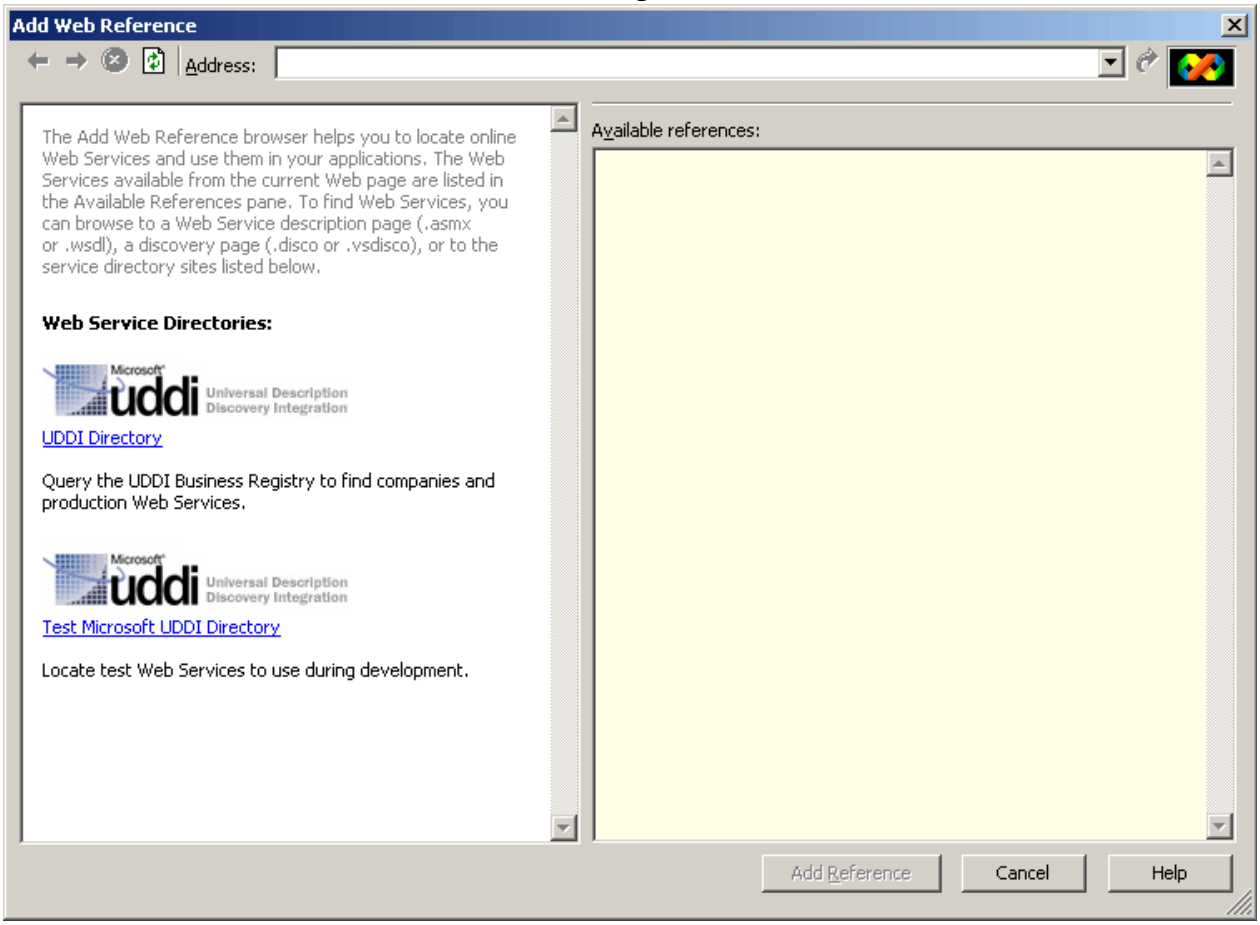
Nhiệm vụ đầu tiên trong việc tiêu thụ một dịch vụ Web XML là thêm một tài liệu tham khảo Web cho các Client của khách hàng bằng cách sử dụng hộp thoại Add Web Reference. Để mở hộp thoại Add Web Reference, thực hiện các bước sau đây.

1)Kích chuột phải vào nút References trong cửa sổ Solution Explorer, như thể hiện trong hình 7,10.



Hình 7-10. The Solution Explorer window

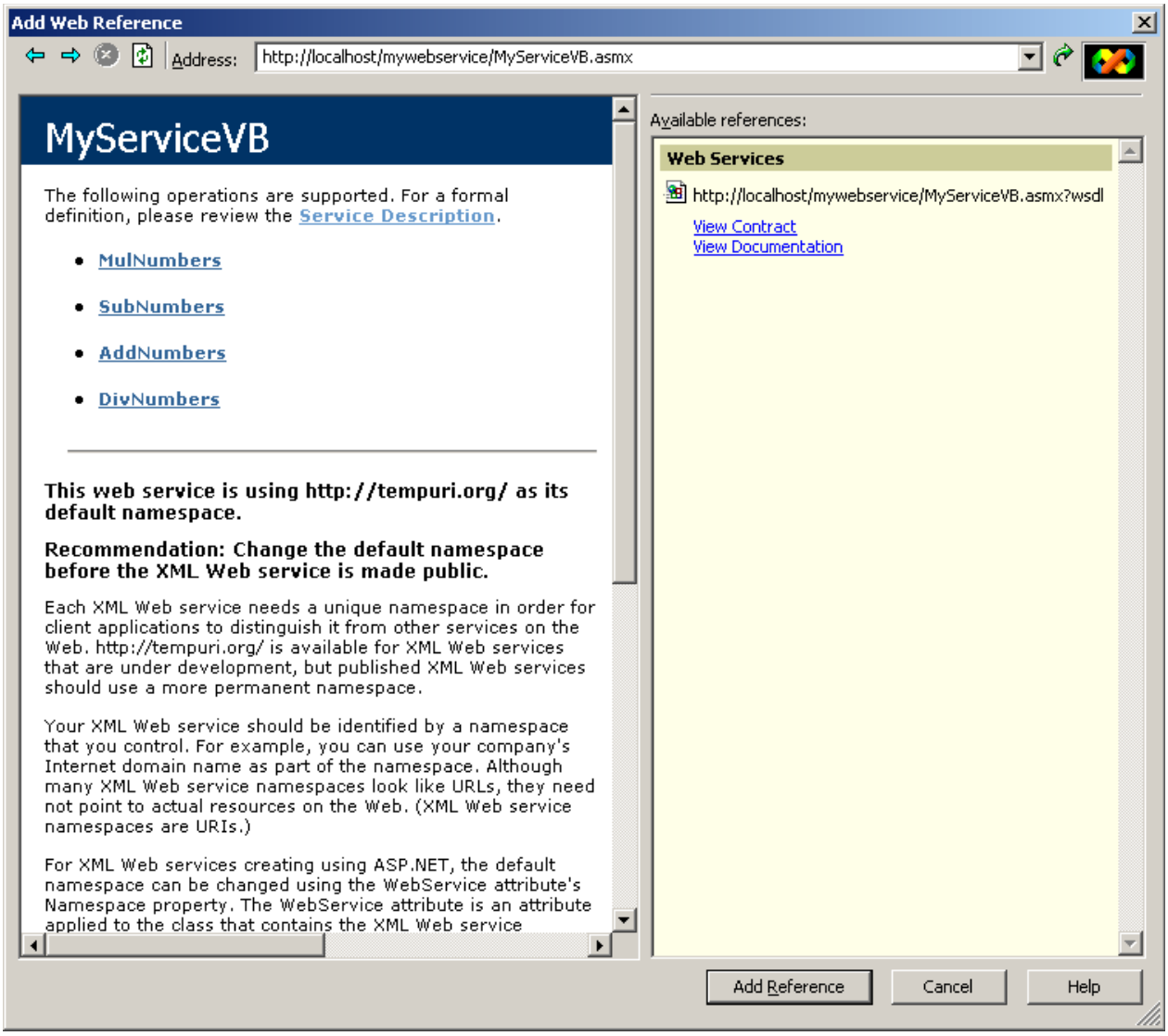
2) Chọn Add Web Reference từ menu chuột phải. Tham khảo hộp thoại Add Web Reference mở ra, như thể hiện trong Hình 7,11.



Hình 7-11. The Add Web Reference dialog box

Trong thanh địa chỉ của hộp thoại Add Web Reference, chỉ định đường dẫn của dịch vụ Web XML mà bạn muốn sử dụng. Khi hộp thoại tham khảo Web Tiện phát hiện ra các dịch vụ Web XML mà bạn đang tìm kiếm, nó sẽ hiển thị thông tin về các dịch vụ Web XML. Thông tin này cũng bao gồm các thông tin chi tiết về các phương pháp mà các dịch vụ Web XML phơi bày.

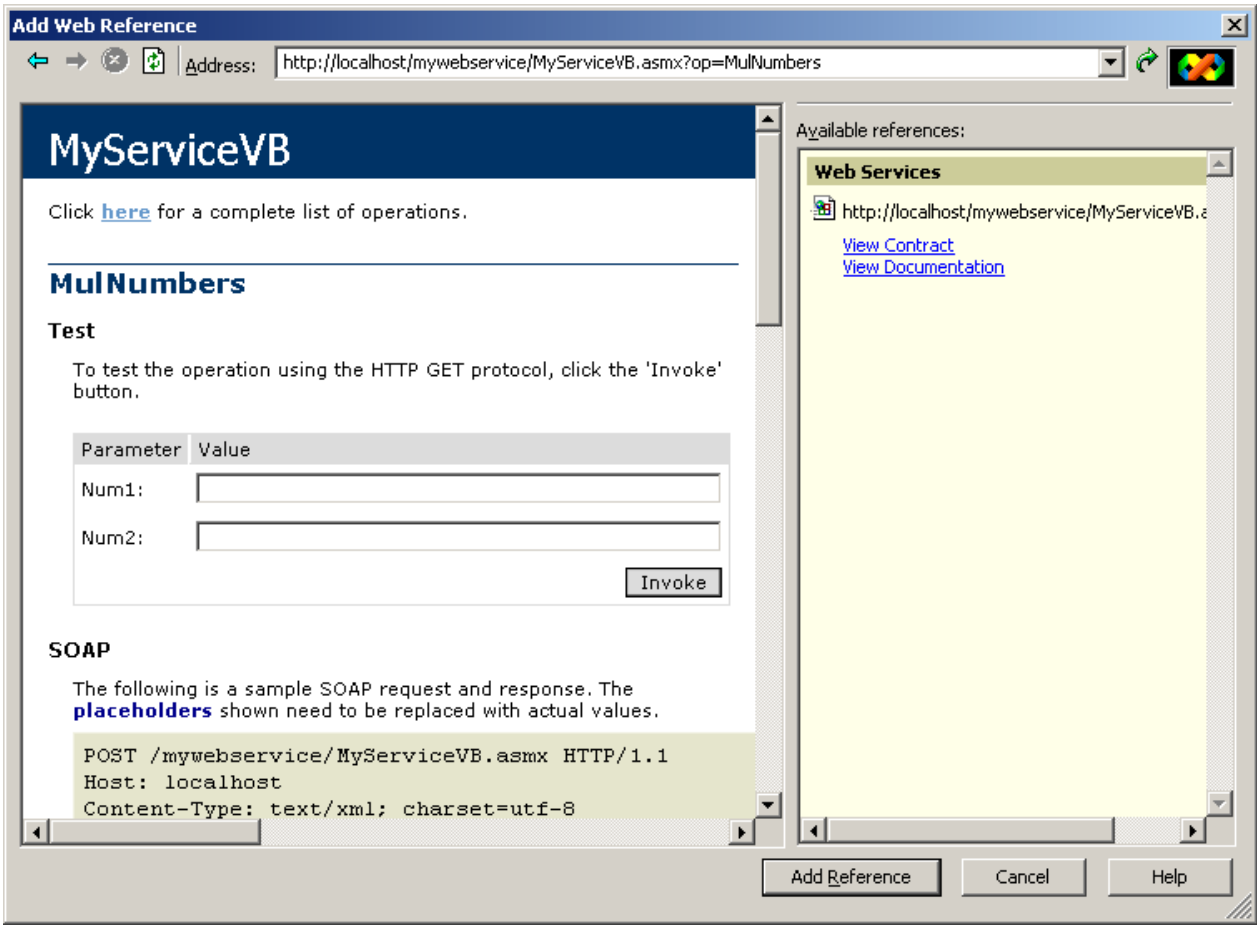
Hình 7,12 cho thấy các chi tiết của một dịch vụ Web XML trong hộp thoại Add Web Reference.



Hình 7-12. Details of a Web service displayed in the Add Web Reference dialog box

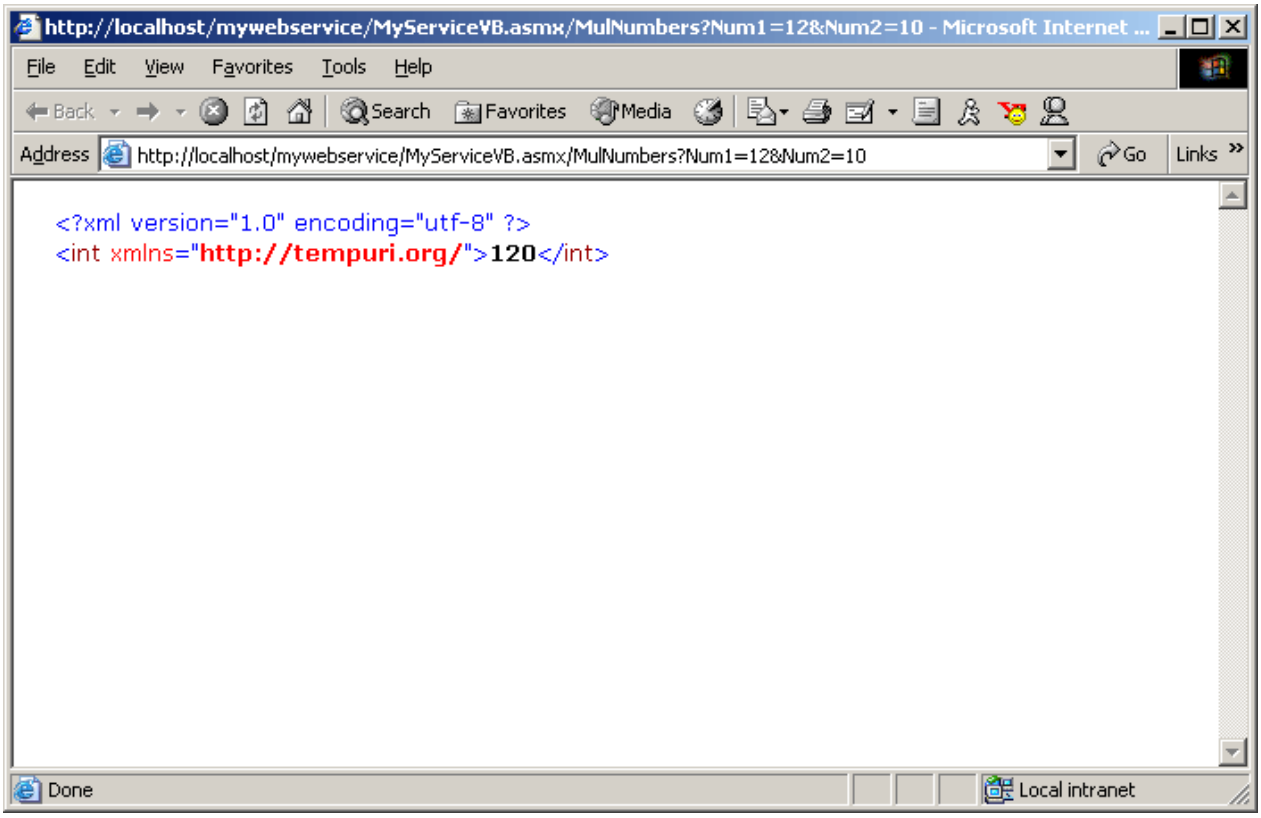
Bạn cũng có thể xem chi tiết, chẳng hạn như các thông số phương pháp, cho mỗi phương pháp tiếp xúc bởi các dịch vụ Web XML bằng cách nhấp chuột vào một phương pháp trong hộp thoại Add Web Reference. Bạn cũng có thể thử nghiệm một dịch vụ Web XML bằng cách sử dụng hộp thoại này trước khi sử dụng nó trong ứng dụng. Để thử nghiệm một phương pháp tiếp xúc của một dịch vụ Web XML, thực hiện các bước sau:

1) Nhấp một phương thức từ danh sách các phương pháp trong hộp thoại Add Web Reference. Hình 7,13 hiển thị trang được tạo ra tự động bởi Visual Studio NET để thử nghiệm một phương pháp tiếp xúc của một dịch vụ Web XML.



Hình 7-13. Testing an exposed method of an XML Web service

2)Nhập các thông số thích hợp cho các phương thức và nhấn Invoke. Kết quả của thử nghiệm được hiển thị trong một cửa sổ Internet Explorer, như thể hiện trong hình 7,14.



Hình 7-14. The results of testing an exposed method of an XML Web service

Ngoài ra, Add Web Reference cho phép bạn xem và đáp ứng yêu cầu SOAP, HTTP-GET yêu cầu và phản ứng, và yêu cầu HTTP POST và đáp ứng cho một phương thức dịch vụ Web XML. Các đoạn mã sau đây hiển thị và đáp ứng yêu cầu SOAP, HTTP-GET yêu cầu và phản ứng, và yêu cầu HTTP POST và phản ứng đối với một phương thức giao tiếp của một dịch vụ Web XML.

SOAP

```
POST /mywebserviceCS/MyServiceCS.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/MulNumbers"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <MulNumbers xmlns="http://tempuri.org/">
         <num1>int</num1>
         <num2>int</num2>
      </MulNumbers>
   </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <MulNumbersResponse xmlns="http://tempuri.org/">
         <MulNumbersResult>int</MulNumbersResult>
      </MulNumbersResponse>
   </soap:Body>
</soap:Envelope>
```

Lưu ý

Các giao thức SOAP cho phép các ứng dụng để trao đổi thông tin có cấu trúc và đánh máy trên web bằng cách sử dụng các tiêu chuẩn dựa trên XML. Khi một ứng dụng client liên lạc với một dịch vụ Web XML bằng cách sử dụng SOAP, các dữ liệu được trao đổi giữa các ứng dụng của khách hàng và các dịch vụ Web XML theo một định dạng chuẩn. Định dạng này bao gồm các dữ liệu được mã hóa trong một tài liệu XML. Các tài liệu XML bao gồm một yếu tố Envelope gốc, mà lần lượt bao gồm một yếu tố trên cơ thể bắt buộc. Các phần tử Body có chứa các dữ liệu cụ thể vào tin nhắn. Các yếu tố Envelope có thể chứa một phần tử Header tùy chọn, trong đó có thông tin bổ sung không trực tiếp liên quan đến tin nhắn. Mỗi phần tử con của phần tử Tiêu đề được gọi là một tiêu đề SOAP.

