Bài 4
Giới thiệu tổng quan về WCF

MỤC TIÊU

- Giới thiệu WCF
- Công cụ phát triển WCF
- Các tính năng của WCF
- Khác biệt giữa WCF và Web Service ASP.NET
- Kiến trúc của WCF
- Tìm hiểu về các contract
- Mô hình giao tiếp ABC trong WCF
- Tạo dự án WCF đầu tiên

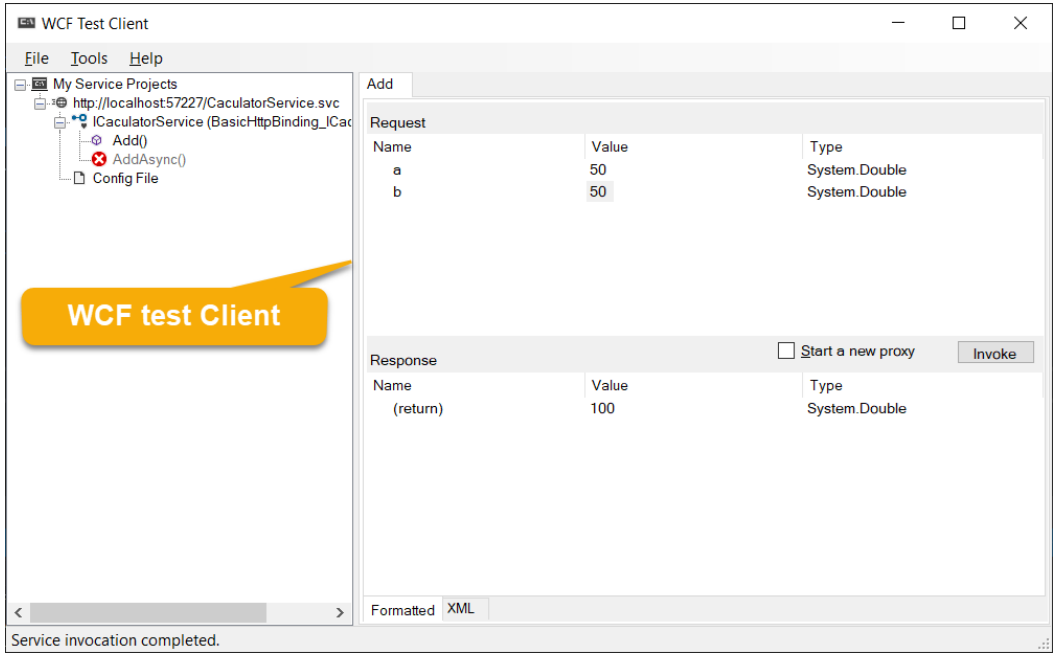
Giới thiệu WCF

WCF là mô hình phát triển ứng dụng hướng dịch vụ (SOA) trên nền tảng của Microsoft, có khả năng thích ứng cao với những thay đổi thực tế của doanh nghiệp. WCF (Windows Communication Foundations) kết hợp các đặc điểm từ công nghệ phân tán của ASP.NET Web Services, .NET Remoting, Message Queuing và Enterprise Services cho phép xây dựng ứng dụng linh động, tin cậy, an toàn, bảo mật cao.

Được giới thiệu từ năm 2006 với phiên bản đầu tiên là 3.0, đến nay WCF đã không ngừng được Microsoft cải tiến, nâng cấp và phiên bản chính thức mới nhất là WCF 4.5. Trên phiên bản này, Microsoft đã đưa thêm nhiều tính năng hỗ trợ giúp cho việc lập trình mô hình WCF dễ dàng và chặt chẽ hơn.

