Bài 6

SERVICES TRONG WINDOWS COMMUNICATION FOUNDATION

Mục lục

Bài này sẽ tập trung vào các thành phần cho riêng phía dịch vụ của Windows Communication Foundation. Thực chất trong các bài trước chúng ta đã nói tới các chủ đề và khái niệm có liên quan đến dịch vụ và client, rất nhiều trong số chúng vẫn áp dụng được cho phía dịch vụ. Tuy vậy, vẫn còn có một số khái niệm phía dịch vụ chưa được nhắc tới.

Bài này sẽ tập trung vào các vấn đề sau:

- Tổng quan về các khái niệm phía dịch vụ của Windows Communication Foundation
- Các phép hành xử (behaviors)
- Xử lý lỗi

1 Tổng quan

1.1 Các kiểu dịch vụ

Có ba kiểu dịch vụ trong WCF là: Định kiểu, Không định kiểu, và Bản tin định kiểu. Sau đây chúng ta sẽ xem xét chi tiết các kiểu dịch vụ của WCF.

1.1.1 Typed Service (Định kiểu)

Dịch vụ có định kiểu là loại dịnh vụ đơn giản nhất trong số ba kiểu dịch vụ. Tuy vậy nó cung cấp phần lớn các tính năng cần thiết để bạn phát triển các dịch vụ WCF.

Một thuật ngữ hay dùng cho dịch vụ định kiểu là "mô hình tham số", mô hình này định nghĩa một cách chính xác kiểu dịch vụ này làm gì. Các dịch vụ định kiểu không giới hạn ở việc định kiểu của các tham số hay kết quả trả về. Các dịnh vụ kiểu này có thể chấp nhận các kiểu đơn giản cũng như phức hợp. Tuy nhiên khi truyền các tham số hay trả về giá trị kiểu phức hợp, ta cần phải định nghĩa data contract cho các kiểu đó.

Dịch vụ định kiểu trong Windows Communication Foundation xử lý tất cả các bản tin, bạn không cần phải làm việc trực tiếp ở mức bản tin. Ví dụ sau đây mô tả một dịch vụ định kiểu. Contract dịch vụ định nghĩa 2 thao tác. Thao tác thứ nhất không cần tham số, thao tác thứ 2 chấp nhận 2 tham số kiểu là int (số nguyên):

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    string SayHello();
    [OperationContract]
    string GetData(int firstValue, int secondValue);
```

```
}
```

Dịch vụ định kiểu còn hỗ trợ các kiểu tham số với từ khoá `ref` và `out`.

1.1.2 Untyped service (Không định kiểu)

Các dịch vụ không định kiểu hơi phức tạp hơn so với dịch vụ định kiểu do chúng làm việc trực tiếp với bản tin. Trong kiểu dịch vụ này, bạn định nghĩa các bản tin và nội dung của chúng. Nghĩa là với kiểu dịch vụ này, bạn làm việc ở mức bản tin, các đối tượng bản tin được chuyển qua lại giữa client và dịch vụ, và dịch vụ có thể trả về một đối tượng bản tin nếu được yêu cầu. Hơn nữa, kiểu dịch vụ này cho bạn khả năng truy nhập tới nội dung bản tin.

Ví dụ sau đây mô tả cách tạo ra dịch vụ không định kiểu, trong ví dụ này, thay vì gửi các kiểu là các tham số, các đối tượng bản tin được chuyển qua lại giữa client và dịch vụ:

```
[ServiceContract]
public interface IServiceClass
{
    [OperationContract]
    string InitiateOrder();
    [OperationContract]
    BookOrder PlaceOrder(BookOrder request);
    [OperationContract]
    string FinalizeOrder();
}
[MessageContract]
public class BookOrder
{
    private string isbn;
    private int quantity;
    private string ordernumber;
    public BookOrder(BookOrder message)
    {
        isbn = message.isbn;
        quantity = message.quantity;
    }
    [MessageHeader]
    public string ISBN
    {
        get { return isbn; }
        set { isbn = value; }
    }
```