HTTP-GET

```
GET /mywebserviceCS/MyServiceCS.asmx/MulNumbers?num1=string&num2=
    string HTTP/1.1
Host: localhost
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">int</int>
```

HTTP-POST

```
POST /mywebserviceCS/MyServiceCS.asmx/MulNumbers HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length
num1=string&num2=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">int</int>
```

Cuối cùng, nhấn vào nút Add Reference để thêm một tham chiếu đến các dịch vụ Web XML vào ứng dụng Client của bạn. NET Studio Visual thêm một nút Web References, trong đó có tên giống như máy tính cung cấp dịch vụ Web XML, ứng dụng Client của bạn. Bạn có thể đổi tên các tài liệu tham khảo web để thay đổi không gian tên của lớp proxy được tạo ra. Khi bạn nhấn vào Add Reference, NET Visual Studio bao gồm một tài liệu tham khảo web và thêm các tập tin sau đây để ứng dụng Client của bạn:

.wsdl

.disco

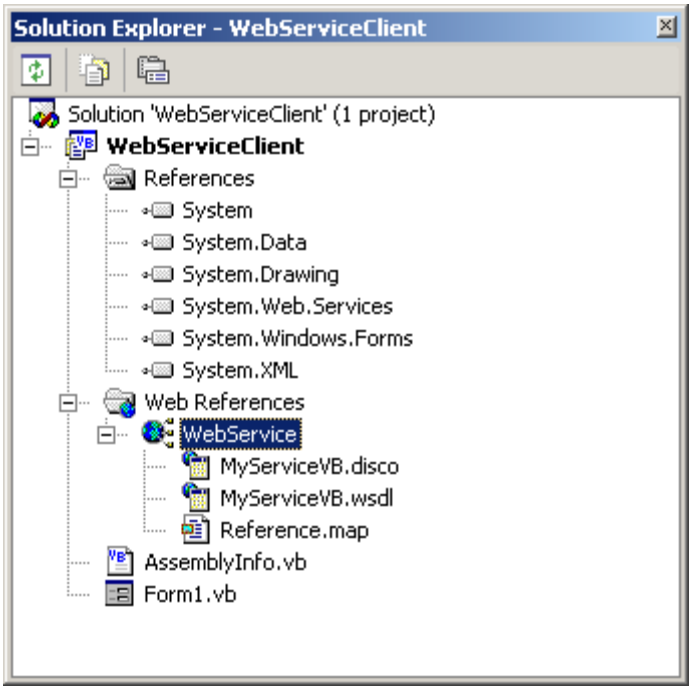
.map

Hình 7,15 hiển thị một nút Web References.

Tập tin wsdl chứa các lớp proxy tạo ra bởi NET Studio Visual. Như đã mô tả ở trên, lớp proxy có chứa các hướng dẫn để kêu gọi các phương pháp dịch vụ Web XML. Ngoài ra, lớp proxy điều hành giữa một dịch vụ Web XML và ứng

dụng một Client. Để xem wsdl file. NET Studio Visual tạo ra, mở rộng nút Web Các tài liệu tham khảo thuộc dự án của bạn trong Solution Explorer. Dưới nút Web Các tài liệu tham khảo, xác định vị trí các nút localhost.

Nút này có chứa không gian tên có chứa các lớp proxy được tạo ra bởi NET Studio Visual. Nút localhost chứa WSDL, disco., Và các tập tin Reference.map. Bạn có thể thay đổi không gian tên localhost mặc định bằng cách kích chuột phải vào localhost và cách nhấp vào menu chuột phải chọn Rename.



Hình 7-15. A Web References node in a client application

Sau khi bạn tạo ra các lớp proxy, bạn cần tạo một đối tượng của lớp proxy để truy cập vào các phương thức giao tiếp của một dịch vụ Web XML. Để tham khảo một lớp proxy trong mã của bạn, bạn cần phải cung cấp tên đầy đủ của lớp proxy. Nừu ứng dụng của bạn trong không gian tên MyApps, và lớp proxy, MyWebService, trong không gian tên WebServices, sau đó trong Visual Basic. NET, bạn phải tham khảo đến lớp MyWebService như WebServices.MyWebService. Tuy nhiên, trong Visual C #, bạn phải tham khảo

đến lớp MyWebService như MyApps.WebServices.MyWebService. Các mã sau đây sẽ hiển thị như thế nào để tạo ra một đối tượng của lớp proxy.

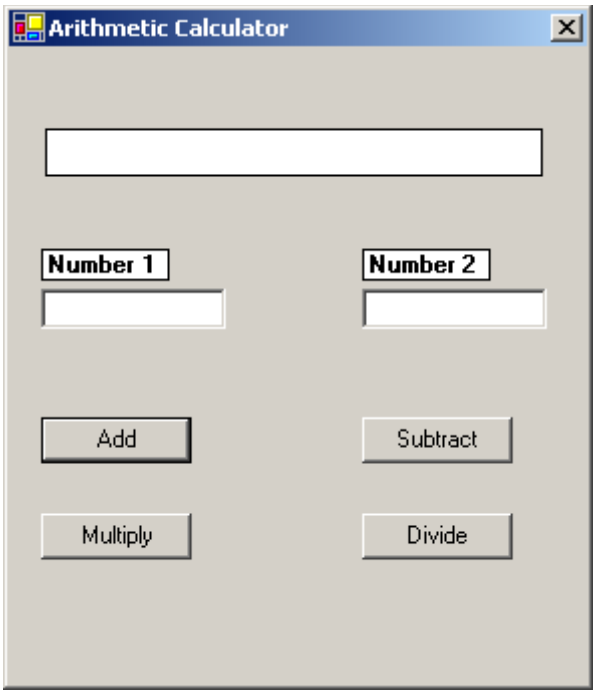
Visual Basic .NET

```
Private VBService As New WebServices.MyWebServiceVB()
```

Visual C#

```
private MyApps.WebServices.MyWebServiceCS CSService = new
   MyApps.WebServices.MyWebServiceCS();
```

Cuối cùng, bạn có thể truy cập vào các phương thức giao của một dịch vụ Web XML bằng cách sử dụng các đối tượng lớp proxy. Hình 7,16 hiển thị giao diện của một ứng dụng, trong đó tiêu thụ các phương thức giao tiếp bởi các dịch vụ Web XML.



Hình 7-16. The client application interface

Các mã sau đây sẽ hiển thị mã mà ứng dụng của Client sử dụng để truy cập vào các phương thức giao tiếp của một dịch vụ Web XML. Trong đoạn mã

sau, các phương thức giao tiếp với các dịch vụ Web XML được gọi là bên trong các nút bấm mã xử lý sự kiện của giao diện khách hàng.

Visual Basic .NET

```
Private Sub BtnAdd_Click(ByVal sender As System.Object, ByVal e As _
   System.EventArgs) Handles BtnAdd.Click
   Dim Result as Integer = VBService.AddNumbers(Val(TxtNum1.Text), _
      Val(TxtNum2.Text))
   LblResult.Text = Result.ToString
End Sub

Private Sub BtnSub_Click(ByVal sender As System.Object, ByVal e As _
   System.EventArgs) Handles BtnSub.Click
   Dim Result as Integer = VBService.SubNumbers(Val(TxtNum1.Text), _
      Val(TxtNum2.Text))
   LblResult.Text = Result.ToString
End Sub

Private Sub BtnMul_Click(ByVal sender As System.Object, ByVal e As _
   System.EventArgs) Handles BtnMul.Click
   Dim Result as Integer= VBService.MulNumbers(Val(TxtNum1.Text), _
      Val(TxtNum2.Text))
   LblResult.Text = Result.ToString
End Sub

Private Sub BtnDiv_Click(ByVal sender As System.Object, ByVal e As _
   System.EventArgs) Handles BtnDiv.Click
   Dim Result as Integer = VBService.DivNumbers(Val(TxtNum1.Text), _
      Val(TxtNum2.Text))
   LblResult.Text = Result.ToString
End Sub
```

Visual C#

```
private void BtnAdd_Click(object sender, System.EventArgs e)
{
   int result = CSService.AddNumbers(Int32.Parse(TxtNum1.Text),
      Int32.Parse(TxtNum2.Text));
```

```
    LblResult.Text= result.ToString();
}

private void BtnSub_Click(object sender, System.EventArgs e)
{
    int result = CSService.SubNumbers(Int32.Parse(TxtNum1.Text),
        Int32.Parse(TxtNum2.Text));
    LblResult.Text= result.ToString();
}

private void BtnMul_Click(object sender, System.EventArgs e)
{
    int result = CSService.MulNumbers(Int32.Parse(TxtNum1.Text),
        Int32.Parse(TxtNum2.Text));
    LblResult.Text= result.ToString();
}

private void BtnDiv_Click(object sender, System.EventArgs e)
{
    int result = CSService.DivNumbers(Int32.Parse(TxtNum1.Text),
        Int32.Parse(TxtNum2.Text));
    LblResult.Text= result.ToString();
}
```

Khi một ứng dụng khách hàng tiêu thụ một dịch vụ Web XML, ứng dụng client gọi phương thức web trên các dịch vụ Web XML và vượt qua các đối số nếu phương thức web đòi hỏi họ. Các phương thức web cũng có thể trả về một giá trị cho Client. Những lập luận rằng các Client gửi đến một dịch vụ Web XML và dịch vụ Web XML trả về các giá trị được vận chuyển qua mạng. Trước khi các đối tượng hoặc các giá trị có thể được vận chuyển qua mạng, họ cần phải được chuyển đổi thành một định dạng mà có thể được vận chuyển. Trong quá trình đó các đối tượng và các giá trị được chuyển đổi sang một định dạng thích hợp cho giao thông vận tải được gọi là serialization.

Serialization

Tuần tự hóa là quá trình mà các đối tượng hoặc các giá trị được chuyển đổi sang một định dạng mà có thể được tiếp tục tồn tại và vận chuyển. Serialization cho phép bạn lưu các trạng thái của các đối tượng từ bộ nhớ đến một phương tiện lưu trữ như các tập tin. Bạn cũng có thể sử dụng tuần tự để vận chuyển các đối tượng và các giá trị trên mạng. Để đọc trạng thái của các đối tượng hoặc các giá trị mà bạn vẫn tồn tại hoặc vận chuyển sử dụng serialization, bạn sử dụng một quá trình được gọi là deserialization. Deserialization là bổ sung cho serialization. NET Framework hỗ trợ hai loại serialization, nhị phân serialization và XML serialization.

Serialization nhị phân chuyển đổi trạng thái của một đối tượng bằng cách viết các lĩnh vực công cộng và tư nhân, tên của lớp, và lắp ráp có chứa các lớp thành một dòng các byte. Để làm cho đối tượng của bạn serializable, bạn cần phải đánh dấu các lớp học với các thuộc tính Serializable như thể hiện trong các mã sau đây.

Visual Basic .NET

```
<Serializable()> _
Public Class MySerializableClass
   Public n1 as Integer
   Public s1 as String
   ' Implement the class
End Class
```

Visual C#

```
[Serializable()]
public class MySerializableClass {
   public int n1;
```

```
    public string s1;
    // Implement the class
}
```

Điều quan trọng cần lưu ý rằng các thuộc tính Serializable không phải là thừa kế. Đó là, nếu một lớp học, MyDerivedClass, xuất phát từ MySerializableClass lớp trong mã trước đây, lớp MyDerivedClass không trở thành serializable. Bạn cần phải rõ ràng đánh dấu MyDerivedClass với các thuộc tính Serializable để làm cho nó tuần tự. Các mã sau đây cho thấy làm thế nào để serialize một đối tượng của MySerializableClass.

Visual Basic .NET

```
Imports System.IO
Imports System.Runtime.Serialization.Formatters.Binary

Public Class SerializeDemo
    Public Shared Sub Main()
        Dim obj as New MySerializableClass()
        obj.n1 = 10
        obj.s1 = "Hello"
        Dim stream As New FileStream("MyFile.dat", FileMode.Create)
        Dim formatter As New BinaryFormatter()
        formatter.Serialize(stream, obj)
    End Sub
End Class
```

Visual C#

```
using System;
```

```
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

public class SerializeDemo
{
   [STAThread]
   static void Main(string[] args)
   {
      MySerializableClass obj = new MySerializableClass();
      obj.n1 = 10;
      obj.s1 = "Hello";
      FileStream stream = new FileStream("MyFile.dat",
         FileMode.Create);
      BinaryFormatter formatter = new BinaryFormatter();
      formatter.Serialize(stream, obj);
   }
}
```

XML serialization chuyển đổi các lĩnh vực công cộng và tài sản của một đối tượng, thông số, và giá trị trả lại các phương pháp vào một luồng XML phù hợp với một tài liệu XML cụ thể định nghĩa schema (XSD). Để tạo ra các lớp học có thể được tuần tự bằng cách sử dụng XML serialization, bạn sử dụng công cụ XML Schema Definition (XSD.exe). Để serialize và deserialize đối tượng bạn sử dụng lớp XmlSerializer. Dịch vụ Web XML được tạo ra bằng cách sử dụng ASP.NET sử dụng lớp XmlSerializer để tạo ra dòng XML truyền dữ liệu giữa các ứng dụng dịch vụ Web. Các mã sau đây cho thấy làm thế nào để sử dụng XmlSerializer để tuần tự hóa một đối tượng của MySerializableClass.

Visual Basic .NET

```
Imports System.IO
Imports System.Xml.Serialization


Public Class SerializeDemo
   Public Shared Sub Main()
      Dim obj As New MySerializableClass()
      obj.n1 = 10
      obj.s1 = "Hello"
      Dim stream As New FileStream("MyFile.xml", FileMode.Create)
      Dim formatter As New XmlSerializer(GetType(MySerializableClass))
      formatter.Serialize(stream, obj)
   End Sub
End Class
```

Visual C#

```
using System;
using System.IO;
using System.Xml.Serialization;
using System.Reflection;


public class SerializeDemo
{
   [STAThread]
   static void Main(string[] args)
   {
```

```
        MySerializableClass obj = new MySerializableClass();
        obj.n1 = 10;
        obj.s1 = "Hello";
        FileStream stream = new FileStream("MyFile.xml",
            FileMode.Create);
        XmlSerializer formatter = new XmlSerializer(obj.GetType());
        formatter.Serialize(stream, obj);
    }
}
```

Tóm tắt nội dung của bài

Dịch vụ Web XML là những thành phần chương trình cho phép bạn xây dựng khả năng mở rộng, mềm dẻo trong các ứng dụng nền tảng độc lập. XML Web dịch vụ cho phép các ứng dụng khác nhau để trao đổi tin nhắn bằng cách sử dụng các giao thức chuẩn như HTTP, XML, XSD, SOAP, WSDL.

Bạn tạo ra một dịch vụ Web XML cung cấp chức năng cụ thể cho các ứng dụng của Client thông qua web. Studio Visual IDE cung cấp cho bạn với môi trường cần thiết để tạo ra các dự án để phân phối các ứng dụng máy tính, các ứng dụng Web, và các dịch vụ Web XML.

Sau khi bạn tạo ra một dịch vụ Web XML, bạn cần để triển khai các dịch vụ Web XML đến một máy chủ Web để làm cho nó có sẵn cho các ứng dụng mà muốn sử dụng các dịch vụ Web XML. Bạn có thể triển khai một dịch vụ Web XML đến một máy chủ Web bằng cách sao chép các tập tin dịch vụ Web XML đến máy chủ Web.

XML Web dịch vụ khám phá ra cơ chế cho phép một ứng dụng Client để xác định vị trí hoặc khám phá những tài liệu mô tả một dịch vụ Web XML. XML

Web dịch vụ khám phá cơ chế trả về một tài liệu mô tả dịch vụ cho Client. Tài liệu mô tả của dịch vụ được viết trong WSDL và chứa các thông tin về khả năng của các dịch vụ Web XML, vị trí của nó, và làm thế nào để tương tác với nó.

Sau khi bạn tạo ra một dịch vụ Web XML và xuất bản nó, bất kỳ ứng dụng có quyền truy cập vào nó có thể truy cập dịch vụ Web XML của bạn và tiêu thụ dịch vụ của mình. Các ứng dụng mà tiêu thụ một dịch vụ web được biết đến như là dịch vụ Web trên Client. Một dịch vụ web trên Client có thể là một thành phần, dịch vụ, hoặc các ứng dụng máy tính.

2.2. Trình tự thao tác

2.2.1Tạo và triển khai hoạt động các dịch vụ Web XML

Trong phần này, bạn sẽ tạo ra một dịch vụ Web XML cho phép bạn xây dựng, phân phối, giải pháp truy cập dữ liệu. Lab này sẽ mô phỏng một hệ thống đặt phòng hãng hàng không cho phép bạn xem các thông tin chuyến bay và đặt chỗ chuyến bay từ một máy tính cục bộ hoặc từ xa. Hệ thống này bao gồm những phần sau đây:

-Cơ sở dữ liệu hàng không có chứa hai bảng, vé máy bay và đặt chỗ(Flights and Bookings), để lưu trữ thông tin về đặt phòng khách chuyến bay

-Một dịch vụ Web XML cung cấp thông tin chuyến bay cho khách hàng

-Một ứng dụng Client Windows mà làm việc với các dịch vụ Web XML

Tạo một dịch vụ Web XML

Bạn sẽ tạo ra một dịch vụ Web XML cho phép các ứng dụng của Client để truy cập và sửa đổi thông tin về đặt vé máy bay trong cơ sở dữ liệu hàng không. Để tạo ra các dịch vụ Web XML, thực hiện các bước sau:

1) Mở Visual Studio. NET.

2)Chọn New Project từ trình đơn menu File. Hộp thoại sẽ New Project mở ra.

3)Chọn ASP.NET Web Service từ cửa sổ Templates. Nhập http://localhost/AirlineServices trong trường Location.

4)Xóa file Service1.asmx. Kích chuột phải vào dự án AirlineServices, sau đó chọn Add Web Service để mở hộp thoại Add New Item.

5)Trong hộp thoại Add New Item, bạn viết vào tên FlightBooking.asmx, và kích Open.

6)Mở xem mã lệnh của tập tin FlightBooking.asmx.

7)Nhập đoạn mã sau cho lớp FlightBooking.