Công cụ phát triển WCF

- Visual Studio 2017 hoặc cao hơn sử dụng để soạn thảo code
- WCF test Client (Có sẵn khi cài Visual Studio) sử dụng để test nhanh dịch vụ đã tạo



Các tính năng của WCF 1/2

Transaction (Giao dịch):

- Một giao dịch là một đơn vị của công việc. Một giao dịch đảm bảo chắc chắn rằng mọi thứ diễn ra trong giao dịch thành công hay thất bại đều là kết quả tổng thể.
- WCF cho phép đưa vào việc xử lý giao dịch như trên với các liên lạc. Nhà phát triển có thể nhóm các liên lạc với nhau thành các giao dịch.
- Ở mức doanh nghiệp, tính năng này cho phép bạn thực hiện các công việc giao dịch qua các nền tảng khác nhau.

Host linh động:

- Dịch vụ WCF có thể được hosting lên nhiều môi trường khác nhau, bao gồm IIS, Windows service, Self-hosting...

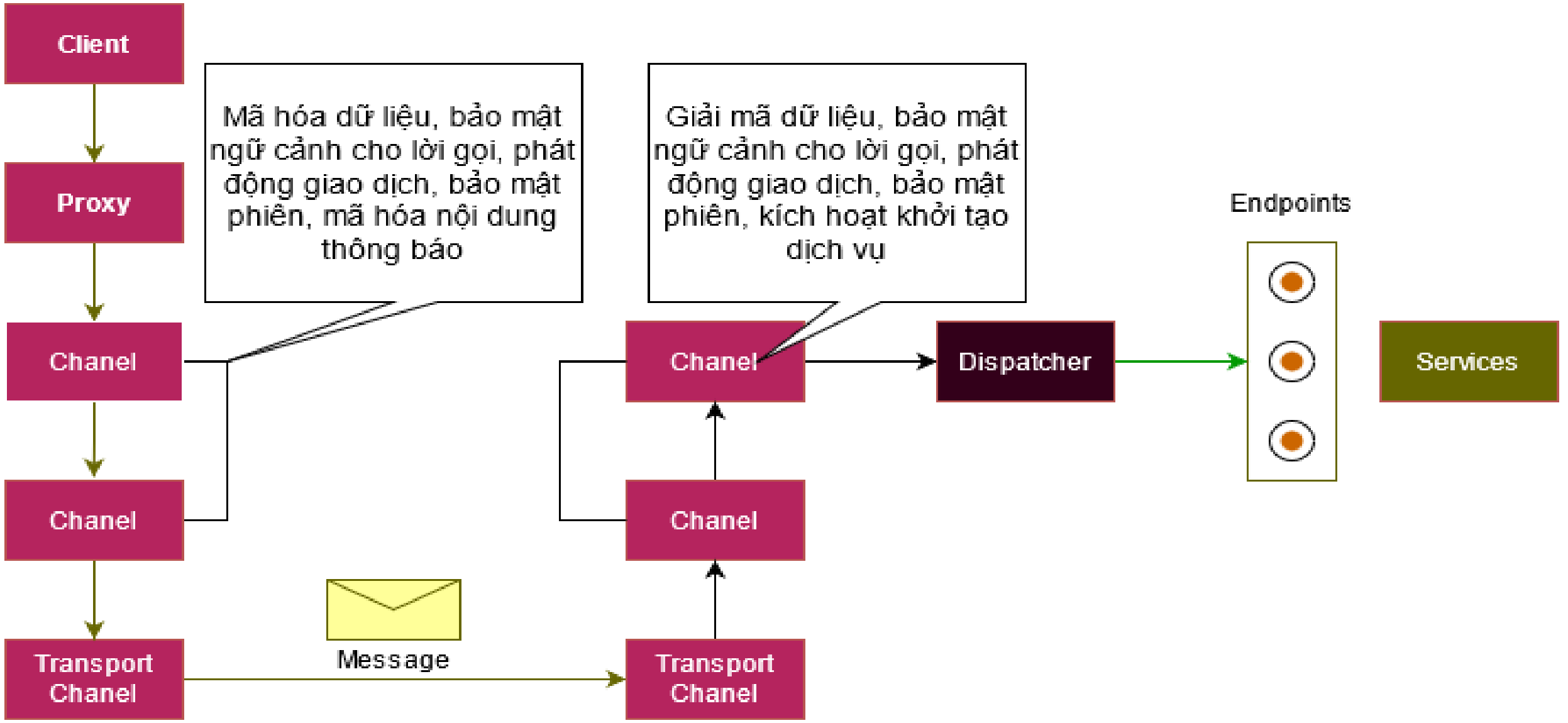
Bảo mật

- Trong WCF, tất cả mọi thứ từ các bản tin tới các client hay server đều phải xác thực và WCF có tính năng để đảm bảo rằng các bản tin không bị lẫn trong quá trình vận chuyển.
- WCF bao gồm việc đảm bảo tính toàn vẹn và bảo mật của bản tin.
- WCF còn cho phép bạn tích hợp ứng dụng của bạn với cơ sở hạ tầng bảo mật sẵn có, bao gồm cả các chuẩn bên ngoài môi trường Windows bằng cách sử dụng các bản tin SOAP bảo mật.

Khác biệt giữa WCF và Web Service ASP.NET

WCF Service	ASP.NET Web Service
Hỗ trợ được tất cả giao thức: HTTP, HTTPs, WSHTTP, TCP, MSMQ	Chỉ hỗ trợ giao thức HTTP, HTTPs
Hỗ trợ đảm bảo giao tác an toàn - Atomic Transactions	Không hỗ trợ đảm bảo giao tác an toàn - Atomic Transactions