```
        [MessageBodyMember]
        public int Quantity
        {
            get { return quantity; }
            set { quantity = value; }
        }
        [MessageBodyMember]
        public string OrderNumber
        {
            get { return ordernumber; }
            set { ordernumber = value; }
        }

        public class ServiceClass : IServiceClass
        {
            string IServiceClass.InitiateOrder()
            {
                return "Initiating Order...";
            }
            public BookOrder PlaceOrder(BookOrder request)
            {
                BookOrder response = new BookOrder(request);
                response.OrderNumber = "12345678";
                return response;
            }
            string IServiceClass.FinalizeOrder()
            {
                return "Order placed successfully.";
            }
        }
    }
```

1.1.3 Typed message service (Bản tin định kiểu)

Trong một dịch vụ bản tin có định kiểu (typed message service) còn được gọi là "mô hình contract bản tin" bạn định nghĩa các bản tin và nội dung của chúng. Ở đây, các bản tin được mô tả với thuộc tính `MessageContract`, là các contract bản tin thường được sử dụng để định nghĩa các lớp bản tin. Các bản tin này sẽ được sử dụng để gửi qua lại trong các thao tác của dịch vụ.

```
    [ServiceContract]
```

```
public interface IBookOrder
{
    [OperationContract]
    void PlaceOrder(MyContract Contract);
}
[MessageContract]
public class MyContract
{
    [MessageHeader]
    string Title;
    [MessageBodyMember]
    decimal cost;
}
```

1.2 Các contract dịch vụ

Trong các bài trước các bạn đã biết cách để định nghĩa một contract dịch vụ, sử dụng thuộc tính `ServiceContract` trên một lớp hay một giao diện. Một dịch vụ trong WCF thông thường chứa một hay một vài các thao tác, các thao tác này được mô tả bởi lớp thuộc tính là `OperationContract`. Những phương thức của một lớp (giao diện) không có thuộc tính `OperationContract` thì không thuộc vào các thao tác của dịch vụ. Phía client sẽ không thể truy xuất tới các phương thức này. Điều này cũng giống như việc sử dụng thuộc tính `WebMethod` để định nghĩa các phương thức trong dịch vụ web.

1.3 Service Endpoints (Các điểm cuối dịch vụ)

Các client chỉ có thể truy nhập tới dịch vụ và sử dụng các thao tác của dịch vụ thông qua các điểm cuối dịch vụ. Điểm cuối dịch vụ có thể được định nghĩa thông qua mã nguồn hoặc thông qua tập tin cấu hình. Một dịch vụ có thể có một hay nhiều điểm cuối, mỗi điểm cuối có một địa chỉ (address), một binding, và một contract dịch vụ.

1.3.1 Định nghĩa điểm cuối bằng mã nguồn

Như đã nói ở trên, một điểm cuối dịch vụ cần có một địa chỉ, một binding, và một contract dịch vụ. Hai dòng đầu của đoạn mã nguồn bên dưới dùng để định nghĩa các địa chỉ mà dịch vụ sẽ sử dụng. Dòng tiếp tạo ra một đối tượng ServiceHost để chứa dịch vụ.

```
Uri bpa = new Uri("net.pipe://localhost/NetNamedPipeBinding");
Uri tcpa = new Uri("net.tcp://localhost:8000/TcpBinding");
ServiceHost sh = new ServiceHost(typeof(ServiceClass), bpa, tcpa);
NetNamedPipeBinding pb = new NetNamedPipeBinding();
```