Visual Basic .NET

```
Imports System.Web.Services
Imports System.Data
Imports System.Data.SqlClient

<WebService(Namespace := "http://tempuri.org/")> _
Public Class FlightBooking
    Inherits System.Web.Services.WebService

    Private flights_data As DataTable
    Private dataadapter As SqlDataAdapter
    Private connection As SqlConnection
    Private bookings_data As DataTable
    Private dataadapter1 As SqlDataAdapter
    Private flt_bkgs As DataSet
```

```
#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()
        ' Initialize the SQL connection
        connection = New SqlConnection("server=localhost;user id=sa;" & _
            "password=;initial catalog=Airline")

        ' Create a DataTable to store flights info
        dataadapter = New SqlDataAdapter("select * from flights", _
            connection)
        Dim commandbuilder As New SqlCommandBuilder(dataadapter)
        dataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        flights_data = New DataTable("Flights")
        dataadapter.Fill(flights_data)

        ' Create a DataTable to store bookings info
        dataadapter1 = New SqlDataAdapter("select * from bookings", _
            connection)
        Dim commandbuilder1 As New SqlCommandBuilder(dataadapter1)
        dataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        bookings_data = New DataTable("Bookings")
        dataadapter1.Fill(bookings_data)

        ' Initialize a DataSet to store flights booking information
        flt_bkgs = New DataSet("Flight Bookings")
```

```
    flt_bkgs.Tables.Add(flights_data)
    flt_bkgs.Tables.Add(bookings_data)
    Dim dr As New DataRelation("Flight Bookings", _
        flt_bkgs.Tables(0).Columns(0), flt_bkgs.Tables(1).Columns(0))
    flt_bkgs.Relations.Add(dr)

    '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.IContainer

'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()

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.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    <WebMethod()> _
    Public Function getInfo() As DataSet
        Return flt_bkgs
    End Function

    <WebMethod()> _
    Public Function update(ByVal ds As DataSet) As DataSet
        ' Update the database tables
        dataadapter.Update(ds.Tables(0))
        dataadapter1.Update(ds.Tables(1))

        ' Update the flights_data DataTable
        flights_data = Nothing
        dataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        flights_data = New DataTable("Flights")
```

```
        dataadapter.Fill(flights_data)

        ' Update the bookings_data DataTable
        bookings_data = Nothing
        dataadapter1.MissingSchemaAction = MissingSchemaAction.AddWithKey
        bookings_data = New DataTable("Bookings")
        dataadapter1.Fill(bookings_data)

        ' Update the flt_bkgs DataSet
        flt_bkgs = New DataSet("Flight Bookings")
        flt_bkgs.Tables.Add(flights_data)
        flt_bkgs.Tables.Add(bookings_data)
        Dim dr As New DataRelation("Flight Bookings", _
            flt_bkgs.Tables(0).Columns(0), flt_bkgs.Tables(1).Columns(0))
        flt_bkgs.Relations.Add(dr)

        ' Return the updated DataSet
        Return flt_bkgs
    End Function
End Class
```

Visual C#

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
```

```
using System.Web.Services;
using System.Data.SqlClient;

namespace CSAirlineServices
{
   /// <summary>
   /// Summary description for FlightBooking.
   /// </summary>
   public class FlightBooking : System.Web.Services.WebService
   {
      private DataTable flights_data ;
      private SqlDataAdapter dataadapter ;
      private SqlConnection connection  ;
      private DataTable bookings_data  ;
      private SqlDataAdapter dataadapter1  ;
      private DataSet flt_bkgs  ;

      public FlightBooking()
      {
         // Initialize the SQL connection
         connection = new SqlConnection("server=localhost; " +
            "user id=sa;password=;initial catalog=Airline");

         // Create a DataTable to store flights info
         dataadapter = new SqlDataAdapter("select * from flights",
            connection);
         SqlCommandBuilder commandbuilder = new
```

```
    SqlCommandBuilder(dataadapter);
dataadapter.MissingSchemaAction =
    MissingSchemaAction.AddWithKey;
flights_data = new DataTable("Flights");
dataadapter.Fill(flights_data);

// Create a DataTable to store bookings info
dataadapter1 = new SqlDataAdapter("select * from bookings",
    connection);
SqlCommandBuilder commandbuilder1 = new
    SqlCommandBuilder(dataadapter1);
dataadapter.MissingSchemaAction =
    MissingSchemaAction.AddWithKey;
bookings_data = new DataTable("Bookings");
dataadapter1.Fill(bookings_data);

// Initialize a DataSet to store flights booking information
flt_bkgs = new DataSet("Flight Bookings");
flt_bkgs.Tables.Add(flights_data);
flt_bkgs.Tables.Add(bookings_data);
DataRelation dr = new DataRelation("Flight Bookings",
    flt_bkgs.Tables[0].Columns[0],
    flt_bkgs.Tables[1].Columns[0]);
flt_bkgs.Relations.Add(dr);

//CODEGEN: This call is required by the ASP.NET Web Services
//Designer
```

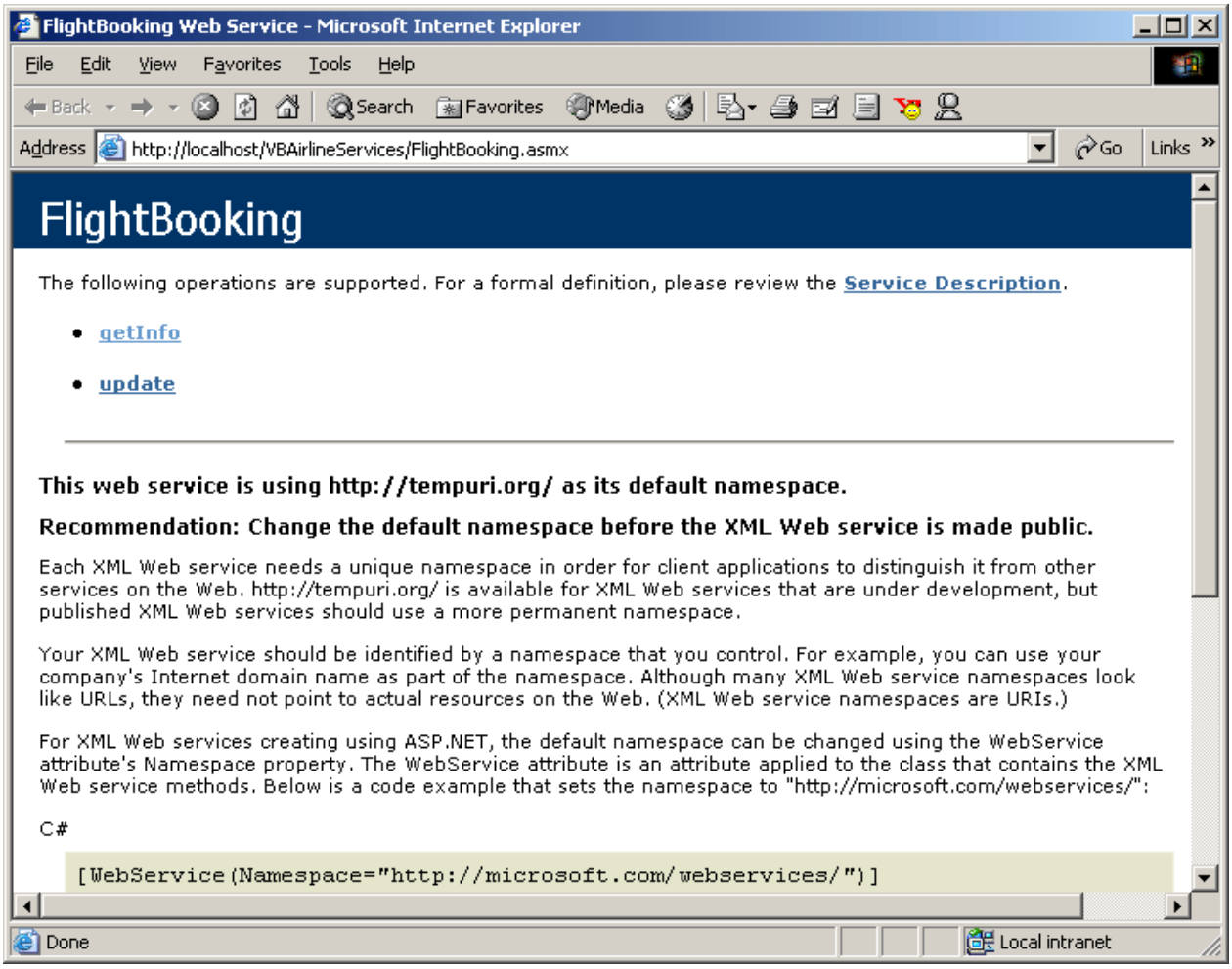
```
    InitializeComponent();
}

#region Component Designer generated code

//Required by the Web Services Designer
private IContainer components = null;

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if(disposing && components != null)
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}
```

```
#endregion

 [WebMethod()]
public DataSet getInfo()
{
    return flt_bkgs;
}

[WebMethod()]
public DataSet update(DataSet ds  )
{
    // Update the data base tables
    dataadapter.Update(ds.Tables[0]);
    dataadapter1.Update(ds.Tables[1]);

    // Update the flights_data DataTable
    flights_data = null;
    dataadapter.MissingSchemaAction =
        MissingSchemaAction.AddWithKey;
    flights_data = new DataTable("Flights");
    dataadapter.Fill(flights_data);

    // Update the bookings_data DataTable
    bookings_data = null;
    dataadapter1.MissingSchemaAction =
        MissingSchemaAction.AddWithKey;
```

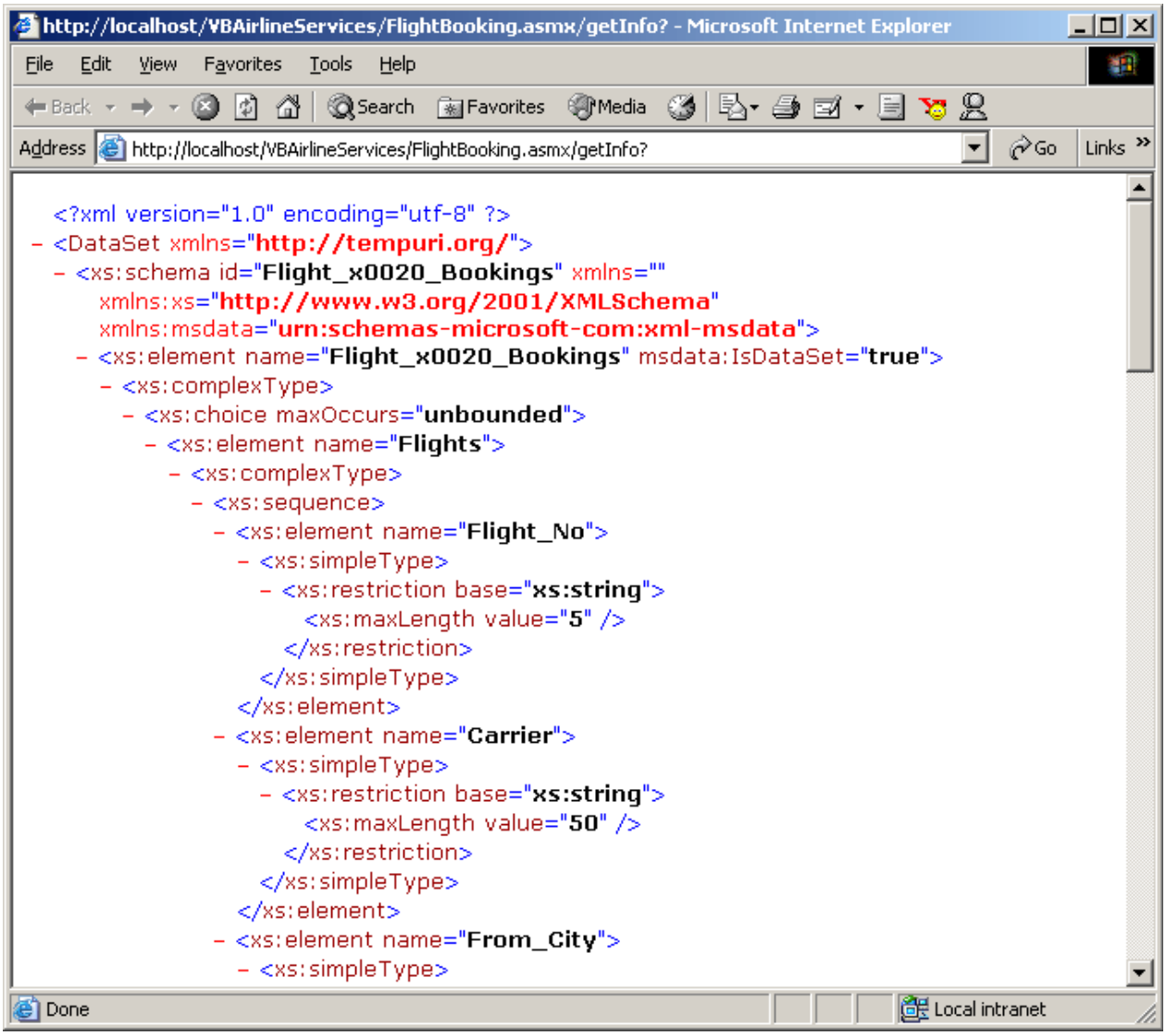
```
            bookings_data = new DataTable("Bookings");
            dataadapter1.Fill(bookings_data);

            // Update the flt_bkgs DataSet
            flt_bkgs = new DataSet("Flight Bookings");
            flt_bkgs.Tables.Add(flights_data);
            flt_bkgs.Tables.Add(bookings_data);
            DataRelation dr = new DataRelation("Flight Bookings",
               flt_bkgs.Tables[0].Columns[0],
               flt_bkgs.Tables[1].Columns[0]);
            flt_bkgs.Relations.Add(dr);

            // return the updated DataSet
            return flt_bkgs;
        }
    }
}
```

8)Chọn Build Solution từ menu Build.

9)Kích chuột phải vào tập tin FlightBooking.asmx trong Solution Explorer và chọn View Browser từ menu chuột phải. Trang mô tả của dịch vụ được hiển thị hiện bên dưới.



10) Nhấp vào liên kết getInfo. Trên các trang được hiển thị, hãy nhấp vào nút Invoke để thử nghiệm các phương thức getInfo. Nếu phương thức trả về thành công, đầu ra XML được hiển thị, thể hiện bên dưới.



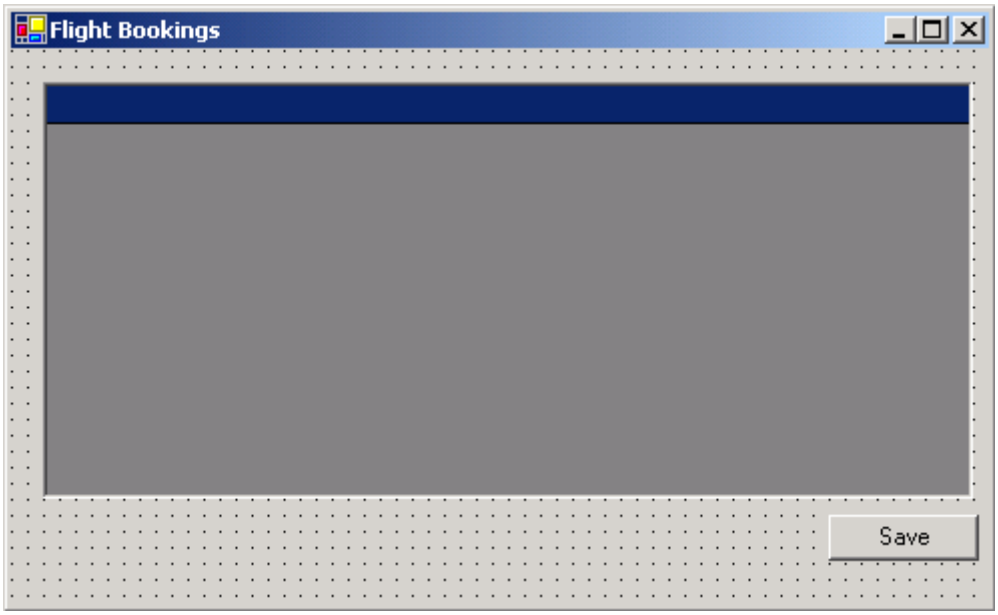
2.2.2. Tạo một ứng dụng Windows Client

Trong phần thực hành này, bạn sẽ tạo ra một ứng dụng Windows cho phép bạn truy cập các dịch vụ Web FlightBookings. Để tạo ra một ứng dụng, hãy làm theo các bước sau:

1)Từ menu File, trỏ đến Add Project, và sau đó chọn New Project để thêm một dự án mới.

2)Chọn ứng dụng Windows Application từ cửa sổ Templates và AirlineBooking loại trong trường Name và nhấn OK.

3)Thêm một điều khiển DataGrid và một Button trong Windows Forms Toolbox. Thay đổi thuộc tính Text của form thành Flight Bookings và thuộc tính Text của nút nhấn thành Save để giao diện giống như hình sau:



4)Trong AirlineBooking project, kích chuột phải vào References , và chọn Add Web Reference từ menu chuột phải để mở hộp thoại Add Web Reference. Thiết đặt là http://localhost/AirlineServices/FlightBooking.asmx trong Address box và nhấn Enter. Nhấp vào nút Add Reference để thêm một Web reference cho dự án.

5)Kích chuột phải vào tập tin Form1.vb hoặc Form1.cs trong Solution Explorer, và chọn View Code từ menu chuột phải để xem mã lệnh. Thay thế nó bằng đoạn code sau đây.

Visual Basic .NET

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Private service As localhost.FlightBooking
    Private ds As DataSet
```

```
#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    Friend WithEvents DataGrid1 As System.Windows.Forms.DataGrid
```

```
Friend WithEvents Button1 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
    Me.DataGrid1 = New System.Windows.Forms.DataGrid()
    Me.Button1 = New System.Windows.Forms.Button()
    CType(Me.DataGrid1, _
        System.ComponentModel.ISupportInitialize).BeginInit()
    Me.SuspendLayout()
    '
    'DataGrid1
    '
    Me.DataGrid1.DataMember = ""
    Me.DataGrid1.HeaderForeColor = _
        System.Drawing.SystemColors.ControlText
    Me.DataGrid1.Location = New System.Drawing.Point(16, 24)
    Me.DataGrid1.Name = "DataGrid1"
    Me.DataGrid1.Size = New System.Drawing.Size(504, 232)
    Me.DataGrid1.TabIndex = 0
    '
    'Button1
    '
    Me.Button1.Location = New System.Drawing.Point(440, 264)
    Me.Button1.Name = "Button1"
    Me.Button1.TabIndex = 1
    Me.Button1.Text = "Save"
    '
    'Form1
```

```
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(544, 309)
        Me.Controls.AddRange(New System.Windows.Forms.Control() _
            {Me.Button1, Me.DataGrid1})
        Me.Name = "Form1"
        Me.Text = "Flight Bookings"
        CType(Me.DataGrid1, _
            System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As _
        System.EventArgs) Handles MyBase.Load
        service = New localhost.FlightBooking()
        ds = service.getInfo
        DataGrid1.DataSource = ds
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As _
        System.EventArgs) Handles Button1.Click
        Try
            Dim ds1 As DataSet = service.update(ds)
            ds = ds1
```

```
            DataGrid1.DataSource = ds
            DataGrid1.DataMember = ds.Tables(0).TableName
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
End Class
```

Visual C#

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace AirlineBooking
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        internal System.Windows.Forms.Button Button1;
        internal System.Windows.Forms.DataGrid DataGrid1;
        /// <summary>
        /// Required designer variable.
        /// </summary>
```

```
private System.ComponentModel.Container components = null;
private localhost.FlightBooking service ;
private DataSet ds ;

public Form1()
{

   //
   // Required for Windows Form Designer support
   //
   InitializeComponent();
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
   if( disposing )
   {
      if (components != null)
      {
         components.Dispose();
      }
   }
   base.Dispose( disposing );
}
```

```
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Button1 = new System.Windows.Forms.Button();
    this.DataGrid1 = new System.Windows.Forms.DataGrid();
    ((System.ComponentModel.ISupportInitialize)
        (this.DataGrid1)).BeginInit();
    this.SuspendLayout();
    //
    // Button1
    //
    this.Button1.Location = new System.Drawing.Point(432, 256);
    this.Button1.Name = "Button1";
    this.Button1.TabIndex = 3;
    this.Button1.Text = "Save";
    this.Button1.Click += new
        System.EventHandler(this.Button1_Click);
    //
    // DataGrid1
    //
    this.DataGrid1.DataMember = "";
    this.DataGrid1.HeaderForeColor =
```

```
        System.Drawing.SystemColors.ControlText;
    this.DataGrid1.Location = new System.Drawing.Point(8, 16);
    this.DataGrid1.Name = "DataGrid1";
    this.DataGrid1.Size = new System.Drawing.Size(504, 232);
    this.DataGrid1.TabIndex = 2;
    //
    // Form1
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(520, 285);
    this.Controls.AddRange(new System.Windows.Forms.Control[]
        {this.Button1,
         this.DataGrid1});
    this.Name = "Form1";
    this.Text = "Flight Bookings";
    this.Load += new System.EventHandler(this.Form1_Load);
    ((System.ComponentModel.ISupportInitialize)
       (this.DataGrid1)).EndInit();
    this.ResumeLayout(false);
}
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
```