Mặc định WCF sử dụng chuẩn SOAP để gửi và nhận thông điệp, nhưng WCF có thể hỗ trợ nhiều định dạng thông điệp khác nhau như: binary, MTOM (Message Transfer Optimized Mechanism),…	Chỉ có thể gửi và nhận thông điệp dạng chuẩn SOAP.
Thông qua DataContractSerializer, có thể xác định thuộc tính nào, thể hiện nào cần chuyển đổi sang XML	Sử dụng XmlSerializer để chuyển đổi với nhiều hạn chế như chỉ có thuộc tính kiểu Public hay class dẫn xuất từ IEnumerablemới chuyển được.

Kiến trúc của WCF



Các Contracts

Các Contracts rất hữu ích cho việc xây dựng các ứng dụng dịch vụ WCF. Nó xác định giao thức (ràng buộc) mà dịch vụ sử dụng, giao tiếp sẽ được thực hiện như thế nào, định dạng trao đổi thông điệp sẽ sử dụng, v.v.

Sau đây là các hợp đồng có sẵn trong WCF:

- Service contracts
- Data Contracts
- Message contracts
- Fault Contract
- Operation Contract

Service contract

Service contracts xác định Giao diện cho dịch vụ.

Nó có thể được định nghĩa như sau.

Syntax

```
01. [ServiceContract]
02. public interface IService1
03. {
04.
05.     // TODO: Add your service operations here
06. }
```

- Một dịch vụ WCF có thể có nhiều hơn một service contract.
- Một dịch vụ WCF cần Khai báo ít nhất một service contract.
- Thuộc tính [ServiceContract] cần được sử dụng khi khai báo service contract.
- Nó cho phép xác định một service contract hoạt động theo nó để đưa dịch vụ ra bên ngoài.
- Nó ánh xạ interface and methods dịch vụ WCF tới một mô tả độc lập với nền tảng.

Operation Contract

Operation Contract xác định phương thức triển khai tới máy khách để trao đổi thông tin giữa máy khách và máy chủ. Operation Contract mô tả chức năng nào sẽ được cung cấp cho khách hàng, chẳng hạn như cộng, trừ, v.v.