```
    NetTcpBinding tcpb = new NetTcpBinding();
    ServiceMetadataBehavior behave = new ServiceMetadataBehavior();
    sh.Description.Behaviors.Add(behave);
    sh.AddServiceEndpoint(typeof(IMetadataExchange),
        MetadataExchangeBindings.CreateMexTcpBinding(), "mex");
    sh.AddServiceEndpoint(typeof(IMetadataExchange),
    MetadataExchangeBindings.CreateMexNamedPipeBinding(), "mex");
    sh.AddServiceEndpoint(typeof(IServiceClass), pb, bpa);
    sh.AddServiceEndpoint(typeof(IServiceClass), tcpb, tcpa);
    sh.Open();
```

1.3.2 Định nghĩa điểm cuối bằng tệp tin cấu hình

Sau đây là một ví dụ dùng tập tin cấu hình để định nghĩa điểm cuối.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="WCFService.Service" behaviorConfiguration="Service1Behavior">
        <host>
          <baseAddresses>
            <add baseAddress = "http://localhost:8731/Service/" />
            <add baseAddress = "net.tcp://localhost:8731/Service/" />
          </baseAddresses>
        </host>
        <endpoint address ="tcpmex" binding="mexTcpBinding"
contract="IMetadataExchange" />
        <endpoint address ="" binding="wsHttpBinding" contract="TempWCF.IService1"
/>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="Service1Behavior">
          <serviceMetadata httpGetEnabled="True"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
```

```
  </system.serviceModel>
</configuration>
```

2 Các phép hành xử dịch vụ (Service behaviors)

Contract dịch vụ định nghĩa các đầu vào, đầu ra, kiểu dữ liệu và các tính năng của một dịch vụ. Contract dịch vụ kết hợp với các thông tin về địa chỉ và binding làm cho dịch vụ có thể truy nhập được bởi các client. Tuy vậy với tất cả các thông tin và tính năng này, ta vẫn không điều khiển được các khía cạnh khác của dịch vụ như việc điều khiển thực thi của dịch vụ hay điều khiển luồng hoặc các thực thể của dịch vụ.

Câu trả lời ở đây là sử dụng các behavior (các hành xử). Các service behavior (hành xử dịch vụ) là các đối tượng dùng để thay đổi và điều khiển đặc tính của các dịch vụ Windows Communication Foundation trong khi thực thi. Khi cài đặt một contract dịch vụ, bạn có thể điều khiển nhiều đặc tính khi thực thi của dịch vụ.

Có hai kiểu behavior trong Windows Communication Foundation: service behavior và operation behavior (hành xử thao tác). Việc định nghĩa các behavior cũng giống như các đối tượng khác trong WCF bằng cách sử dụng các lớp thuộc tính. Ta sử dụng lớp thuộc tính `ServiceBehavior` và `OperationBehavior` để định nghĩa các behavior. Ví dụ sau đây sử dụng hai lớp thuộc tính này.

```
    [ServiceContract]
    public interface IServiceClass
    {
        [OperationContract]
        int Add(int x, int y);

        [OperationContract]
        int Subtract(int x, int y);
    }
    [ServiceBehavior]
    public class ServiceClass : IServiceClass
    {
        [OperationBehavior]
        public int Add(int x, int y)
        {
            return x + y;
        }

        [OperationBehavior]
```

```
        public int Subtract(int x, int y)
        {
            return x - y;
        }
    }
```

2.1 Lớp thuộc tính ServiceBehavior

Thuộc tính `ServiceBehavior` chỉ được áp dụng ở mức class. Tuy vậy nó có thể áp dụng cho cả các giao diện và các lớp. Sử dụng các tham số của lớp thuộc tính `ServiceBehavior` ta có thể điều khiển một số khía cạnh khác nhau của dịch vụ:

Tham số	Mô tả
Concurrency	Điều khiển xử lý luồng cho một đối tượng. Chỉ hợp lệ khi tham số `Instancing` có giá trị không phải là `PerCall`
Instancing	Điều khiển việc tạo mới đối tượng cũng như điều khiển thời gian sống của đối tượng đó. Giá trị mặc định là `PerCall`, nghĩa là mỗi lần phương thức được gọi thì tạo mới một đối tượng. Nói chung, trong các dịch vụ có hỗ trợ phiên làm việc, các giá trị `PerSession` hay `Shareable` có thể cho lại hiệu năng tốt hơn, tuy vậy lại tốn kém trong việc quản lý đồng bộ.
Throttling	Có thể thiết đặt thông qua tập tin cấu hình để giới hạn số lượng đồng thời các lời gọi, kết nối, tổng số thể hiện, và các thao tác đợi. Tham số này chỉ có tác dụng khi việc điều khiển luồng cho phép nhiều lời gọi đồng thời.
Transaction	Điều khiển các khía cạnh của một giao dịch như tự động hoàn thành, mức cách li, và việc sử dụng lại các đối tượng.
Quản lý phiên	Cung cấp khả năng tự động đóng một phiên hoặc nạp chồng các hành xử mặc định

Service behavior được cấu hình thông qua thành phần <serviceBehaviors> trong tập tin cấu hình. Ví dụ sau đây mô tả cách thiết lập behavior cho debug dịch vụ và siêu dữ liệu của dịch vụ:

```
    <behaviors>
      <serviceBehaviors>
        <behavior name="serviceBehavior">
          <serviceDebug includeExceptionDetailInFaults="true"/>
```

```
          <serviceMetadata />
        </behavior>
      </serviceBehaviors>
    </behaviors>
```

Để gắn một tập các behavior tới một dịch vụ, sử dụng thuộc tính behaviorConfiguration của thành phần <service> như ví dụ sau:

```
    <services>
      <service behaviorConfiguration="TempWCFServiceBehavior"
        name="TempWCF.ServiceClass">
       ...
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="TempWCFServiceBehavior">
          <serviceMetadata httpGetEnabled="true" />
          <serviceDebug includeExceptionDetailInFaults="false" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
```

Ngoài việc sử dụng tập tin cấu hình, bạn còn có thể sử dụng mã nguồn để định nghĩa các service behavior:

```
ServiceHost host = new ServiceHost(typeof(ServiceClass));

ServiceMetadataBehavior mexBehavior =
host.Description.Behaviors.Find<ServiceMetadataBehavior>();

if (mexBehavior == null)
{
  mexBehavior = new ServiceMetadataBehavior();
  mexBehavior.HttpGetEnabled = true;
  host.Description.Behaviors.Add(mexBehavior);
}

host.Open();
```

2.2 Lớp thuộc tính OperationBehavior

Sử dụng `ServiceBehavior` bạn có thể điều khiển các behavior ở mức dịch vụ, tuy nhiên bạn còn có thể điều khiển các behavior ở mức các thao tác của dịch vụ sử dụng lớp thuộc tính `OperationBehavior`. Sử dụng lớp thuộc tính này bạn có thể điều khiển các khía cạnh sau:

Điều khiển	Mô tả
Transaction	Cung cấp khả năng tự động hoàn thành với luồng giao dịch và các tuỳ chọn cần thiết
Caller Identity	Khi được hỗ trợ bởi binding, sẽ cung cấp khả năng thực thi dưới định danh của hàm gọi (caller)
Việc sử dụng lại đối tượng	Cung cấp khả năng nạp chồng chế độ InstanceMode của ServiceBehavior

3 Xử lý lỗi

Windows Communication Foundation sử dụng hai hệ thống báo lỗi là các đối tượng Exception và các bản tin lỗi SOAP. Trong các ứng dụng trên nền .NET Framework thì lỗi được biểu diễn dưới dạng các đối tượng Exception hoặc lớp kế thừa của Exception. Trong các ứng dụng SOAP, thông tin về lỗi được liên lạc bằng các bản tin lỗi SOAP. Do vậy, các lỗi (exception) ở trong WCF cần phải được chuyển về dạng bản tin lỗi SOAP mới có thể gửi tới client.