```
        {
            Application.Run(new Form1());
        }

        private void Button1_Click(object sender, System.EventArgs e)
        {
            try
            {
                DataSet ds1 = service.update(ds);
                ds = ds1;
                DataGrid1.DataSource = ds;
                DataGrid1.DataMember = ds.Tables[0].TableName;
            }
            catch (Exception ex )
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void Form1_Load(object sender, System.EventArgs e)
        {
            service = new localhost.FlightBooking();
            ds = service.getInfo();
            DataGrid1.DataSource = ds;
        }
    }
}
```

6) Chọn Build Solution từ menu Build.

7)Chạy các ứng dụng Windows. Kiểm tra các ứng dụng bằng cách tạo, xóa, và sửa đổi hồ sơ.

Lưu ý

Các trường Flight_No và Flight_Date của table là khóa chính và khóa ngoại. kiểm tra table Flights đảm bảo rằng <Công suất = Seats_Booked.

Các trường Flight_No, Seat_No, và trường Flight_Date là các khóa của table Bookings. Các trường Flight_No và Flight_Date của table Bookings tham chiếu đến các trường của table Flights.

III. Tóm tắt trình tự thực hiện hoặc quy trình công nghệ

STT	*Tên các bước công việc*	*Dụng cụ, thiết bị, vật tư*	*Yêu cầu kỹ thuật*	*Các chú ý về an toàn lao động*
1	**Tạo một XML Web Service** - Mở Visual Studio.Net - Tạo mới một Project -Chọn ASP.NET Web Service -Add Web Service -FlightBooking.asmx - Viết code - Build solution - Chạy ứng dụng	Máy tính cài Visual studio.net	Thao tác đúng trình tự thực hiện, viêt code đảm bảo đúng cú pháp không lỗi, Tham chiếu đến các namespace cho các đối tượng	
2	**Tạo ứng dụng trên Client giao tiếp với dịch vụ XML web service trên Server** - Mở visual studio và tạo mới một ứng dụng - Đặt tên cho ứng dụng - Thêm các điều khiển DataGrid và Botton - Tham chiếu đến dịch vụ web từ server - Viết code cho ứng dụng - Build ứng dụng - Chạy ứng dụng	Máy tính cài đặt Visual Studio.NET	Thao tác đúng trình tự thực hiện, viêt code đảm bảo đúng cú pháp không lỗi, Tham chiếu đến các namespace cho các đối tượng	

Bài 3:Tạo và sử dụng đối tượng .Net Remoting

Ứng dụng phân tán giúp để thiết lập thông tin liên lạc giữa các đối tượng đang chạy trong quá trình khác nhau nằm trên cùng một máy tính hoặc đặt trên các máy tính tại các địa điểm địa lý khác nhau. NET Framework. Cung cấp một tập hợp các lớp và phương thức cho phép bạn thiết lập một kết nối và giao tiếp giữa các đối tượng một cách dễ dàng.

Với cơ sở hạ tầng lập trình trước đó, tạo ra một ứng dụng phân tán đòi hỏi một kiến thức chuyên sâu về các giao thức truyền tải được sử dụng trong quá trình giao tiếp. Sử dụng các NET Framework., Bạn có thể thiết lập thông tin liên lạc giữa các đối tượng mà không cần phải biết về các giao thức hoặc về các cơ chế mã hóa và giải mã tham gia vào sự phát triển của một ứng dụng được cung cấp. Trong bài này, bạn sẽ học cách tạo ra, cấu hình, và an toàn đối tượng NET Remoting bằng cách sử dụng các lớp mà các NET Framework cung cấp.

I. Mục tiêu

Học xong bài này người học đạt được các kỹ năng sau:

- Kỹ năng lập trình với các ứng dụng .Net Remoting
- Kỹ năng triển khai các ứng dụng .Net Remoting
- Kỹ năng cấu hình dịch vụ ứng dụng .Net Remoting

II. Nội dung

3.1. Lý thuyết liên quan

3.1.1. Tìm hiểu về .Net Remoting

Tổng quan về NET Remoting.

Hãy xem xét ví dụ sau: Bạn đã tạo ra một ứng dụng máy tính bỏ túi (pocket PC) bằng cách sử dụng Visual C # chạy trên Microsoft Windows CE để truy cập vào doanh số bán hàng mới nhất dữ liệu từ máy tính Microsoft Windows Server 2000 đó không phải là một phần của cùng một mạng. Một thành phần máy chủ xuất ra dữ liệu này cho các ứng dụng máy tính bỏ túi được tạo ra bằng cách sử dụng Microsoft Visual Basic 6.0. Thành phần máy chủ này lấy dữ liệu có liên quan từ một cơ sở dữ liệu Microsoft SQL Server 2000. Đây là một hình thức rất phổ biến mà ngày nay có thể truy cập thông tin bất cứ lúc nào, bất cứ nơi nào, trên bất kỳ thiết bị nào. Trong hình thức này, các ứng dụng có thể dễ dàng giao tiếp với nhau bằng cách sử dụng các lớp NET Remoting được cung cấp bởi các NET Framework.

Hệ thống NET Remoting cung cấp một số dịch vụ để kích hoạt các đối tượng, kiểm soát thời gian sống của từng đối tượng, và các thông điệp vận chuyển đến và đi từ các đối tượng từ xa bằng cách sử dụng các kênh truyền thông. Các kênh truyền thông là những đối tượng vận chuyển tin nhắn giữa các đối tượng từ xa. Các kênh truyền thông được nói lại chi tiết trong phần 3 của bài này. Bất kỳ thông điệp được gửi cùng một kênh truyền thông được mã hóa và giải mã bằng cách sử dụng định dạng serialization formatters, chẳng hạn như các mã hóa nhị phân của SOAP object,được định dạng tuần tự hóa các đối tượng có thể giúp bạn mã hóa và giải mã các tin nhắn được gửi hoặc nhận được từ một đối tượng từ xa. Hai loại mã hóa có thể cho tất cả các tin nhắn: nhị phân và mã hóa XML. ứng dụng trong đó hiệu suất là rất quan trọng sử dụng mã hóa nhị phân. Trong trường hợp khả năng tương tác với các hệ thống điều khiển từ xa khác là điều cần thiết, mã hóa XML là sự lựa chọn thích hợp.

Hệ thống NET Remoting cho phép bạn thực hiện các thông tin liên lạc giữa các đối tượng khác nhau trong các lĩnh vực ứng dụng hoặc các quá trình

khác nhau bằng cách sử dụng giao thức vận chuyển khác nhau, chẳng hạn như HTTP và TCP / IP, định dạng tuần tự, chẳng hạn như nhị phân hoặc SOAP.

Truy cập các đối tượng Across Remoting Boundaries

Hệ thống NET Remoting cho phép Client gọi các phương thức trên các đối tượng qua các phạm vi của remoting. Phạm vi của Remoting gồm các miền ứng dụng, tiến trình (processes), và các máy tính. Một miền ứng dụng, như đã thảo luận ở trong phần, "Tìm hiểu về NET Framework.", là ranh giới mà trong đó một ứng dụng chạy. Nhiều hơn một ứng dụng tên miền có thể chạy trong một quá trình duy nhất. Giao tiếp giữa các đối tượng vượt phạm vi truy cập từ xa Net Remoting yêu cầu:

-Một đối tượng máy chủ cho nhiều chức năng có thể gọi tới từ bên ngoài ranh giới của nó

-Một Client tạo ra các lời gọi đến các đối tượng máy chủ (server object)

-Một cơ chế vận chuyển để vượt qua các cuộc gọi từ một đầu vào khác

Để truy cập vào các đối tượng và gọi các phương thức trên, bạn cần con trỏ hoặc tham chiếu đến các đối tượng. Tuy nhiên, truy cập các đối tượng qua các phạm vi truy cập từ xa là khó khăn vì phụ thuộc vaò địa chỉ. Địa chỉ trong một quá trình không có bất kỳ ý nghĩa gì trong các tiến trình khác nhau. Để xử lý vấn đề này, bạn có thể sử dụng một bản sao của đối tượng máy chủ trong các ứng dụng của máy khách(client). Các máy khách có thể gọi một phương thức trên các bản sao trên máy cục bộ (local) của các đối tượng máy chủ.

Khi bạn sao chép các đối tượng trong tiến trình máy khách, hãy chắc chắn rằng các đối tượng không chứa một số lượng lớn các phương thức, và các đối

tượng không phải là các thành phần có kích thước lớn. Đối tượng rất lớn với nhiều phương pháp có thể là sự lựa chọn cho người nghèo sao chép hoặc đi ngang qua giá trị các quá trình khác bởi vì khách hàng yêu cầu chỉ có giá trị trả về bởi một hoặc một vài phương pháp của đối tượng máy chủ. Sao chép một đối tượng máy chủ toàn bộ quá trình khách hàng là một sự lãng phí tài nguyên của khách hàng, trong đó bao gồm băng thông, bộ nhớ và thời gian xử lý. Ngoài ra, nhiều đối tượng máy chủ lộ chức năng công cộng, nhưng yêu cầu dữ liệu cá nhân trên máy chủ để thực hiện nội bộ.

Cảnh bảo

Khi bạn sao chép đối tượng máy chủ, bạn có thể cho phép mã độc hại để kiểm tra dữ liệu nội bộ, tạo ra tiềm năng cho các vấn đề an ninh

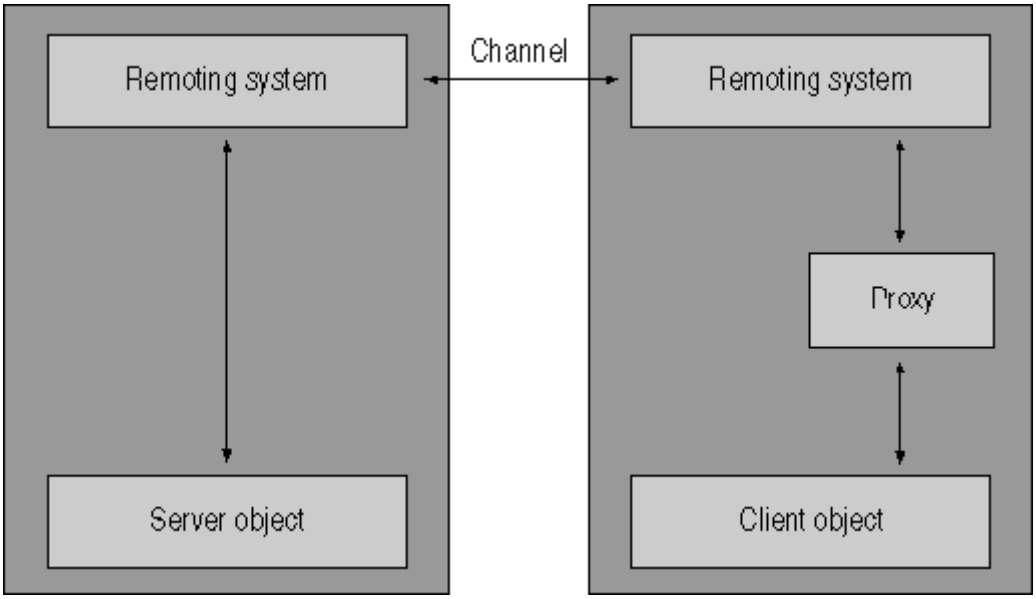
Thật khong tốt khi sao chép các đối tượng, chẳng hạn như là một đối tượng FileInfo, quá trình khách hàng bởi vì một đối tượng FileInfo chứa một tham chiếu đến một tập tin hệ điều hành, trong đó có một địa chỉ duy nhất trong bộ nhớ của quá trình máy chủ. Bạn có thể sao chép địa chỉ này cùng với các đối tượng, nhưng địa chỉ là quá trình cụ thể và sẽ không làm việc trong quá trình khác. Trong tình huống này, bạn vượt qua các đối tượng máy chủ như là một tài liệu tham khảo cho các ứng dụng của khách hàng. Các ứng dụng của máy khách sử dụng tài liệu tham khảo để gọi các phương thức trên đối tượng máy chủ. Hệ thống truy cập từ xa xử lý các lời gọi phương thức và gửi nó đến đối tượng máy chủ. Sau khi phương thức này được thực thi, kết quả được trả về cho các ứng dụng của máy khách.

Kiến trúc .Net Remoting

Để giao tiếp giữa các đối tượng máy chủ và máy khách trong NET Remoting, bạn cần phải sử dụng các tham chiếu đối tượng của đối tượng máy chủ trong các ứng dụng của máy khách. Khi bạn tạo một thể hiện của các đối tượng từ xa bằng cách sử dụng các từ khoá mới, máy khách của bạn nhận được một tham chiếu đến đối tượng máy chủ. Sau khi có được các dữ liệu tham khảo đối tượng, máy khách có thể gọi các phương thức trên đối tượng máy chủ như là đối tượng cư trú trong tiến trình Process của máy khách và không chạy trên một máy tính riêng biệt.

NET Remoting sử dụng các đối tượng proxy để cho phép việc sử dụng các đối tượng máy chủ trong tiến trình máy khách. Khi bạn tạo một thể hiện của các loại đối tượng từ xa trong các ứng dụng của máy khách hàng, NET Remoting tạo ra một đối tượng proxy và gửi nó cho các ứng dụng của máy khách. Đối tượng proxy này có chứa dữ liệu tham chiếu cho tất cả các phương thức và các thuộc tính của đối tượng máy chủ. Khi bạn gọi một phương thức đó là trên đối tượng proxy đó, hệ thống truy cập từ xa nhận được cuộc gọi từ các luồng đến tiến trình của máy chủ và gọi đối tượng của máy chủ, và trả về giá trị trả lại các proxy client, trả về kết quả cho các ứng dụng của máy khách hàng.

Hình 4.1 cho thấy làm thế nào một máy khách giao tiếp với một đối tượng từ xa.



Hình 4-1. Communication between a client and a remote object

Để tạo ra một hệ thống remoting để thực hiện các hoạt động trước đó, bạn cần phải học lập trình mạng và biết về một mảng rộng các giao thức, chẳng hạn như giao thức TCP / IP và HTTP, và về chi tiết kỹ thuật định dạng serialization. Tuy nhiên, trong NET Remoting., Một sự kết hợp của các công nghệ thực hiện các nhiệm vụ cấp thấp, chẳng hạn như mở một kết nối mạng, định dạng thông điệp, viết tin nhắn vào các luồng streams, và gửi byte dữ liệu tới ứng dụng tiếp nhận nó. Sự kết hợp của công nghệ được biết đến như là kênh giao thông truyền tải dữ liệu (transport channel).

Để hiểu rõ vai trò của các kênh NET Remoting, hãy xem xét ví dụ sau: Bạn có một ứng dụng đang chạy trên một máy tính, và bạn muốn sử dụng chức năng tiếp xúc của một loại đó là có sẵn trên máy tính khác. Nếu bạn cấu hình đúng cho cả mày khách và các đối tượng máy chủ, các ứng dụng của máy khách chỉ có thể tạo ra một thể hiện mới của lớp máy chủ.

Hệ thống truy cập từ xa tạo ra một đối tượng proxy đại diện cho lớp trong máy chủ và trả về một tham chiếu của đối tượng proxy cho đối tượng máy khách. Khi một máy khách gọi bất kỳ phương thức trên các đối tượng từ xa, hệ thống truy cập từ xa xử lý các lời gọi phương thức, kiểm tra các loại thông tin, và gửi các lời gọi qua các kênh đến tiến trình máy chủ. Một kênh nghe nhận được yêu cầu từ các ứng dụng của máy khách và chuyển cho hệ thống máy chủ truy cập từ xa. Hệ thống điều khiển từ xa máy chủ tạo ra hoặc nằm đối tượng được yêu cầu và gọi nó. Sau khi cuộc gọi được xử lý, hệ thống điều khiển từ xa máy chủ tạo ra một thông điệp trả lời, và các kênh của máy chủ gửi nó đến các kênh máy khách. Hệ thống điều khiển từ xa của khách hàng sau đó trả về kết quả của các cuộc gọi đến các đối tượng máy khách thông qua các đối tượng proxy. Vai trò của kênh được giải thích chi tiết trong bài sau.

Đối tượng Remotable và Nonremotable

Trong ứng dụng phân tán, có hai loại của các đối tượng: remotable và nonremotable. Đối tượng Nonremotable tôi không cung cấp các hệ thống remoting với một phương pháp hoặc sao chép hoặc sử dụng chúng trong một miền ứng dụng khác. Vì vậy, bạn có thể truy cập các đối tượng này chỉ trong phạm vi ứng dụng của riêng mình. Các đối tượng Remotable hoặc có thể được truy cập bên ngoài miền ứng dụng của họ hoặc ngữ cảnh bằng cách sử dụng một proxy hoặc sao chép và thông qua bên ngoài miền ứng dụng hoặc ngữ cảnh của họ. Điều này ngụ ý rằng một số đối tượng remotable được thông qua tham khảo và một số được thông qua giá trị. Ví dụ, nếu bạn có một đối tượng lớn với một số phương thức, lựa chọn tốt nhất của bạn là làm cho đối tượng nonremotable. Sau đó, tạo một đối tượng remotable nhỏ kích thước có thể được công bố, sao chép đến các ứng dụng của máy khách. Sử dụng các đối tượng remotable chỉ đạo các cuộc gọi đến đối tượng nonremotable lớn hơn.

Có hai loại của các đối tượng remotable:

- Marshal-by-value objects.

Những đối tượng được sao chép và thông qua giá trị ra khỏi miền ứng dụng.

- Marshal-by-reference objects

Các máy khách sử dụng các đối tượng này cần một proxy để truy cập các đối tượng từ xa.

Marshal-by-Value Objects

Đối tượng Marshal-by-value cần thực hiện giao diện ISerializable hoặc cần được đánh dấu với thuộc tính SerializableAttribute để hệ thống truy cập từ xa có thể serialize các đối tượng tự động. Khi máy khách gọi một phương thức trên đối tượng Marshal-by-value, hệ thống truy cập từ xa tạo ra một bản sao của các đối tượng này và chuyển các bản sao cho các miền ứng dụng của máy khách. Sau khi ứng dụng máy khách nhận được bản sao, bản sao trong lĩnh vực ứng dụng máy khách xử lý bất kỳ lời gọi tới phương thức. Ngoài ra, khi các đối tượng marshal-by-value được thông qua như các đối số, một bản sao của đối tượng được truyền cho phương thức.

Để cải thiện hiệu suất và giảm thời gian xử lý, di chuyển trạng thái đầy đủ của đối tượng và chức năng của nó cho miền ứng dụng mục tiêu. Sử dụng các đối tượng marshal-by-value làm giảm thời gian và tốn kém các chuyến đi trong đường biên miền mạng, quá trình, và ứng dụng. Bạn cũng sử dụng các đối tượng marshal-by-value trực tiếp từ trong miền ứng dụng ban đầu của đối tượng.

Marshal-by-Reference Objects

Đối tượng Marshal-by-Reference Objects là những đối tượng remotable mở rộng các lớp System.Marshal-ByRefObject. Khi một máy khách tạo ra một thể hiện của một đối tượng Marshal-by-Reference Objects trong phạm vi ứng dụng riêng của mình, cơ sở hạ tầng NET Remoting tạo ra một đối tượng proxy trong miền ứng dụng gọi đại diện cho các đối tượng Marshal-by-Reference Objects và trả về một tài liệu tham khảo đó proxy để người gọi. Khách hàng sau đó làm cho các cuộc gọi phương thức để các đối tượng proxy. Hệ thống truy cập từ xa Marshal những cuộc gọi, trả chúng về cho miền ứng dụng máy chủ, và gọi các cuộc gọi trên đối tượng thực tế.

Trước khi bạn tạo ra một ứng dụng phân tán, bạn nên biết những điều sau đây:

Bạn có thể gọi một đối tượng được tạo ra trong một miền cụ thể trực tiếp từ tên miền đó.

Nếu bạn cần phải gọi một đối tượng từ một tiến trình bên ngoài miền ứng dụng riêng của mình, bạn nên công bố các đối tượng. Khi bạn xuất bản một đối tượng, bạn có sao chép các đối tượng đến một miền ứng dụng khác hoặc tạo một proxy của các đối tượng trong một miền ứng dụng khác.

Nếu kích thước của đối tượng là lớn hoặc nếu nó có chứa một số lượng lớn các phương thức, đối tượng không thể có hiệu quả công bố, tiêu thụ trong đường biên miền. Do đó, bạn phải quyết định loại đối tượng bạn muốn xuất bản dựa trên các yêu cầu của các ứng dụng của bạn.

Trong bài học này, bạn đã học về các vấn đề cơ bản của. NET Remoting, tìm hiểu qua các đối tượng và các tiến trình, và các kiến trúc NET Remoting. Ngoài ra, bạn đã học về đối tượng remotable và nonremotable. Bạn cũng đã học về các

đối tượng Marshal-by-Reference Objects và các đối tượng Marshal-by-Value Objects.

Trong bài học tiếp theo, bạn sẽ học cách tạo ra và kích hoạt các đối tượng. Ngoài ra, bạn sẽ tìm hiểu về quá trình tồn tại của các đối tượng. Bạn cũng sẽ tìm hiểu để kiểm soát các đối tượng.

3.1.2. Làm việc với các đối tượng Server-Activated and Client-Activated

Trong bài trước, bạn đã học về Remoting NET và kiến trúc của nó. Như đã đề cập trước đó, NET Remoting có máy chủ và các tiến trình trên máy khách. Để giao tiếp với tiến trình trên máy chủ, bạn sử dụng các tham chiếu của các đối tượng máy chủ trong các ứng dụng của máy khách. Trong bài học này, bạn sẽ tìm hiểu làm thế nào để tạo ra các máy chủ và các đối tượng máy khách. Bạn cũng sẽ tìm hiểu làm thế nào để kích hoạt các đối tượng và kiểm soát việc cho thuê của các đối tượng này.

Tìm hiểu về Remote Object Activation

Khi bạn phát triển một đối tượng, bạn không cần phải theo dõi việc tạo ra của đối tượng. Bạn chỉ cần đảm bảo rằng các đối tượng đáp ứng các cuộc gọi phương thức. Tuy nhiên, khi bạn phát triển một đối tượng từ xa, bạn cần phải theo dõi việc tạo ra và khởi tạo của đối tượng vì cách cư xử một đối tượng từ xa phụ thuộc vào đối tượng được tạo ra và kích hoạt như thế nào. Bạn phải nhận thức của các đối tượng từ xa được kích hoạt. Bạn cần xác định cho hệ thống truy cập từ xa loại kích hoạt là cần thiết trên các đối tượng trước khi hệ thống remoting cung cấp cho các đối tượng máy khách.

Trong hai phần tiếp theo, bạn sẽ tìm hiểu về hai loại chế độ kích hoạt trong hệ thống NET Remoting: máy chủ kích hoạt và kích hoạt máy khách.

Kích hoạt máy chủ

Trong kích hoạt máy chủ, các đối tượng được tạo ra trên máy chủ khi bạn gọi một phương thức trong lớp máy chủ. Tuy nhiên, các đối tượng không được tạo ra khi bạn sử dụng các từ khoá mới để tạo ra một thể hiện của lớp máy chủ.

Hãy xem xét ví dụ sau: văn phòng của các tổ chức ”An Sinh Xã Hội” có một thành phần dịch vụ cho phép các tổ chức, chẳng hạn như ngân hàng và văn phòng căn hộ, để kiểm tra tính hợp lệ của một mã số bảo hiểu xã hội (social security number - SSN). Máy khách kết nối với dịch vụ này và cung cấp số an sinh xã hội của một người để xác nhận. Dịch vụ xác nhận số lượng và trả về thông tin cá nhân thích hợp. Trong trường hợp này, máy khách luôn luôn kết nối đến máy chủ tại văn phòng An Sinh Xã Hội. Dịch vụ được kích hoạt chỉ khi một phương thức gọi đến từ các máy khách yêu cầu xác nhận của một số SSN. Điều này được biết đến như kích hoạt máy chủ. Các mã sau đây sẽ hiển thị kích hoạt máy chủ của các đối tượng.

Visual Basic .NET