Syntax

```
01. public interface IService1
02. {
03.     [OperationContract]
04.     string GetData(int value);
05. 
06.     [OperationContract]
07.     CompositeType GetDataUsingDataContract(CompositeType composite);
08. 
09. }
```

Data Contract

Data Contract xác định kiểu dữ liệu cho các biến giống như thuộc tính get và set nhưng sự khác biệt là Data Contract trong WCF được sử dụng để tuần tự hóa và giải mã hóa dữ liệu phức tạp. Nó xác định cách các kiểu dữ liệu được tuần tự hóa và giải mã hóa.

Sử dụng tuần tự hóa, bạn có thể chuyển đổi một đối tượng thành một chuỗi byte có thể được truyền qua mạng.

Syntax

```
01. [DataContract]
02. public class Student
03. {
04.     private string _Name;
05.
06.     private string _City;
07.
08.
09.     [DataMember]
10.     public string Name
11.     {
12.         get { return _Name; }
13.         set { _Name = value; }
14.     }
15. }
```

Fault Contract 1/2

Trong một ứng dụng Windows điển hình, chúng tôi sử dụng khối try..catch...finally để xử lý và bắt các ngoại lệ.

Nhưng khối try..catch không hoạt động trong các dịch vụ WCF, trừ khi chúng ta ném và tạo ra ngoại lệ.

Trong WCF, các lỗi có thể được xử lý và thông báo lỗi có thể được chuyển đến các ứng dụng khách bằng SOAP Fault Contract.

Fault Contract có thể cung cấp chế độ xem lỗi dạng văn bản cho ứng dụng máy khách, điều này giúp cho việc kiểm tra và thu thập thông tin lỗi dex dàng hơn

Fault Contract 2/2

Ví dụ muốn tạo một customize lỗi

```
01. [DataContract()]
02. public class CustomException {
03.  [DataMember()]
04.  public string Title;
05.  [DataMember()]
06.  public string ExceptionMessage;
07.  [DataMember()]
08.  public string InnerException;
09.  [DataMember()]
10.  public string StackTrace;
11. }
```

```
01. [ServiceContract()]
02. public interface ISimpleCalculator {
03.  [OperationContract()]
04.  [FaultContract(typeof(CustomException))]
05.  int Add(int num1, int num2);
06. }
```

```
01. public int Add(int num1, int num2) {
02.  //Do something
03.  CustomException ex = new CustomException();
04.  ex.Title = "Error Funtion:Add()";
05.  ex.ExceptionMessage = "Error occur while doing add function.";
06.  ex.InnerException = "Inner exception message from serice";
07.  ex.StackTrace = "Stack Trace message from service.";
08.  throw new FaultException(ex, "Reason: Testing the Fault contract");
09. }
```

```
01. try {
02.  MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy = new MyCalculatorServiceProx
03.  Console.WriteLine("Client is running at " + DateTime.Now.ToString());
04.  Console.WriteLine("Sum of two numbers... 5+5 =" + proxy.Add(5, 5));
05.  Console.ReadLine();
06. } catch (FaultException < MyCalculatorService.CustomException > ex) {
07.  //Process the Exception
08. }
```

Client

Message contract 1/2

Thông thường để gửi dữ liệu qua SOAP, chúng ta cần kiểm soát dữ liệu được gửi đi, chẳng hạn như chúng ta có thể cần gửi dữ liệu trong header hoặc body của SOAP. Trong trường hợp đó, sử dụng DataContract và DataMember không phải là giải pháp.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns="h
    none">http://tempuri.org/IEmpService/
  </s:Header>
  <s:Body>
    <GetData xmlns="http://tempuri.org/">
      <requestData xmlns:d4p1="http://sch
      xmlns:i="http://www.w3.org/2001/XML