Có hai kiểu lỗi SOAP có thể gửi trả lại phía client như sau:

- **Được khai báo**: Những thao tác có khai báo lỗi sẽ được mô tả với thuộc tính `FaultContract`. Việc mô tả bằng thuộc tính mô tả `FaultContract` nhằm mục đính đặc tả kiểu của lỗi SOAP.
- **Không khai báo:** Lỗi dạng này không được khai báo ở các thao tác trong dịch vụ

Khi xây dựng dịch vụ, ta nên khai báo các lỗi có thể xảy ra ở một thao tác sử dụng thuộc tính `FaultContract`. Điều này cho phép đặc tả một cách chính thức tất cả các lỗi SOAP mà client có thể nhận được khi sử dụng thao tác này. Để đảm bảo tính bảo mật, chỉ những thông tin mà client cần biết liên quan tới lỗi thì mới nên trả về trong lỗi SOAP.

3.1 Ví dụ xử lý lỗi sử dụng FaultContract

Bước 1: Tạo một bản tin lỗi theo ý.

```
namespace FaultContractSample
{
    using System.Runtime.Serialization;

    /// <summary>
    /// Đây là cấu trúc được sử dụng để định nghĩa một bản tin báo lỗi.
    /// Lỗi chi tiết có thể xem ở thuộc tính Message của cấu trúc này.
    /// </summary>
    [DataContract]
    public struct CustomFaultMsg
    {
        /// <summary>
        /// Thuộc tính này được sử dụng để chứa thông tin chi tiết về lỗi xảy
ra.
        /// Thông tin về lỗi sẽ được gửi từ dịch vụ tới client
        /// </summary>
        [DataMember]
        public string Message { get; set; }
    }
}
```

Bước 2: Gắn lỗi SOAP với một thao tác trong dịch vụ

```
using System.Runtime.Serialization;
using System.ServiceModel;

/// <summary>
/// Định nghĩa thông tin về một nhân viên
/// </summary>
[DataContract]
public class Person
{
    [DataMember]
    public int PersonId;

    [DataMember]
    public string FullName;
    [DataMember]
    public int Age;
}

[ServiceContract]
```

```
public interface IStaffInformation
{
    [OperationContract]
    bool HasPerson(int personId);

    /// <summary>
    /// Lấy về thông tin của một nhân viên.
    /// Nếu có nhân viên với ID trùng với personId thì
    /// trả về thông tin nhân viên đó.
    /// Trong trường hợp không có nhân viên phù hợp thì báo lỗi bằng bản tin
lỗi
    /// </summary>
    /// <param name="personId">ID của nhân viên</param>
    /// <returns>Thông tin của nhân viên</returns>
    [OperationContract]
    [FaultContract(typeof(CustomFaultMsg))]
    Person GetPerson(int personId);

    [OperationContract]
    Person[] GetAll();

    [OperationContract]
    void AddPerson(Person person);

    /// <summary>
    /// Sửa thông tin của một nhân viên.
    /// Nếu có nhân viên với ID trùng với personId thì
    /// cập nhật thông tin nhân viên đó.
    /// Trong trường hợp không có nhân viên phù hợp thì báo lỗi bằng bản tin
lỗi
    /// </summary>
    /// <param name="personId">ID của nhân viên</param>
    /// <param name="person">Thông tin mới về nhân viên</param>
    [OperationContract]
    [FaultContract(typeof(CustomFaultMsg))]
    void EditPerson(int personId, Person person);

    /// <summary>
    /// Xoá thông tin một nhân viên
    /// Nếu có nhân viên với ID trùng với personId thì xoá thông tin nhân
viên đó.
```

```
    /// Trong trường hợp không có nhân viên phù hợp thì báo lỗi bằng bản tin
lỗi
    /// </summary>
    /// <param name="personId">ID của nhân viên</param>
    [OperationContract]
    [FaultContract(typeof(CustomFaultMsg))]
    void DeletePerson(int personId);
}
```