```
Imports System.Runtime.Remoting
Public Class SSNServer
   Inherits MarshalByRefObject
   Public Function ValidateSSN(Byval number as Long) As String
      ' Return the address
      Dim address As String
      ' Do some work here to validate the SSN
      Return address
   End Function
```

```
End Class
```

Visual C#

```
using System;
using System.Runtime.Remoting;
namespace SSNComponentCSharp
{
   public class SSNServer : MarshalByRefObject
   {
      public SSNServer()
      {
      }
      public String ValidateSSN(long number)
      {
         // Return the address
         String address;
         // Do some work here to validate the SSN
         return address;
      }
   }
}
```

Mã trước trên hiển thị một mẫu đối tượng trên máy chủ mà mà khi chạy phải truyền vào cho nó một tham số kiểu ***long*** và trả về địa chỉ tương ứng. Các

mã sau đây minh họa làm thế nào để gọi phương thức ValidateSSN. Bởi vì đây là trong chế độ kích hoạt máy chủ, đối tượng được tạo ra chỉ khi bạn gọi phương thức ValidateSSN, không phải khi bạn tạo ra một thể hiện của lớp.

Visual Basic .NET

```
Dim serverInstance as New SSNServer()
' Remote Object on the server is created in the next line
Console.WriteLine(serverInstance.ValidateSSN(242990307))
```

Visual C#

```
SSNServer serverInstance = new SSNServer();
// Remote Object on the server is created in the next line
Console.WriteLine(serverInstance.ValidateSSN(242990307));
```

Bạn có thể tạo một đối tượng kích hoạt máy chủ như là một đối tượng Singleton hoặc SingleCall dựa trên các yêu cầu của ứng dụng từ xa của bạn. Singleton đối tượng có thể có chỉ có một dụ bất kể số lượng máy khách mà họ có. Những đối tượng này cũng có một thời gian sống mặc định. Vì vậy, nếu bạn tạo một đối tượng máy chủ như là một đối tượng Singleton, một trường hợp duy nhất của đối tượng máy chủ quản lý tất cả các máy khách. Khi bạn khai báo một đối tượng như là một đối tượng SingleCall, hệ thống remoting tạo ra một đối tượng mỗi lần một phương thức máy khách gọi một đối tượng từ xa. Để đăng ký một đối tượng máy chủ như là một đối tượng Singleton, bạn chỉ định các loại của các đối tượng như WellKnownObjectMode.Singleton. Các mã sau đây cho thấy làm thế nào để xác định một đối tượng như WellKnownObjectMode.Singleton.

Visual Basic .NET

```
Imports System.Runtime.Remoting
Public Class SSNServer
   Inherits MarshalByRefObject
   Sub New()
      RemotingConfiguration.ApplicationName = "testService"
      RemotingConfiguration.RegisterWellKnownServiceType( _
         GetType(testService), "MyUri", _
         WellKnownObjectMode.Singleton)
      Console.WriteLine("Press Enter to Stop")
      Console.ReadLine()
   End Sub
End Class
Class testService
   'Service Component
End Class
```

Visual C#

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
```

```
namespace SSNComponentCSharp
{
   public class SSNServer
   {
      public SSNServer()
      {
         RemotingConfiguration.ApplicationName = "testService";
         RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(testService),
            "MyUri",WellKnownObjectMode.Singleton);
         Console.WriteLine("Press enter to stop.");
         Console.ReadLine();
      }
   }
}

class testService : MarshalByRefObject
{
   // Service Object Registered in the SSNServer class above.
}
```

Lưu ý

Bạn cần phải thêm một tham chiếu đến System.Runtime.Remoting để sử dụng các lớp trong không gian tên System.Runtime.Remoting.Channels.Tcp.

Để đăng ký một đối tượng máy chủ như là một đối tượng SingleCall, bạn chỉ định các loại của các đối tượng như WellKnownObjectMode.SingleCall. Các mã sau đây cho thấy làm thế nào để xác định loại của một đối tượng như WellKnownObjectMode.SingleCall.

Visual Basic .NET

```
Imports System.Runtime.Remoting
Public Class SSNServer
   Inherits MarshalByRefObject
   Sub New()
      RemotingConfiguration.ApplicationName = "testService"
      RemotingConfiguration.RegisterWellKnownServiceType( _
         GetType(testService), "MyUri", _
         WellKnownObjectMode.SingleCall)
      Console.WriteLine("Press Enter to Stop")
      Console.ReadLine()
   End Sub
End Class
```

```
Class testService
    'Service Component
End Class
```

Visual C#

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;


namespace SSNComponentCSharp
{
    public class SSNServer
    {
        public SSNServer()
        {
            RemotingConfiguration.ApplicationName = "testService";
            RemotingConfiguration.RegisterWellKnownServiceType(
                typeof(testService),
                "MyUri",WellKnownObjectMode.SingleCall);
            Console.WriteLine("Press enter to stop.");
            Console.ReadLine();
```

```
        }
    }
}


class testService : MarshalByRefObject
{
    // Service Object Registered in the SSNServer class above.
}
```

Kích hoạt Client

Khách hàng kích hoạt đối tượng được tạo ra trên máy chủ khi bạn tạo một thể hiện bằng cách sử dụng các từ khoá mới. Ví dụ, bạn có thể tạo ra một ứng dụng chat cho phép người sử dụng để giao tiếp với những người dùng khác trên một mạng nội bộ. Khi người dùng đăng nhập vào, một thể hiện mới của đối tượng máy chủ trò chuyện được tạo ra. Constructor mặc định của đối tượng máy chủ trò chuyện chứa đoạn mã để xác định vị trí người dùng đang trực tuyến và hiển thị nó cho người sử dụng đăng nhập. Trong trường hợp này, các đối tượng từ xa trên máy chủ sẽ được tạo ra khi bạn tạo ra các ví dụ bằng cách sử dụng các từ khoá mới .

Miền ứng dụng của máy khách định nghĩa đời sống của các đối tượng máy khách kích hoạt. Những đối tượng có mặt trong miền ứng dụng của máy khách. Trong các máy khách kích hoạt, khi máy khách cố gắng để tạo ra một thể hiện của đối tượng máy chủ, kết nối được đến máy chủ, và các proxy client được tạo ra. Các đối tượng proxy được tạo ra bằng cách sử dụng một tham chiếu đối tượng, mà là thu được sau khi các đối tượng từ xa được tạo ra trên máy chủ. Khi

một máy khách tạo ra một trường hợp khách hàng kích hoạt, các ví dụ của các đối tượng từ xa phục vụ một máy khách cụ thể cho đến khi hết hạn hợp đồng thuê và thu gom rác tái chế bộ nhớ của nó. Thuê kiểm soát thời gian một đối tượng từ xa vẫn còn trong bộ nhớ. Nếu một máy khách tạo ra hai trường hợp mới của một đối tượng từ xa, mỗi tham chiếu đến các đối tượng từ xa gọi chỉ có các ví dụ cụ thể trong các ứng dụng máy chủ.

Trong COM, máy khách giữ một đối tượng trong bộ nhớ bằng cách duy trì một tham chiếu đến nó. Khi máy khách COM cuối cùng phát hành tham chiếu cuối cùng của mình cho một đối tượng, đối tượng xóa chính nó.Máy khách kích hoạt cho phép bạn kiểm soát thời gian sống của đối tượng máy chủ mà không cần duy trì các tài liệu tham khảo hoặc liên tục ping để xác nhận sự tồn tại của máy chủ hoặc máykhách.

Đối tượng máy khách kích hoạt sử dụng hợp đồng thuê cả cuộc đời để xác định thời gian tồn tại của mình. Khi một máy khách tạo ra một đối tượng, nó xác định một khoảng thời gian mặc định cho các đối tượng nên tồn tại. Nếu đối tượng từ xa đạt đến kết thúc của thời gian đời mặc định của nó, các địa chỉ liên lạc đối tượng máy khách và yêu cầu liệu có nên tiếp tục tồn tại và trong bao lâu. Nếu máy khách không có sẵn, một giới hạn thời gian mặc định được quy định cụ thể đối tượng máy chủ chờ đợi trong khi cố gắng để liên lạc với máy khách trước khi đánh dấu chính nó có sẵn để thu gom rác thải. Các máy khách thậm chí có thể yêu cầu một cuộc đời vô hạn, mặc định ngăn chặn các đối tượng từ xa được tái chế cho đến khi tên miền ứng dụng của máy chủ hư hỏng.

Nhiệm vụ Remoting

Sau khi tìm hiểu về đối tượng server-active và client-active, bạn có thể chuyển sự chú ý của bạn remoting cơ bản các nhiệm vụ là rất cần thiết để thiết lập truyền thông giữa một máy khách và một đối tượng từ xa. Bạn thực hiện các nhiệm vụ truy cập từ xa sau đây để xuất bản bất kỳ dịch vụ bên ngoài lĩnh vực dịch vụ:

-Xác định các miền ứng dụng này sẽ lưu trữ các dịch vụ

-Xác định các chế độ kích hoạt: Server-active hoặc Client-active

-Xác định và tạo ra một kênh và một cổng

-Xác định cách ứng dụng máy khách có được các thông tin siêu dữ liệu về dịch vụ

Miền ứng dụng của host có thể là một dịch vụ của Windows, giao diện điều khiển ứng dụng, ứng dụng Windows Forms, hoặc một dịch vụ Web XML sử dụng ASP.NET. Sau khi chọn một miền ứng dụng cho dịch vụ của bạn, bạn xác định các chế độ kích hoạt cho dịch vụ này.

Một khi bạn đã xác định chế độ kích hoạt và các thông tin khác có liên quan như tên ứng dụng và các thiết bị đầu cuối cho hệ thống truy cập từ xa, sau đó bạn cấu hình hệ thống của bạn bằng cách sử dụng các tập tin cấu hình. Để cấu hình một hệ thống từ xa, hãy gọi các phương thức Configure của lớp RemotingConfiguration. Các mã sau đây cho thấy làm thế nào để gọi phương thức Cấu hình của lớp RemotingConfiguration.

Visual Basic .NET

```
RemotingConfiguration.Configure("configuration.config")
```

Visual C#

```
RemotingConfiguration.Configure("configuration.config");
```

Trong đoạn mã trên, các tập tin Configuration.config có chứa thông tin về cấu hình hệ thống truy cập từ xa. Bạn cũng có thể cấu hình hệ thống remoting lập trình. Khi bạn cấu hình hệ thống remoting lập trình, bạn không yêu cầu các tập tin cấu hình.

Tiếp theo, tạo kênh thích hợp, TcpChannel hoặc HttpChannel, và đăng ký nó bằng cách sử dụng phương pháp RegisterChannel của lớp RemotingConfiguration. Nếu bạn sử dụng file cấu hình để cấu hình hệ thống remoting, bạn có thể tạo ra các kênh thích hợp chỉ đơn giản bằng cách tải các tập tin cấu hình bằng cách sử dụng phương thức RemotingConfiguration.Configure.

Cuối cùng, công bố dịch vụ để nó có thể truy cập từ bên ngoài miền. Điều này cho phép các ứng dụng máy chủ để hoạt động.

Sử dụng Lifetime Leases

Thời gian sống của một đối tượng marshal-by-tham chiếu là thời gian mà đối tượng nằm trong bộ nhớ. Một đối tượng marshal-by-tham chiếu vẫn tồn tại trong bộ nhớ mãi mãi bất kể các loại đối tượng. Nói cách khác, tất cả các đối tượng marshal-by-tham khảo kích hoạt máy chủ và Client-active có một cuộc đời của riêng mình. Những đối tượng được phát hành từ bộ nhớ sau khi cuộc đời hết hạn và đối tượng được đánh dấu để thu gom rác thải. GC sau đó loại bỏ các đối tượng từ bộ nhớ.

NET Remoting hệ thống xóa một đối tượng chỉ khi nó được đánh dấu là đã sẵn sàng để thu gom rác thải. Người quản lý cho thuê đời của miền ứng dụng máy chủ chịu trách nhiệm xác định các đối tượng đã sẵn sàng để thu gom rác

thải. Tuy nhiên, một đối tượng tài trợ có thể yêu cầu một hợp đồng thuê mới cho một đối tượng cụ thể bằng cách đăng ký với người quản lý cho thuê.

Bất cứ khi nào một đối tượng marshal-by-tham khảo được truy cập từ xa bên ngoài một miền ứng dụng, một hợp đồng thuê suốt đời được tạo ra cho đối tượng đó. Mỗi miền ứng dụng có chứa một người quản lý cho thuê, quản lý hợp đồng cho thuê trong phạm vi của nó. Người quản lý cho thuê định kỳ xem xét lại các hợp đồng cho thuê hết hạn. Nếu hợp đồng thuê hết hạn, người quản lý cho thuê đi qua danh sách các nhà tài trợ cho các đối tượng đó và hỏi nếu bất kỳ người trong số họ muốn gia hạn hợp đồng. Nếu không ai trong số các nhà tài trợ gia hạn hợp đồng, người quản lý cho thuê loại bỏ hợp đồng thuê. Sau đó, đối tượng được xóa, và thu gom rác thải đòi bộ nhớ đối tượng. Do đó, thời gian tồn tại của một đối tượng có thể được lâu hơn so với thuê cuộc đời của mình.

Khởi tạo Lifetime Leases

Để khởi tạo một lifetime lease, bạn ghi đè lên chức năng InitializeLifetimeService của lớp MarshalByRefObject. Sau đây là cú pháp được sử dụng để ghi đè lên chức năng InitializeLifetimeService.