        <d4p1:EmpId>12</d4p1:EmpId>

      </requestData>
    </GetData>
  </s:Body>
</s:Envelope>
```

```
namespace WcfServiceDemo
{
    [DataContract]
    public class ResponseData

    [DataContract]
    public class RequestData
    {
        [DataMember]
        public Int32 EmpId { get; set; }
    }
}
```

DataContract

Request tham số cách thông thường

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <GetDataResponse xmlns="http://tempuri.org/">
      <GetDataResult xmlns:a="http://schemas.datacontract.org/2004/07/WcfServiceDemo"
      xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:EmpEmail>hunghv@gmail.com</a:EmpEmail>
        <a:EmpId>12</a:EmpId>
        <a:EmpName>Hoàng Văn Hưng</a:EmpName>
        <a:EmpPhone>0985421127</a:EmpPhone>
      </GetDataResult>
    </GetDataResponse>
  </s:Body>
</s:Envelope>
```

Response dạng DataContact

Message contract 2/2

Message contract giúp chúng ta kiển soát được dữ liệu request, response ở header hay body

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns=
    none">http://tempuri.org/IEmpService/GetData</Action>

    <h:EmpId xmlns:h="http://tempuri.org/">25</h:EmpId>

  </s:Header>
  <s:Body>
    <RequestData xmlns="http://
  </s:Body>
</s:Envelope>
```

MessageContract

```
amespace WcfServiceDemo

[MessageContract]
public class ResponseData...

[MessageContract]
public class RequestData
{
    [MessageHeader]
    public Int32 EmpId { get; set; }
}
```

Request trên Header

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <ResponseData xmlns="http://tempuri.org/">
      <EmpEmail>hunghv@gmail.com</EmpEmail>
      <EmpId>25</EmpId>
      <EmpName>Hoàng Văn Hưng</EmpName>
      <EmpPhone>0985421127</EmpPhone>
    </ResponseData>
  </s:Body>
</s:Envelope>
```

Response

Mô hình giao tiếp trong WCF

WCF tuân theo kiến trúc Client – Server. Trong đó giao tiếp giữa Client và Server được thiết lập bằng cách sử dụng các điểm kết nối (Endpoints) do WCF Service cung cấp.

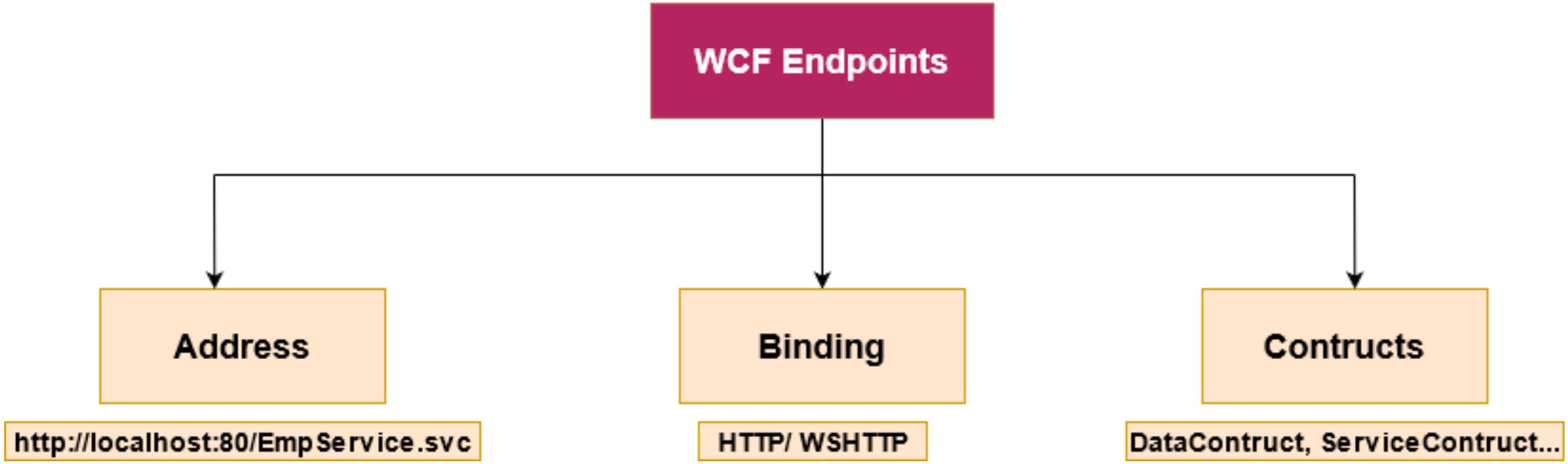
Một service có thể chấp nhận và xử lý nhiều yêu cầu khác nhau thông các endpoint riêng biệt.