Bước 3: Kiểm tra điều kiện, báo lỗi khi cần:

```
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;

public class StaffInformation : IStaffInformation
{
    readonly List<Person> personCollection = new List<Person>();

    public bool HasPerson(int personId)
    {
        return personCollection.Any(x => x.PersonId == personId);
    }

    public Person GetPerson(int personId)
    {
        if (!HasPerson(personId))
        {
            var error = new CustomFaultMsg
            { Message = "Không có nhân viên nào với ID là : " + personId };
            throw new FaultException<CustomFaultMsg>(error);
        }

        return personCollection.FirstOrDefault(x => x.PersonId == personId);
    }

    public void AddPerson(Person person)
    {
        personCollection.Add(person);
    }

    public Person[] GetAll()
```

```
    {
        return personCollection.ToArray();
    }

    public void EditPerson(int personId, Person person)
    {
        Person p = GetPerson(personId);
        p.FullName = person.FullName;
        p.Age = person.Age;
    }

    public void DeletePerson(int personId)
    {
        Person p = GetPerson(personId);
        personCollection.Remove(p);
    }
}
```

Bước 4: Xử lý lỗi phía client

```
    private void buttonGetInformation_Click(object sender, EventArgs e)
    {
        var client = new StaffInformationClient();
        try
        {
            Person p = client.GetPerson(Convert.ToInt32(textBoxID.Text));
            dataGrid.DataSource = new[] { p };
        }
        catch (FaultException<CustomFaultMsg> ex)
        {
            MessageBox.Show(ex.Message, "Error");
        }
    }
```

Sau đây là một số kết quả khi thực hiện:

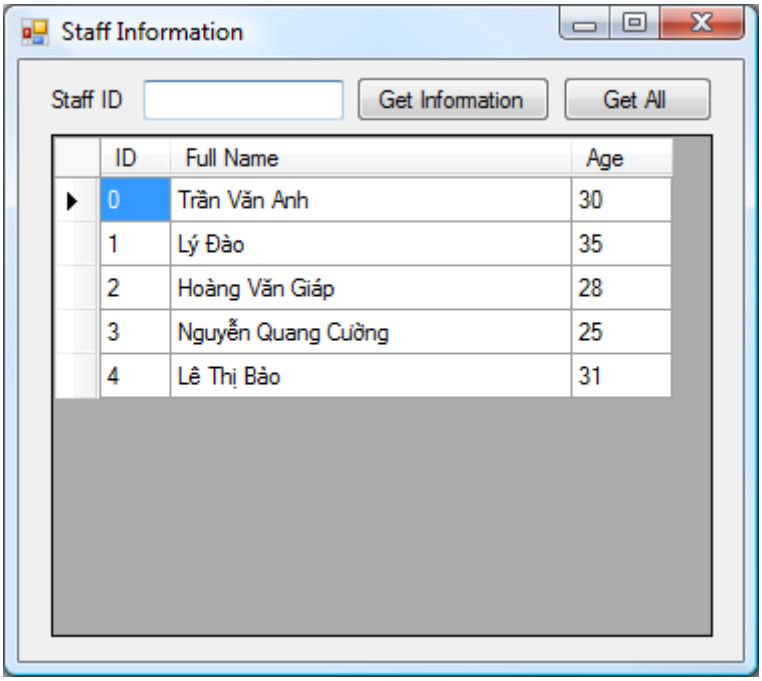

Figure 1 Những nhân viên có trong cơ sở dữ liệu

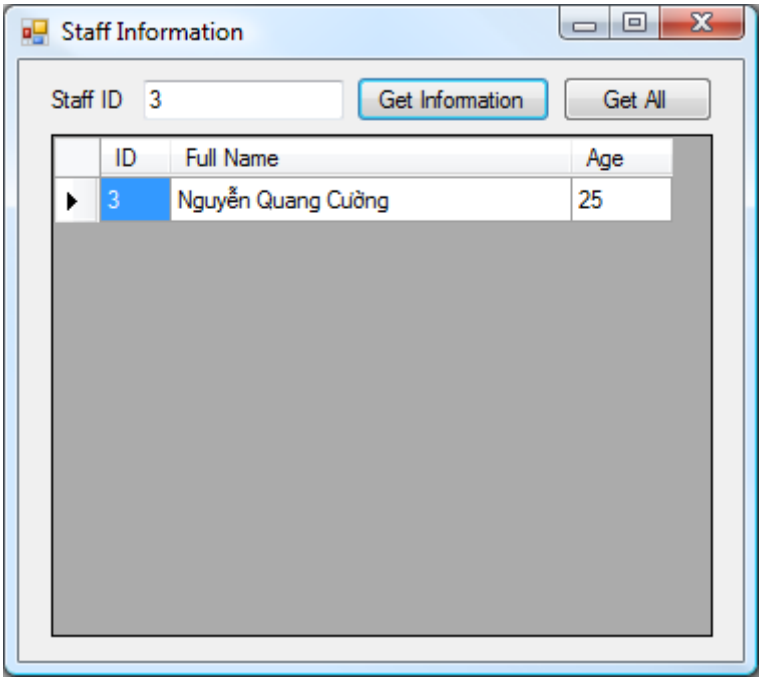

Figure 2 Thông tin về nhân viên có ID là 3

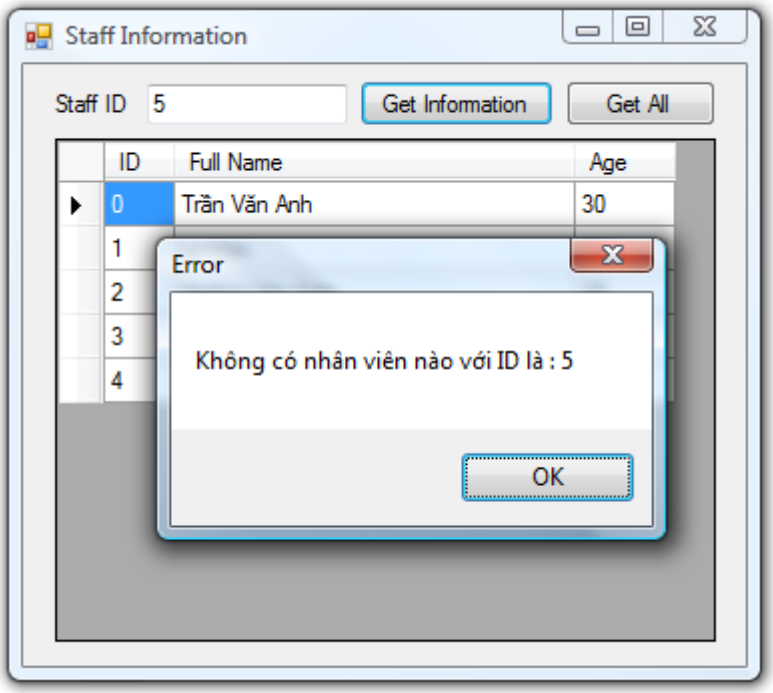

Figure 3 Báo lỗi khi lấy thông tin về nhân viên ID là 5

4 Câu hỏi ôn tập

1. Làm thế nào để dịch vụ của WCF báo cho client biết là có lỗi nào đó xảy ra trong quá trình thực hiện?

Trả lời: Có nhiều cách để báo cho client biết về lỗi xảy ra, thông thường ta sử dụng cách báo lỗi bằng lớp thuộc tính mô tả FaultContract cho một thao tác có lỗi, đồng thời trong thao tác đó throw một đối tượng Exception.

5 Tài liệu tham khảo

1. Discover Mighty Instance Management Techniques For Developing WCF Apps (URL: http://msdn.microsoft.com/en-us/magazine/cc163590.aspx)
2. ServiceBehaviorAttribute Class (URL: http://msdn.microsoft.com/en-us/library/system.servicemodel.servicebehaviorattribute.aspx)
3. Using the Fault Contracts (SOAP Faults) in WCF Programming (URL: http://www.c-sharpcorner.com/UploadFile/SunilBabuYLV/SOAPFaultContract08262008093633AM/SOAPFaultContract.aspx)