```
Visual Basic .NET
Imports System.Runtime.Remoting.Lifetime
Public Class MyLifetimeControlObject
   Inherits MarshalByRefObject
   Public Overrides Function InitializeLifetimeService() As [Object]
      Dim lease As ILease = CType(MyBase.InitializeLifetimeService(), _
         ILease)
```

```
        If lease.CurrentState = LeaseState.Initial Then
            lease.InitialLeaseTime = TimeSpan.FromMinutes(2)
            lease.SponsorshipTimeout = TimeSpan.FromMinutes(3)
            lease.RenewOnCallTime = TimeSpan.FromSeconds(3)
        End If
        Return lease
    End Function
End Class
```

Visual C#

```
using System;
using System.Runtime;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Lifetime;


public class MyLifetimeControlObject: MarshalByRefObject
{
    public override object InitializeLifetimeService()
    {
        ILease lease = (ILease)base.InitializeLifetimeService();
        if (lease.CurrentState == LeaseState.Initial)
        {
```

```
            lease.InitialLeaseTime = TimeSpan.FromMinutes(2);
            lease.SponsorshipTimeout = TimeSpan.FromMinutes(3);
            lease.RenewOnCallTime = TimeSpan.FromSeconds(3);
        }
        return lease;
    }
}
```

Làm mới Lifetime Leases

Sau khi bạn tạo ra Lifetime Leases cho một đối tượng, bạn chỉ có thể thay đổi thuộc tính CurrentLeaseTime của đối tượng thuê. Các tài sản CurrentLeaseTime trả về số lượng thời gian còn lại cho thuê. Đây là hai cách để gia hạn hợp đồng thuê:

-Một ứng dụng của khách hàng gọi phương thức ILease.Renew.

-Một ứng dụng gia hạn thời gian thuê lại.

Các mã sau đây cho thấy làm thế nào để gia hạn một hợp đồng thuê bằng cách sử dụng một ứng dụng của máy khách.

Visual Basic .NET

```
Dim obj As New RemoteType()
Dim lease As ILease = CType(RemotingServices.GetLifetimeService(obj), _
    ILease)
Dim expireTime As TimeSpan = lease.Renew(TimeSpan.FromSeconds(30))
```

Visual C#

```
RemoteType obj = new RemoteType();

ILease lease = (ILease)RemotingServices.GetLifetimeService(obj);

TimeSpan expireTime = lease.Renew(TimeSpan.FromSeconds(30));
```

Khi bạn xuất bản các đối tượng từ xa, bạn nên biết các thành viên của các đối tượng từ xa mà được công bố. Máy khách chỉ có thể truy cập các thành viên rằng các đối tượng từ xa xuất bản.

Phạm vi công bố

Hệ thống NET Remoting. Công bố các chức năng của các đối tượng máy chủ để ứng dụng máy khách , giả định rằng các đối tượng được chứa tại cục bộ (local) trong các ứng dụng của máy khách. Tuy nhiên, có một số ngoại lệ đối với các đối tượng được xuất khẩu từ xa.

Bảng 4.1 cung cấp thông tin chi tiết về các đối tượng khác nhau và hệ thống NET Remoting. Quản lý các đối tượng này như thế nào.

Objects	Description
Static members	Bạn không thể xuất ra các giá trị number tĩnh, chẳng hạn như các lĩnh vực và phương pháp, từ xa. NET Remoting cần các thể hiện thành viên(instance member) để cho phép giao tiếp giữa máy khách và các đối tượng máy chủ.
Instance fields and accessors	NET Remoting kiểm tra hệ thống cho dù đối tượng mà bạn sử dụng được export như là một đối tượng proxy. Nếu nó là một đối tượng proxy, mays khách có thể trực

	tiếp truy cập vào các ứng dụng thông qua proxy. Nếu các đối tượng export không phải là một proxy, đối tượng proxy cung cấp thể hiện truy cập cho khách hàng.
Private members	Bạn không thể export các thành phần private của một đối tượng từ xa.
Delegates	Delegates là các đối tượng marshal-by-value. Các đối tượng trong các Delegates có thể là một đối tượng remotable, chẳng hạn như một đối tượng serializable, một đối tượng MarshalByRef, hoặc một đối tượng ContextBound.
Overriding methods on an object	Để nâng cao hiệu suất, các phương thức ảo trên một đối tượng luôn luôn thực hiện tại địa phương(local) trong lĩnh vực ứng dụng, trong đó chúng được gọi tới.

Ngoài các đối tượng được liệt kê trong Bảng 4.1, các lời gọi đến Equals () và phương thức ToString () được thực hiện trong các đối tượng từ xa rằng những phương thức ghi đè trong các đối tượng từ xa. Tuy nhiên, các lời gọi GetHashCode () và MemberwiseClone () phương thức được thực hiện tại local.

Trong bài học này, bạn đã học về máy chủ và các đối tượng Client-Active. Bạn cũng có học để tạo ra các đối tượng Singleton và SingleCall, và thiết lập một kết nối giữa máy chủ và các ứng dụng của máy khách. Ngoài ra, bạn xác định các đối tượng mà không thể được export như là các đối tượng từ xa. Bạn cũng đã học về các hợp đồng cho thuê dài suốt đời và làm thế nào để khởi tạo và gia hạn hợp đồng cho thuê thời gian sống của đối tượng. Cuối cùng, bạn đã học về phạm vi công bố.

3.1.3. Kênh vận chuyển thông tin trên miền ứng dụng

Sự hiểu biết Channels

Kênh cho phép một ứng dụng đang chạy trong một miền ứng dụng, tiến trình, hoặc máy tính để gửi thông tin đến một ứng dụng đang chạy trong một miền ứng dụng khác nhau, tiến trình, hoặc máy tính. Ngoài ra, các kênh cho phép các ứng dụng để gửi và nhận thông tin bằng cách sử dụng các giao thức khác nhau, chẳng hạn như TCP và HTTP.

Kênh làm việc truyền tải thông tin phạm vị vùng biên của remoting, chẳng hạn như miền ứng dụng, tiến trình, và máy tính. Một kênh lắng nghe các thông tin tại một ranh giới remoting và gửi thông tin đến một ranh giới remoting. Trước khi một đối tượng từ xa gửi một thông điệp đến một đối tượng từ xa, một kênh chuyển đổi các thông tin vào một định dạng thích hợp chẳng hạn như là định dạng XML hoặc nhị phân. Trước khi gửi thông tin, kênh thực hiện các chuyển đổi cần thiết.

Channel Interfaces

NET Framework cung cấp không gian tên System.Runtime.Remoting.Channels , bao gồm các giao diện và các lớp mà bạn sử dụng để làm việc với các kênh. Tất cả các kênh thực hiện giao diện IChannel. Giao diện IChannel cung cấp các đặc tính như ChannelName và ChannelPriority, xác định duy nhất một kênh và xác định các ưu tiên kênh, tương ứng.

Tùy thuộc vào việc bạn sử dụng một kênh để nhận hoặc gửi thông tin, các kênh được phân loại như máy thu hoặc máy chủ và các kênh người gửi hoặc Client. Một kênh thu hoặc máy chủ thực hiện các giao diện IChannelReceiver, và một người gửi hoặc kênh máy khách thực hiện các giao diện IChannelSender. Giao diện IChannelReceiver quy định cụ thể phương thức, chẳng hạn như StartListening và StopListening, mà phải được thực hiện bởi một lớp kênh thu. Giao diện IChannelSender xác định một phương pháp, CreateMessageSink, mà

phải được thực hiện bởi một người gửi hoặc lớp kênh máy khách. Phương thức CreateMessageSink tạo ra và trả về một bồn rửa thông báo rằng đối tượng kênh sử dụng để gửi thông tin cho một đối tượng từ xa đặt tại một URL cụ thể.

Các lớp học HttpServerChannel và TcpServerChannel thực hiện giao diện IChannelReceiver, trong khi các lớp HttpClientChannel và TcpClientChannel thực hiện giao diện IChannelSender. Một kênh cũng có thể thực hiện các giao diện IChannelSender và IChannelReceiver, cho phép các kênh để gửi cũng như nhận thông tin. Các lớp học HttpChannel và TcpChannel thực hiện các giao diện IChannelSender và IChannelReceiver, cho phép các đối tượng của các lớp này để gửi và nhận thông tin.

Bảng 4.2 cho thấy giao diện khác nhau và các thành viên của họ được thực hiện bởi các lớp kênh.

Table 4-2. Channel Interfaces

Channel	Description
Public properties of IChannel interface	
ChannelName	Thuộc tính này lấy về tên của Channel
ChannelPriority	Thuộc tính này lấy về mức độ ưu tiên của Channel. Một giá trị số cao cho thấy mức độ ưu tiên cao hơn. Các kênh máy khách với một ưu tiên cao hơn được cơ hội kết nối với một đối tượng từ xa trước các máy khác. Mặt khác, ưu tiên trong trường hợp của các kênh máy chủ cho thấy thứ tự mà máy khách sẽ sử dụng các kênh trong khi kết nối với các đối tượng máy chủ.
Public methods of IChannel interface	

Parse	Phương thức này trả về Uniform Resource Identifier (URI) của kênh hiện tại và đối tượng URI được dùng như một tham số.
Public properties of IChannelReceiver interface	
ChannelData	Thuộc tính này nhận về dữ liệu channel-specific, trong đó bao gồm các thông tin về đối tượng được truy cập từ xa.
Public methods of IchannelReceiver Interface	
StartListening	Phương thức này hướng dẫn một kênh để bắt đầu lắng nghe cho các yêu cầu của máy khách.
StopListening	Phương thức này chỉ thị một kênh để ngừng lắng nghe yêu cầu của khách hàng.
GetUrlsForUri	Phương thức này trả về một mảng của tất cả các URL cho một URI.
Public methods of IChannelSender interface	
CreateMessageSink	Phương thức này trả về thông tin sink objecta nhắn mang thông điệp đến một URL cụ thể hoặc một kênh data object.

Đăng ký một kênh

Trước khi một đối tượng từ xa sẽ gửi thông tin đến một đối tượng từ xa khác, bạn cần đăng ký một kênh máy khách với hệ thống remoting. Tương tự như vậy, để cho phép một thành phần từ xa để nhận thông tin, bạn cần đăng ký một kênh máy chủ với các cơ sở hạ tầng remoting. Để đăng ký các kênh với các cơ sở hạ tầng remoting, bạn sử dụng lớp ChannelServices. Lớp ChannelServices cung cấp các phương thức tĩnh cho phép bạn đăng ký các kênh, giải quyết các URL, và khám phá các đối tượng từ xa bằng cách sử dụng các URL đối tượng. Phương thức tĩnh RegisterChannel của lớp ChannelServices cho phép bạn đăng ký một kênh với cơ sở hạ tầng remoting. Các mã sau đây cho thấy làm thế nào để đăng ký một phương thức TcpServerChannel lắng nghe trên cổng 8010.

Visual Basic .NET

```
Dim channel as New TcpServerChannel(8010)
ChannelServices.RegisterChannel(channel)
```

Visual C#

```
TcpServerChannel channel = new TcpServerChannel(8010);
ChannelServices.RegisterChannel(channel);
```

Sau khi bạn đăng ký một kênh trên các máy tính khách và máy chủ, các đối tượng từ xa có thể gọi lần lượt các phương pháp khác nhau. Phần tiếp theo giải thích làm thế nào các đối tượng từ xa sử dụng các kênh giao tiếp qua các vung biên của remoting.

Lựa chọn kênh cho Remoting

Khi một đối tượng máy khách gọi một phương thức trên một đối tượng từ xa, các kênh mang các thông số và thông tin cuộc gọi khác liên quan đến các đối

tượng từ xa. Một máy khách có thể sử dụng bất kỳ các kênh đăng ký trên máy tính của máy khách gọi một phương thức trên đối tượng từ xa. Ngoài ra, bạn có thể tùy chỉnh một kênh hiện có hoặc tạo ra một kênh mà sử dụng một giao thức vận chuyển khác nhau. Bạn nên áp dụng các quy tắc sau đây khi lựa chọn các kênh cho các đối tượng từ xa:

- Một khách hàng không thể gọi một phương thức trên một đối tượng từ xa, trừ khi bạn đăng ký ít nhất một kênh máy khách, chẳng hạn như TcpClientChannel hoặc HttpClientChannel, với hệ thống truy cập từ xa trên máy tính của máy khách. Ngoài ra, bạn nên đăng ký một kênh máy chủ, chẳng hạn như TcpServerChannel hoặc HttpServerChannel, trước khi đăng ký một đối tượng từ xa.

-Bạn phải đăng ký một kênh máy chủ trên máy tính của máy khách nếu một đối tượng từ xa gọi lại một phương thức trên máy khách.

-Tên kênh trong một miền ứng dụng không thể giống nhau. Tuy nhiên, một miền ứng dụng có thể chứa nhiều kênh khác nhau của cùng kiểu. Ví dụ, một miền ứng dụng có thể có nhiều hơn một đối tượng TcpChannel đăng ký với hệ thống remoting. Thuộc tính ChannelName của các kênh này nên xác định duy nhất những kênh này.

- Hai kênh đăng ký với hệ thống truy cập từ xa trên một máy tính không thể lắng nghe trên cùng một cổng.

Khi một máy khách gọi một đối tượng từ xa, cơ sở hạ tầng remoting tạo ra một message có chứa các thông số và các thông tin liên quan đến cuộc gọi khác. Trong quá trình đó các thông số và các thông tin liên quan đến cuộc gọi được

đóng gói vào một message được biết đến marshaling. Cơ sở hạ tầng remoting gửi message đên một đối tượng RealProxy. Các đối tượng RealProxy, lần lượt, chuyển tiếp message đến một tin nhắn giấu(message sink). Một message sink là một đối tượng mà cho phép một máy khách để thiết lập kết nối với các kênh đăng ký các đối tượng từ xa và chuyển tiếp các tin nhắn vào kênh. Khi các kênh trong miền ứng dụng của các đối tượng từ xa nhận được tin nhắn, hệ thống truy cập từ xa trên máy chủ unmarshals tin nhắn và chuyển tiếp cuộc gọi cùng với các thông số và các thông tin khác liên quan đến cuộc gọi đến một đối tượng từ xa thích hợp.

Lưu ý

Khi hệ thống điều khiển từ xa tạo ra một đối tượng RealProxy, hệ thống truy cập từ xa cũng tạo ra một message sink bằng cách gọi phương thức IChannelSender.CreateMessageSink trên kênh được lựa chọn.

Trong quá trình giao tiếp giữa máy khách và các đối tượng từ xa, trọng tải tin nhắn có chứa các thông số marshaled và thông tin cuộc gọi liên quan đến vận chuyển.Các kênh được vận chuyển thông báo có thể sử dụng giao thức vận chuyển khác nhau, chẳng hạn như HTTP và TCP. Tùy thuộc vào các giao thức vận chuyển được sử dụng, các kênh được chia thành hai không gian tên: System.Runtime.Remoting.Channels.Http

và

System.Runtime.Remoting.Channels.Tcp.

HTTP Channels

Các không gian tên System.Runtime.Remoting.Channels.Http cung cấp các lớp kênh sử dụng giao thức HTTP để vận chuyển thông điệp giữa các đối tượng từ

xa. Các không gian tên System.Runtime.Remoting.Channels.Http cung cấp các lớp, chẳng hạn như HttpClientChannel, HttpServerChannel, và HttpChannel. Bạn có thể sử dụng lớp HttpClientChannel để vận chuyển các tin nhắn từ một khách hàng cho một đối tượng từ xa. Để tạo một đối tượng HttpClientChannel và đăng ký nó với các hệ thống truy cập từ xa, sử dụng đoạn mã sau.

Visual Basic .NET

```
ChannelServices.RegisterChannel(New HttpClientChannel())
```

Visual C#

```
ChannelServices.RegisterChannel(new HttpClientChannel());
```

Lưu ý

Các nhà xây dựng cho các đối tượng HttpClientChannel không yêu cầu bạn phải vượt qua số cổng bởi vì các hệ thống truy cập từ xa tự động phân bổ một cổng có sẵn cho kênh.

HttpServerChannel cho phép một đối tượng từ xa để nghe các cuộc gọi từ xa từ các khách hàng. Bạn phải đăng ký đối tượng HttpServerChannel tại một cổng cụ thể. Các mã sau đây cho thấy làm thế nào để tạo ra và đăng ký một đối tượng HttpServerChannel để lắng nghe tại cổng 8080

Visual Basic .NET

```
Dim channel as New HttpServerChannel(8080)
ChannelServices.RegisterChannel(channel)
```

Visual C#

```
HttpServerChannel channel=new HttpServerChannel(8080);
```

```
ChannelServices.RegisterChannel(channel);
```

Lưu ý

Bạn vượt qua số cổng như là một đối số cho các nhà xây dựng HttpServerChannel xác định cổng mà tại đó các kênh máy chủ lắng nghe cho các cuộc gọi từ xa. Chỉ định một cổng đã được sử dụng gây ra một ngoại lệ được ném ra. Vì vậy, nếu bạn không chắc chắn để xác định số cổng, truyền vào 0 cho tham số khởi tạo. Hệ thống truy cập từ xa phân bổ một cổng có sẵn các kênh máy chủ.

Bạn có thể sử dụng lớp HttpChannel để vận chuyển các tin nhắn đến và đi từ các đối tượng từ xa. Các mã sau đây cho thấy làm thế nào để tạo ra và đăng ký một đối tượng HttpChannel để lắng nghe ở cổng 8010.

Visual Basic .NET

```
Dim channel as New HttpChannel(8010)
ChannelServices.RegisterChannel(channel)
```

Visual C#

```
HttpChannel channel = new HttpChannel(8010);
ChannelServices.RegisterChannel(channel);
```

Lưu ý

Các phương thức khởi tạo trong Visual Basic.NET được xác định với tên mới, trong khi đó trong Visual C # các phương thức khởi tạo có tên giống như tên của lớp.

Table 4-3. Constructors of HTTP Channel Classes

Constructor	Description
HttpChannel class	
New() or HttpChannel()	Các nhà Constructor mặc định khởi tạo tất cả các trường là Các HttpChannel class functions cũng như các Channel trên máy khách khi bạn tạo ra một Constructor
New(port) or HttpChannel(port)	Constructor này cho phép các đối tượng HttpChannel hoạt động như một kênh trên máy khách cũng như một kênh máy chủ. *Port* là cổng được chỉ định cho đối tượng HttpChannel lắng nghe đối tượng Remote gọi tới.
New(properties, clientchannelsinkprovider, serverchannelsinkprovider) or HttpChannel(properties, clientchannelsinkprovider, serverchannelsinkprovider)	Thuộc tính là một đối tượng IDictionary có chứa một tập hợp các thuộc tính kênh trong cặp key-value. Clientchannelsinkprovider là một đối tượng IClientChannel-SinkProvider tạo ra client channel sinks. Đối tượng cung cấp dịch vụ Serverchannelsink là một đối tượng IServerChannelSinkProvider tạo ra server channel sinks thông qua đó thông tin được lưu thông.
HttpClientChannel class	

New() or HttpClientChannel()	Constructor này khởi tạo một thể hiện mới của lớp HttpClientChannel với các giá trị mặc định.
New(properties, clientchannelsinkprovider) or HttpClientChannel(properties, clientchannelsinkprovider)	Constructor khởi tạo các lớp HttpClientChannel và thiết lập các thuộc tính theo các cặp key-value trị trong đối tượng thuộc tính. Clientchannelsinkprovider là đối tượng IClientChannelSinkProvider và đối tượng HttpClientChannel sử dụng để tạo ra một client channel sink. Các thông tin được gửi bởi các kênh máy khách thông qua một chuỗi các *client channel sinks*.
New(name, clientchannelsinkprovider) or HttpClientChannel(name, clientchannelsinkprovider)	Constructor này khởi tạo các đối tượng HttpClientChannel và thiết lập thuộc tính ChannelName của nó để đặt tên. Clientchannelsinkprovider là đối tượng IClientChannel-SinkProvider và đối tượng HttpClientChannel sử dụng để tạo ra một client channel sinks. Các tin nhắn được gửi bởi các kênh trên máy khách thông qua một chuỗi các *client channel sinks*.

HttpServerChannel class

New() or HttpServerChannel()	Constructor khởi tạo các đối tượng HttpServerChannel với các giá trị mặc định.
New(port) or HttpServerChannel(port)	Constructor khởi tạo các đối tượng HttpServerChannel để lắng nghe tại cổng quy định.
New(name, port) or HttpServerChannel(name,	Constructor khởi tạo các đối tượng

port)	HttpServerChannel với các thuộc tính ChannelName thiết lập để đặt tên và thiết lập cổng và kênh lắng nghe cho các cuộc lời gọi phương thức từ xa.
New(properties, serverchannelsinkprovider) or HttpServerChannel(properties, serverchannelsinkprovider)	Constructor khởi tạo các đối tượng HttpServerChannel và thiết lập các thuộc tính theo các cặp key-value của thuộc tính trong đối tượng. Serverchannelsinkprovider là đối tượng IServerChannelSinkProvider và đối tượng HttpServerChannel sử dụng để tạo ra server channel sink. Các tin nhắn nhận được bởi các kênh máy chủ thông qua một chuỗi các server channel sink.

Bạn sử dụng kênh HTTP khi khả năng tương tác giữa các thành phần từ xa là mục tiêu chính. HTTP kênh sử dụng lớp SoapFormatter để serialize các tin nhắn trong định dạng XML bằng cách sử dụng giao thức SOAP trước khi gửi các trọng tải tin nhắn qua kênh. Khả năng tương tác của thông điệp SOAP cho phép một máy khách gọi tới các phương thức trên một đối tượng từ xa mà có thể không được sử dụng các NET Framework. Tuy nhiên, nếu hiệu suất đặt là vấn đề chính, thì sử dụng các kênh cho phép bạn để vận chuyển các tin nhắn trong các định dạng nhị phân. Các không gian tên System.Runtime.Remoting.Channels.Tcp cung cấp các lớp kênh cho phép bạn để vận chuyển các tin nhắn đến và đi từ các đối tượng từ xa. Phần tiếp theo sẽ thảo luận về các lớp học được cung cấp bởi các không gian tên System.Runtime.Remoting.Channels.Tcp.

Lưu ý

Tuần tự hóa là một quá trình mà trạng thái của một đối tượng được lưu trữ trên một đĩa hoặc vận chuyển qua một dây dẫn. Deserialization liên quan đến việc tái tạo một đối tượng bằng cách đọc các thông tin trạng thái tồn tại của đối tượng từ một đĩa hoặc trên dây dẫn.

TCP Channels

TCP cho phép bạn để vận chuyển các tin nhắn trên các lĩnh vực ứng dụng bằng cách sử dụng giao thức TCP. Các kênh TCP sử dụng lớp BinaryFormatter để tuần tự tin nhắn thành một luồng nhị phân (binary stream) trước khi một máy khách gửi chúng cho một đối tượng từ xa. Ngoài ra, lớp BinaryFormatter deserializes vận chuyển nhị phân trước khi cung cấp cho các đối tượng từ xa. Các lớp TCP kênh mà bạn có thể sử dụng bao gồm TcpClientChannel, TcpServerChannel, và TcpChannel. Lớp TcpClientChannel cho phép một máy khách để gửi tin nhắn đến một đối tượng từ xa, trong khi lớp TcpServerChannel cho phép một đối tượng từ xa để nhận tin nhắn từ máy khách. Lớp TcpChannel cho phép bạn để vận chuyển các tin nhắn đến và đi từ một đối tượng từ xa. Bạn đăng ký một kênh TCP theo cùng một cách mà bạn đăng ký một kênh HTTP. Các mã sau đây cho thấy làm thế nào để đăng ký TCP kênh.

Visual Basic .NET

```
ChannelServices.RegisterChannel(New TcpClientChannel())
Dim channel as New TcpServerChannel(8070)
ChannelServices.RegisterChannel(channel)
Dim tcpchannel as New TcpChannel(8010)
```

```
ChannelServices.RegisterChannel(tcpchannel)
```

Visual C#

```
ChannelServices.RegisterChannel(new TcpClientChannel());

TcpServerChannel channel=new TcpServerChannel(8070);

ChannelServices.RegisterChannel(channel);

TcpChannel tcpchannel = new TcpChannel(8010);

ChannelServices.RegisterChannel(tcpchannel);
```

Bảng 4.4 shows the constructor methods of various TCP channel classes.

Constructor	Description
TcpChannel class	
New() or TcpChannel()	Các Hàm khởi tạo mặc định tất cả các lĩnh trường. Khi bạn sử dụng constructor mặc định, các chức năng lớp TcpChannel như là kênh trên máy khách.
New(port) or TcpChannel(port)	Constructor này cho phép các đối tượng TcpChannel để hoạt động như một kênh trên máy khách cũng như một kênh máy chủ. *port number* chỉ định cổng mà đối tượng TcpChannel lắng nghe cuộc gọi từ xa.
New(properties, clientchannelsinkprovider, serverchannelsinkprovider) or TcpChannel(properties, clientchannelsinkprovider, serverchannelsinkprovider)	*Properties* là đối tượng IDictionary có chứa một tập hợp các thuộc tính kênh trong cặp key-value. Clientchannelsinkprovider là một đối tượng IClientChannelSink-Provider tạo ra *client channel sinks* qua các luồng thông tin. Serverchannelsink-provider là một đối

	tượng IServerChannelSinkProvider tạo ra server channel sinks thông qua các tin nhắn mà luồng.
TcpClientChannel class	
New() or TcpClientChannel()	Constructor khởi tạo các đối tượng TcpClientChannel với các giá trị mặc định.
New(properties, clientchannelsinkprovider) or TcpClientChannel(properties, clientchannelsinkprovider)	Constructor khởi tạo các đối tượng TcpClientChannel và thiết lập các thuộc tính theo các cặp Key-value trong đối tượng. Clientchannelsinkprovider là đối tượng IClientChannelSinkProvider và đối tượng TcpClientChannel sử dụng để tạo ra *client channel sinks*. Các tin nhắn được gửi bởi các kênh máy khách thông qua một chuỗi các *client channel sinks*.
New(name, clientchannelsinkprovider) or TcpClientChannel(name, clientchannelsinkprovider)	Constructor này khởi tạo các đối tượng TcpClientChannel và thiết lập thuộc tính ChannelName của nó để đặt tên. Clientchannelsinkprovider là đối tượng IClientChannelSink-nhà cung cấp các đối tượng TcpClientChannel sử dụng để tạo *client channel sinks*. Các tin nhắn được gửi bởi các kênh trên máy khách thông qua một chuỗi các *client channel sinks*.
TcpServerChannel class	
New(port) or TcpServerChannel(port)	Constructor khởi tạo các đối tượng TcpServerChannel để lắng nghe một cổng

	được chỉ định.
New(name, port) or TcpServerChannel(name, port)	Constructor khởi tạo các đối tượng TcpServerChannel với các thuộc tính ChannelName thiết lập để đặt tên và thiết lập cổng kênh lắng nghe cho các lời gọi phương thức từ xa.
New(properties, serverchannelsinkprovider) or TcpServerChannel(properties, serverchannelsinkprovider)	Constructor khởi tạo các đối tượng TcpServerChannel và thiết lập các thuộc tính theo các cặp key-value trong đối tượng. Serverchannelsinkprovider là đối tượng IServerChannelSinkProvider và đối tượng TcpServerChannel sử dụng để tạo ra *server channel sink*. Các thông báo về các kênh máy chủ nhận được được định tuyến thông qua một chuỗi các *server channel sinks*..

Lưu ý

Không có constructor mặc định cho lớp TcpServerChannel.

Nếu một đối tượng từ xa sử dụng một đối tượng TcpChannel hoặc TcpServerChannel để nhận thông tin từ các máy khách, các máy khách có thể gọi các phương pháp trên các đối tượng từ xa nếu họ sử dụng hoặc là một TcpChannel hoặc đối tượng TcpClientChannel. Nừu máy khách sử dụng một đối tượng HttpClientChannel hoặc HttpChannel để kết nối với một đối tượng từ xa sử dụng một đối tượng TcpChannel hoặc TcpServerChannel, một ngoại lệ được ném ra. Máy khách nhìn thấy thông báo lỗi, "Kết nối tiềm ẩn đã được ngừng

kết nối: Một lỗi không mong muốn xảy ra trên một quá trình nhận.” - ” The underlying connection was closed: An unexpected error occurred on a receive. "

Quyết định lựa chọn HTTP hoặc TCP kênh phụ thuộc vào các tính năng khác nhau của hai loại kênh trên.

Bảng 4.5 cho thấy và so sánh các tính năng của các kênh HTTP và TCP.

Cho dù bạn chọn HTTP hoặc TCP kênh cho các ứng dụng từ xa của bạn, các kênh thực hiện các nhiệm vụ độc lập của các loại kênh. Các thông điệp mà các kênh gửi hoặc nhận được từ các đối tượng từ xa qua một chuỗi các đối tượng được gọi là channel sinks. Các đối tượng channel sinks thực hiện các nhiệm vụ như định dạng, vận chuyển, và stack building. Trong phần tiếp theo, bạn sẽ tìm hiểu về *channel sinks* và *sink chains* và hiểu được vai trò của chúng trong remoting.

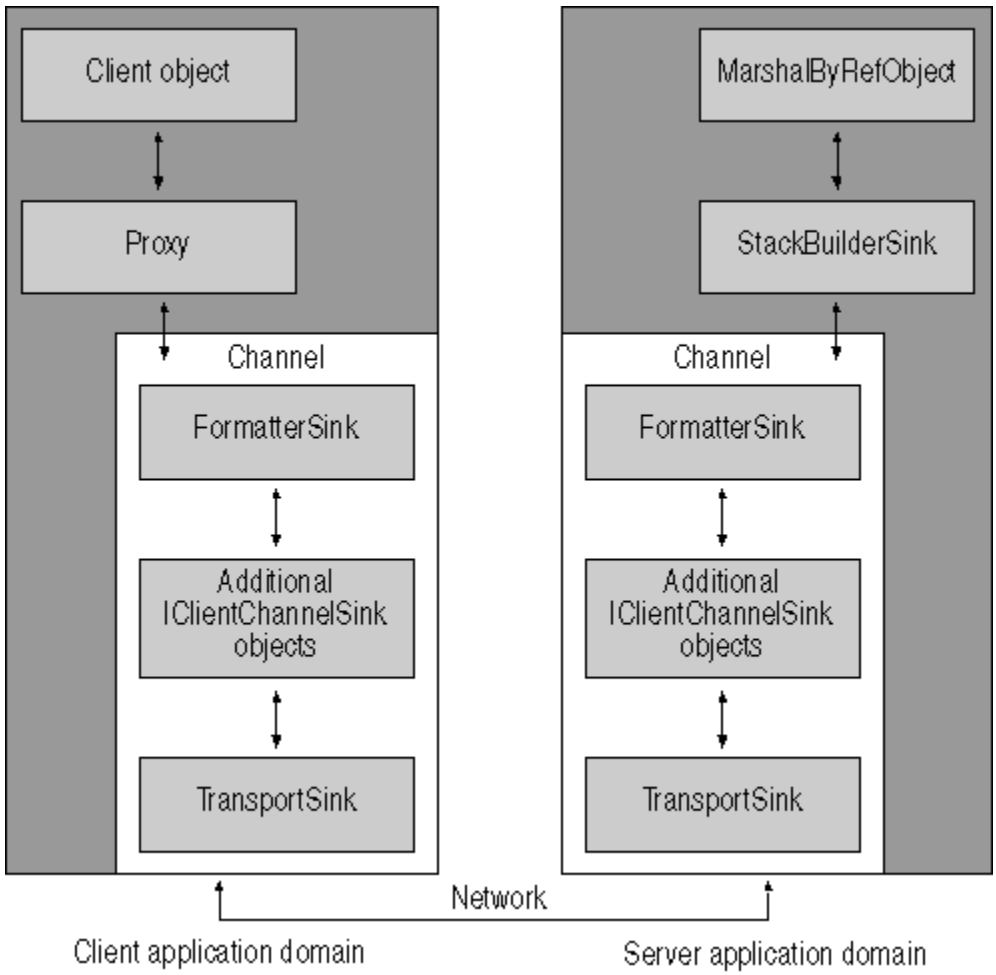
Bảng 4-5. Các tính năng của HTTP và TCP Channels

HTTP channels	TCP channels
Truyền tải thông tin và đi từ các đối tượng từ xa bằng cách sử dụng giao thức HTTP.	Truyền tải thông tin đi và đến từ các đối tượng từ xa bằng cách sử dụng giao thức TCP.
Sử dụng lớp SoapFormatter để serialize và deserialize tin nhắn.	Sử dụng lớp BinaryFormatter để serialize và deserialize tin nhắn.
Tạo hai kết nối theo mặc định để kết nối với một máy chủ nhất định. Bạn	Tạo ra nhiều kết nỗi cũng như số lượng các luồng trên client để yêu cầu. Những

có thể cấu hình kênh này bằng cách thay đổi các thuộc tính clientConnectionLimit trong một tập tin cấu hình ứng dụng.	kết nối socket được đóng cửa nếu các kết nối vẫn không hoạt động trong 15 - 20 giây
Tạo ra và tiêu thụ Channel-DataStore cho các đối tượng tham chiếu.	Tạo ra và tiêu thụ ChannelDataStore cho các đối tượng tham chiếu.

Sinks and Sink Chains

Kênh kết nối các đối tượng qua các biên giới remoting bằng cách cho phép họ gửi tin nhắn cho nhau. Các tin nhắn được thực hiện bởi các kênh đi qua một chuỗi các *channel sinks*. Một *channel sinks* thực hiện một số chức năng nhất định về tin nhắn trước khi chuyển tiếp tin nhắn tới *channel sinks* tiếp theo trong chuỗi. Trong *channel sinks* bạn có thể truy cập vào các thông điệp được chuyển đến hoặc đến từ một đối tượng từ xa. Trong một *channel sinks*, bạn có thể thực hiện các nhiệm vụ, chẳng hạn như khai thác *logging*, áp dụng bộ lọc, mã hóa tin nhắn, và áp đặt các hạn chế bảo mật. Hình 4.2 cho thấy *channel sinks* trên máy khách và máy chủ.



Bạn sử dụng kênh cung cấp *channel sink* để tạo ra *sink chains*. Chúng bao gồm các đối tượng thực hiện IClientChannelSinkProvider, IClientFormatterSinkProvider, hoặc IServerChannelSinkProvider giao diện. Khi bạn kích hoạt một đối tượng từ xa, hệ thống truy cập từ xa lấy *channel sink provider* hiện tại và gọi phương thức CreateSink trong *channel sink provider* để tạo ra các kênh đầu tiên trong chuỗi. *channel sink* đầu tiên trên một máy khách phải là một *formatter sink* (định dạng sink) mà serializes tin nhắn vào một luồng. Sink cuối cùng trên *channel sink chain* có *sink* vận chuyển và gửi các dòng trên mạng.

Các mã sau đây cho thấy làm thế nào để tạo ra một *channel sink chain*..

Visual Basic .NET

```
private Function CreateSinkChain() As IClientChannelSinkProvider
    Dim chain As New FormatterSinkProvider_1
    Dim sink As IClientChannelSinkProvider
    sink = chain
    sink.Next = New FormatterSinkProvider_2
    sink = sink.Next
    return chain
End Function
```

Visual C#

```
private IClientChannelSinkProvider CreateSinkChain()
{
    IClientChannelSinkProvider chain = new FormatterSinkProvider_1();
    IClientChannelSinkProvider sink = chain;
    sink.Next = new FormatterSinkProvider_2();
    sink = sink.Next;
    return chain;
}
```

Lưu ý

Các lớp FormatterSinkProvider_1 và FormatterSinkProvider_2 trong đoạn code trên là các lớp mà bạn tạo ra để tạo ra định dạng tùy chỉnh.

3.2. Trình tự thao tác

Tạo và sử dụng các đối tượng .Net Remoting

Tạo thư viện ChatCoordinator.dll

Bạn sẽ tạo ra thư viện ChatCoordinator.dll. Thư viện ChatCoordinator.dll bao gồm các lớp học ChatCoordinator và SubmitEventArgs. Để tạo thư viện ChatCoordinator.dll, thực hiện các bước sau:

1- Mở Notepad và gõ đoạn mã sau.

Visual Basic .NET

```
Imports System
Imports System.Runtime.Remoting
Imports System.Collections


<Serializable()> _
Public Class SubmitEventArgs
    Inherits EventArgs
    Private message As String = Nothing
    Private username As String = Nothing
    Public Sub New(ByVal contribution As String, ByVal contributor _
        As String)
        Me.message = contribution
        Me.username = contributor
```

```
    End Sub 'New

    Public ReadOnly Property Contribution() As String
        Get
            Return message
        End Get
    End Property

    Public ReadOnly Property Contributor() As String
        Get
            Return username
        End Get
    End Property
End Class 'SubmitEventArgs

Public Delegate Sub SubmissionEventHandler(ByVal sender As Object, _
    ByVal submitArgs As SubmitEventArgs)

Public Class ChatCoordinator
    Inherits MarshalByRefObject
    Public Sub New()
```

```
        Console.WriteLine("ChatCoordinator created. Instance: " & _
            Me.GetHashCode().ToString())
    End Sub 'New
    Public Overrides Function InitializeLifetimeService() As Object
        Return Nothing
    End Function 'InitializeLifetimeService

    Public Event Submission As SubmissionEventHandler

    Public Sub Submit(ByVal contribution As String, ByVal contributor _
        As String)
        Console.WriteLine("{0} says: {1}.", contributor, contribution)
        Dim e As New SubmitEventArgs(contribution, contributor)
        RaiseEvent Submission(Me, e)
    End Sub 'Submit
End Class 'ChatCoordinator
```

Visual C#

```
using System;
using System.Runtime.Remoting;
using System.Collections;
```

```
[Serializable]
public class SubmitEventArgs : EventArgs{


   private string _string = null;
   private string _alias = null;


   public SubmitEventArgs(string contribution, string contributor)
   {
      this._string = contribution;
      this._alias = contributor;
   }


   public string Contribution
   {
      get{ return _string; }
   }


   public string Contributor
   {
      get { return _alias; }
   }
```

```
}
public delegate void SubmissionEventHandler(object sender,
    SubmitEventArgs submitArgs);



public class ChatCoordinator : MarshalByRefObject
{


    public ChatCoordinator()
    {
        Console.WriteLine("ChatCoordinator created. Instance: " +
            this.GetHashCode().ToString());
    }



    public override object InitializeLifetimeService()
    {
        return null;
    }


    public event SubmissionEventHandler Submission;
```

```
   public void Submit(string contribution, string contributor)
   {
      Console.WriteLine("{0} sent: {1}.", contributor, contribution);
      SubmitEventArgs e = new SubmitEventArgs(contribution,
         contributor);

      if (Submission != null)
      {
         Console.WriteLine("Broadcasting...");
         Submission(this, e);
      }
   }
}
```

2-Lưu tập tin như ChatCoordinator.vb hoặc ChatCoordinator.cs.

3-Mở Studio Command Prompt Visual. Loại VBC / t: library ChatCoordinator.vb để biên dịch mã trong ChatCoordinator.vb hoặc loại csc / t: library ChatCoordinator.cs để biên dịch mã trong ChatCoordinator.cs.

Tạo một tập tin cấu hình Cấu hình các đối tượng ChatCoordinator

Bạn sẽ tạo ra một tập tin cấu hình để xác định các cài đặt cho các đối tượng ChatCoordinator remotable mà bạn đã tạo trong bài tập trước. Để tạo tập tin cấu hình, thực hiện các bước sau:

1- Mở Notepad và gõ đoạn mã sau.

```
XML
<configuration>
   <system.runtime.remoting>
      <application>
      <service>
         <wellknown
            mode="Singleton"
            type="ChatCoordinator, ChatCoordinator"
            objectUri="Chat"
         />
      </service>
      <channels>
      <channel
         ref="http"
         port="8080"
      />
      </channels>
   </application>
   </system.runtime.remoting>
</configuration>
```

2- Lưu tập tin như Central.config.

Tạo một ứng dụng Console để Kích hoạt các đối tượng ChatCoordinator

Bạn sẽ tạo ra một giao diện điều khiển ứng dụng để kích hoạt các đối tượng ChatCoordinator bằng cách sử dụng các thiết lập cấu hình từ xa được xác định trong các tập tin Central.config. Để tạo ra các giao diện điều khiển ứng dụng, thực hiện các bước sau:

1- Opens Notepad and type đoạn mã sau.

Visual Basic .NET

```
Imports System
Imports System.Diagnostics
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels


Public Class Server
   Public Shared Sub Main()
      RemotingConfiguration.Configure("Central.config")
      Console.WriteLine("The host application is currently " & _
         "running. Press Enter to exit.")
      Console.ReadLine()
   End Sub 'Main
End Class 'ServerProcess
```

Visual C#

```
using System;
using System.Runtime.Remoting;


public class Server
{
   public static void Main(string[] Args)
   {
      RemotingConfiguration.Configure("Central.config");
      Console.WriteLine("The host application is currently " +
         "running. Press Enter to exit.");
      Console.ReadLine();
   }
}
```

2- Lưu tập tin như Server.vb hoặc Server.cs.

3-Biên dịch mã trong file Server.vb bằng cách gõ VBC / r: ChatCoordinator.dll Server.vb hoặc biên dịch nó trong các tập tin Server.cs bằng cách gõ csc / r: ChatCoordinator.dll Server.cs tại của Visual Studio NET Command Prompt .

Tạo Client Chat

Bạn sẽ tạo ra một ứng dụng chat client dựa trên giao diện điều khiển. Các ứng dụng chat client bao gồm lớp ChatClient, gọi các phương thức trên đối tượng ChatCoordinator remotable. Lớp ChatClient chứa một phương pháp Run và một phương thức SubmissionReceiver. Các phương thức Run cấu hình các đối

tượng máy khách. Các giá trị cấu hình được quy định cụ thể trong file Client.config. Để tạo ra các ứng dụng chat client, thực hiện các bước sau:

1- Mở Notepad và gõ đoạn mã sau.

Visual Basic .NET

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports Microsoft.VisualBasic

Public Class ChatClient
    Inherits MarshalByRefObject

  Public Overrides Function InitializeLifetimeService() As Object
    Return Nothing
  End Function

  Private username As String = Nothing

  Public Sub New(ByVal [alias] As String)
    Me.username = [alias]
  End Sub

  Public Sub Run()
    RemotingConfiguration.Configure("Client.config")
    Dim chatcenter As New ChatCoordinator()
    AddHandler chatcenter.Submission, AddressOf Me.SubmissionReceiver
    Dim userInput As String = ""
    While True
```

```
            Console.WriteLine("Press 0 (zero) and ENTER to Exit:")
            userInput = Console.ReadLine()
            If userInput = "0" Then Exit While
            chatcenter.Submit(userInput, username)
        End While
        RemoveHandler chatcenter.Submission, AddressOf _
            Me.SubmissionReceiver
    End Sub

    Public Sub SubmissionReceiver(ByVal sender As Object, ByVal args As _
        SubmitEventArgs)
        If args.Contributor = username Then
            Console.WriteLine("Your message was broadcast.")
        Else
            Console.WriteLine(args.Contributor & " says:" & _
                args.Contribution)
        End If
    End Sub 'SubmissionReceiver

    Public Shared Sub Main()
        Dim Args() As String = Environment.GetCommandLineArgs()
        If Args.Length <> 2 Then
            Console.WriteLine("You need to type an alias.")
            Return
        End If
        Dim client As New ChatClient(Args(1))
        client.Run()
    End Sub 'Main
End Class 'ChatClient
```

Visual C#

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;

public class ChatClient : MarshalByRefObject
{
   private string username = null;
   public override object InitializeLifetimeService()
   {
      return null;
   }

   public ChatClient(string alias)
   {
      this.username = alias;
   }

   public void Run()
   {
      RemotingConfiguration.Configure("Client.config");
      ChatCoordinator chatcenter = new ChatCoordinator();
      chatcenter.Submission += new
         SubmissionEventHandler(this.SubmissionReceiver);
      String keyState = "";
      while (true)
      {
         Console.WriteLine("Press 0 (zero) and ENTER to Exit:\r\n");
```

```
        keyState = Console.ReadLine();
        if (String.Compare(keyState,"0", true) == 0)
            break;
        chatcenter.Submit(keyState, username);
    }

    chatcenter.Submission -= new
        SubmissionEventHandler(this.SubmissionReceiver);
}

public void SubmissionReceiver(object sender, SubmitEventArgs args)
{
    if (String.Compare(args.Contributor, username, true) == 0)
    {
        Console.WriteLine("Your message was broadcast.");
    }
    else
        Console.WriteLine(args.Contributor
            + " says: " + args.Contribution);
}

public static void Main(string[] Args)
{
    if (Args.Length != 1)
    {
        Console.WriteLine("You need to type an alias.");
        return;
    }
```

```
        ChatClient client = new ChatClient(Args[0]);
        client.Run();
    }
}
```

2-Lưu tập tin như Client.vb hoặc Client.cs.

3-Biên dịch mã trong Client.vb bằng cách gõ VBC / r: ChatCoordinator.dll Client.vb hoặc biên dịch nó bằng cách gõ Client.cs csc / r: ChatCoordinator.dll Client.cs Visual Studio NET Command Prompt.

Tạo một tập tin cấu hình để cấu hình các đối tượng ChatClient

Bạn sẽ tạo ra một tập tin Client.config định các thiết lập để cấu hình các đối tượng ChatClient. Để tạo file Client.config, thực hiện các bước sau:

1- Mở Notepad và gõ đoạn mã sau.

```
XML
<configuration>
   <system.runtime.remoting>
      <application>
      <client>
      <wellknown
         type="ChatCoordinator, ChatCoordinator"
         url="http://localhost:8080/Chat"
      />
      </client>
      <channels>
      <channel
         ref="http"
```

```
        port="0"
      />
      </channels>
      </application>
   </system.runtime.remoting>
</configuration>
```

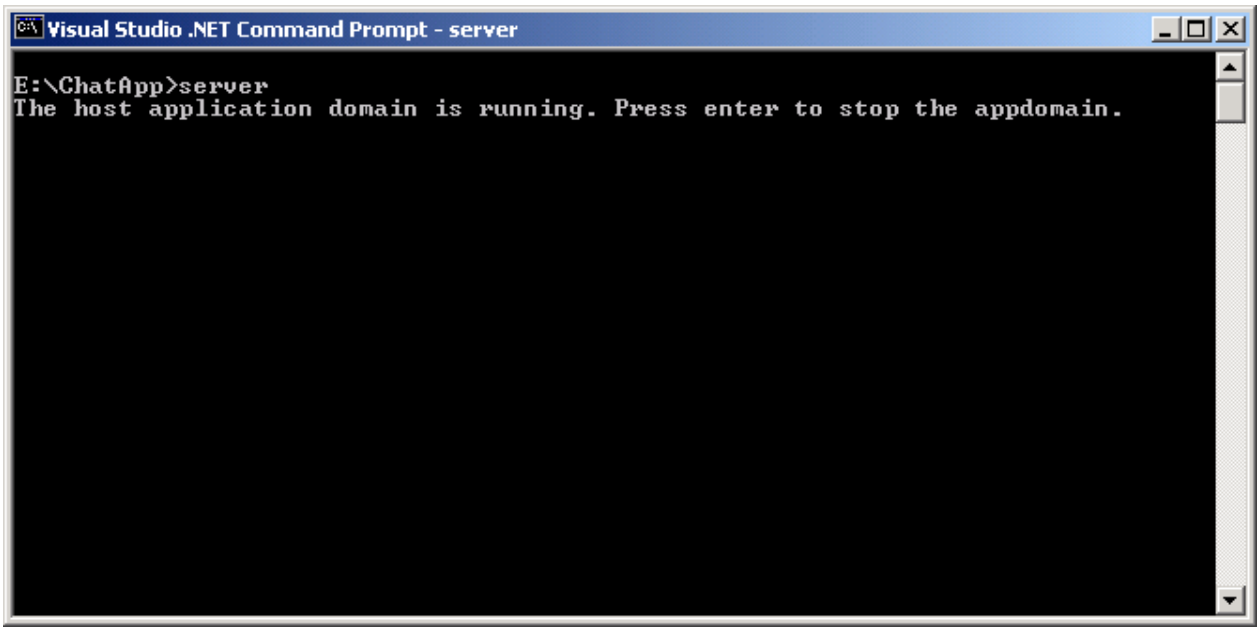
2- Lưu tập tin với tên Client.config.

Thực hiện các ứng dụng chat

Bạn sẽ chạy ứng dụng chat mà bạn đã tạo trong các bài tập trước. Để chạy các ứng dụng chat, thực hiện các bước sau:

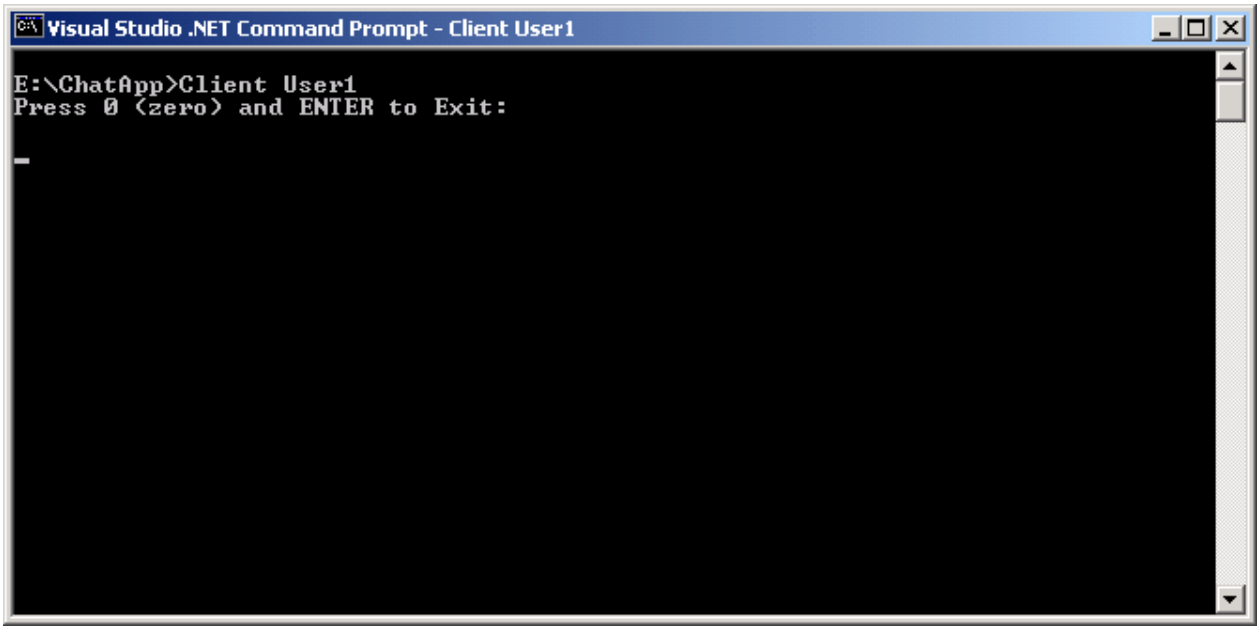
1- Mở Command Studio. Visual Prompt vào thư mục mà bạn đã lưu server.exe, ChatCoordinator.dll, và các tập tin Server.config.

2- Gõ vào Server tại dấu nhắc lệnh. Hiện ra như hình dưới đây.



3- Mở một lệnh Studio Visual Prompt vào thư mục mà bạn đã lưu Client.exe, ChatCoordinator.dll, và các tập tin Client.config.

4-Gõ vào Client UserName tại dấu nhắc lệnh. Như hình dưới đây là hiển thị khi gõ vào Client User1

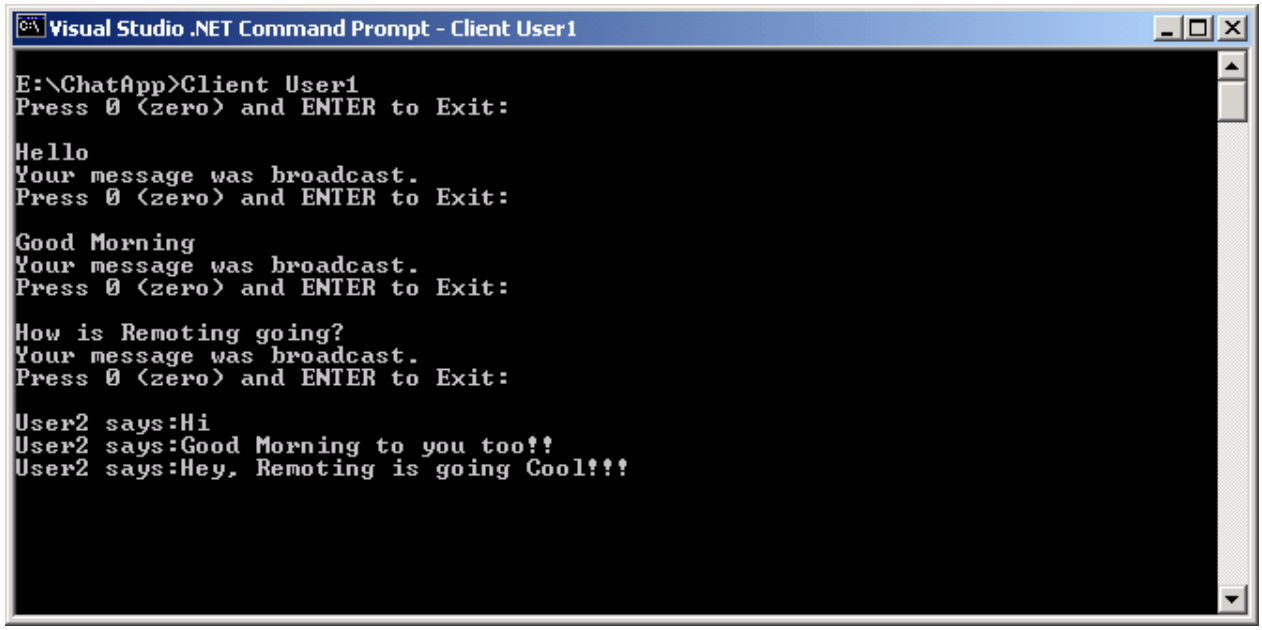


5- Mở một vài Visual Studio .NET Command Prompts và truy cập đến đường dẫn thư mục mà bạn đã lưu Client.exe, ChatCoordinator.dll, và các tập tin Client.config.

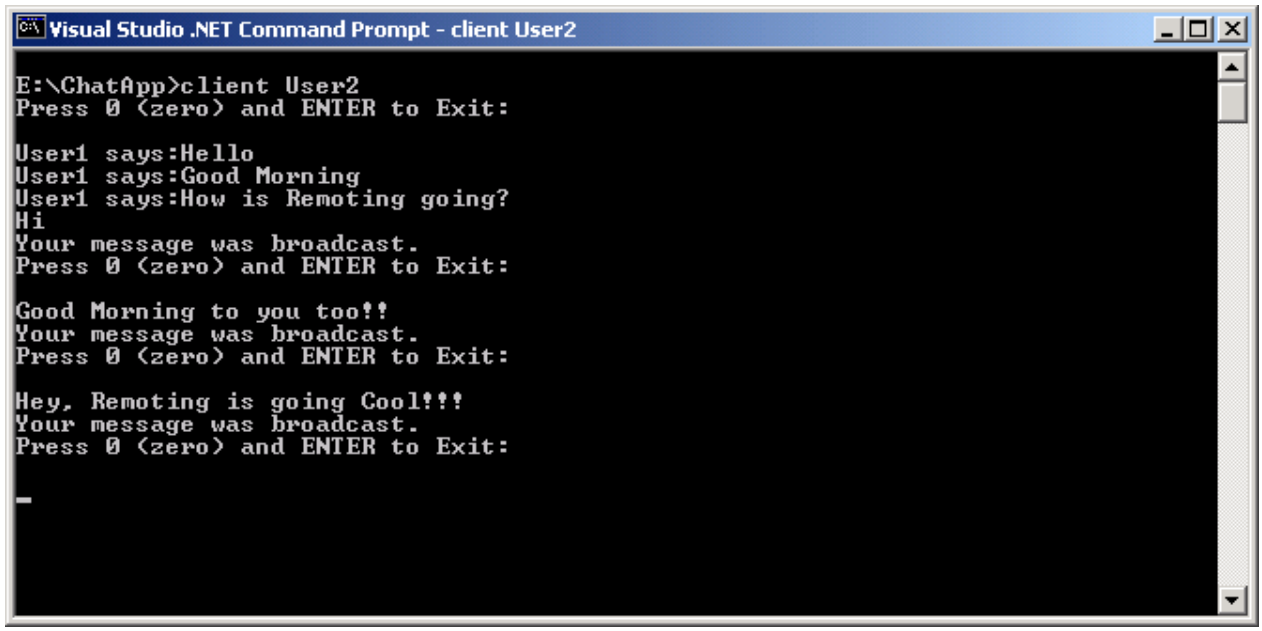
6-Gõ ***Client UserName*** tại dấu nhắc lệnh, sử dụng tên người dùng khác nhau.

7-Gửi tin nhắn từ các ứng dụng của máy khách. Một số thông tin của User1 gửi và nhận được từ những người dùng khác được hiển thị ở đây.

Gửi tin nhắn từ các ứng dụng của khách hàng. Một số thông điệp rằng User1 gửi và nhận được từ những người dùng khác được hiển thị ở đây.



Các thông tin mà User2 nhận được từ và gửi cho những người dùng khác được hiển thị ở đây.



III. Tóm tắt trình tự thực hiện hoặc quy trình công nghệ

STT	*Tên các bước công việc*	*Dụng cụ, thiết bị, vật tư*	*Yêu cầu kỹ thuật*	*Các chú ý về an toàn lao động*
1	Tạo thư viện ChatCoordinator.dll	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	
2	Tạo một tập tin cấu hình Cấu hình các đối tượng ChatCoordinator	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	
3	Tạo một ứng dụng Console để Kích hoạt các đối tượng ChatCoordinator	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	
4	Tạo Client Chat	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	
5	Tạo một tập tin cấu hình để cấu hình các đối tượng ChatClient	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	
6	Thực hiện các ứng dụng chat	Notepad, Visual Studio.NET	Thực hiện đúng các bước của trình tự thực hiện	

IV. TÀI LIỆU THAO KHẢO

1) GIÁO TRÌNH -MCAD/MCSD

DEVELOPINGXML WEB SERVICE SERVER COMPONENTS WITH MICROSOFT VISUAL BASIC.NET WITH MICROSOFT VISUAL C#.NET

2)MSDN LIBRARY

V. MỤC LỤC