Mỗi endpoint được tạo thành từ ba thành phần với tên gọi tắt rất dễ nhớ là ABC, trong đó:

- **A – Address (Where)**: địa chỉ của service.
- **B – Binding (How):** Cách thức giao tiếp với service. Thành phần này xác định loại giao thức kết nối giữa client và service (như HTTP, TCP, MSMQ,…), kênh xử lý và kiểu mã hóa thông điệp.
- **C – Contact (What):** Thông tin mô tả các chức năng của service. Cụ thể, đây là các class được định nghĩa bên service để với các phương thức mà client có thể yêu cầu service thực hiện.

WCF Endpoints

- Endpoints là sự kết nối của 3 thành phần gọi tắt là ABC, đó là
- A - Address, B - Binding và C - Contracts



WCF Endpoints (1/2)

Tạo một ứng dụng **Console App**, Add Reference Wcf Service và cấu hình endpoint trong file **App.config** gồm các thành phần mã có dạng sau

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <behaviors>
            <serviceBehaviors>
                <behavior name="WcfServiceDemo">
                    <serviceMetadata httpGetEnabled="true"/>
                </behavior>
            </serviceBehaviors>
        </behaviors>
        <services>
            <service name="WcfServiceDemo.CaculatorService" behaviorConfiguration="WcfServiceDemo">
                <host>
                    <baseAddresses>
                        <add baseAddress="http://localhost:8000/"/>
                    </baseAddresses>
                </host>
                <endpoint address="CaculatorService" binding="basicHttpBinding" contract="WcfServiceDemo.ICaculatorService" />
            </service>
        </services>
    </system.serviceModel>
</configuration>
```

behaviors 1

host 2

endpoint 3

Code C#

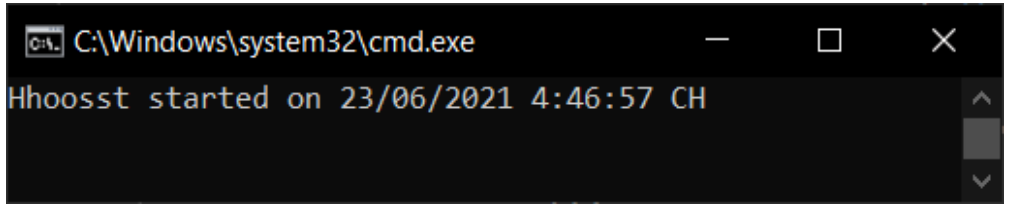
```csharp
using System;
using System.ServiceModel;

namespace WcfServiceHost
{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceHost host = new ServiceHost(typeof(WcfServiceDemo.CaculatorService));

            host.Open();
            Console.WriteLine("Hhoosst started on " + DateTime.Now.ToString());
            Console.ReadLine();
        }
    }
}
```

WCF Endpoints (2/3)

Khởi động dự án Console App đã cấu hình màn hình cmd hiện lên thông báo

Server đã khởi động theo link

http://localhost:8000



CaculatorService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost:8000/?wsdl
```

You can also access the service description as a single file:

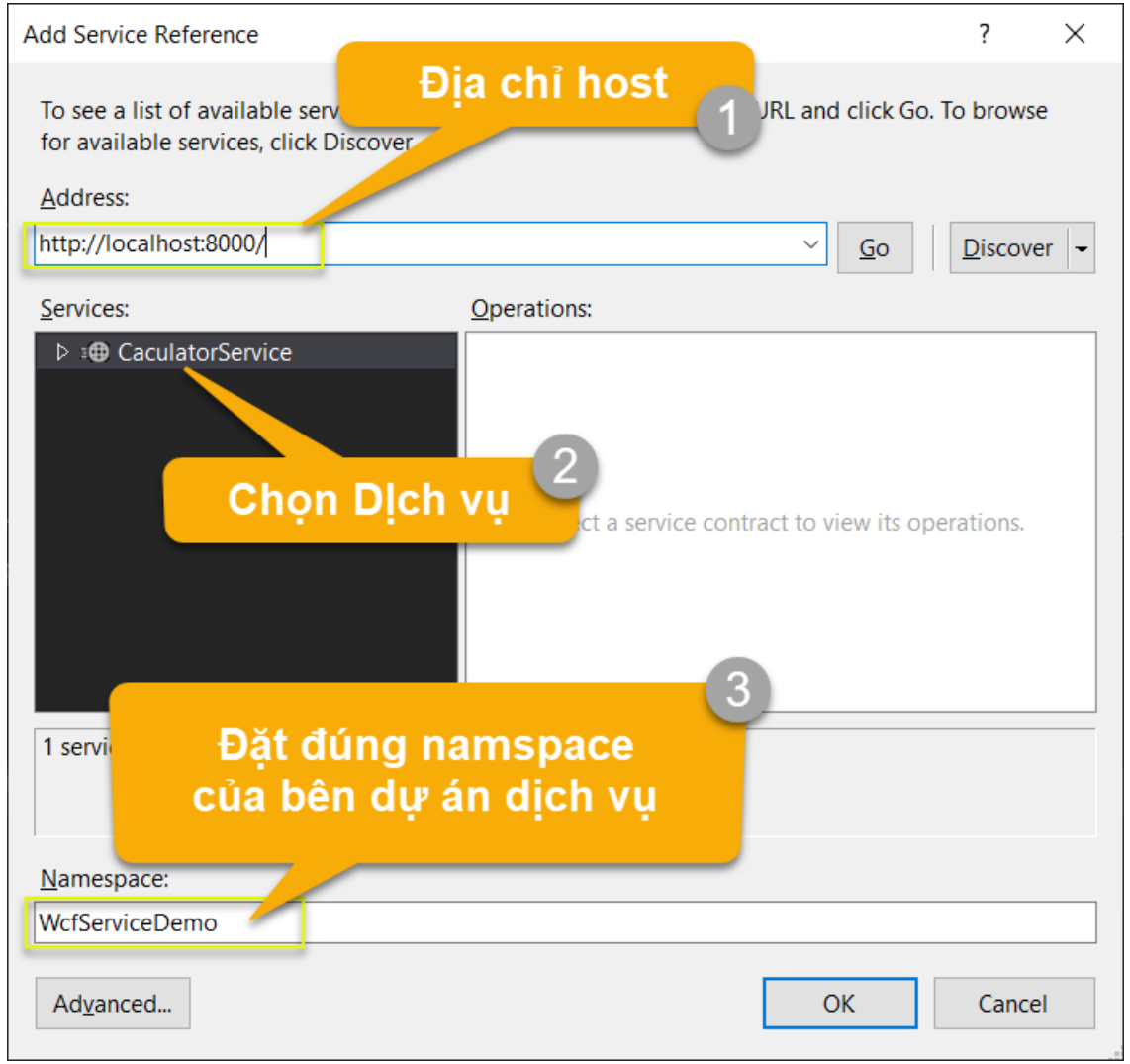
```
http://localhost:8000/?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        CaculatorServiceClient client = new CaculatorServiceClient();
```

WCF Endpoints (3/3)

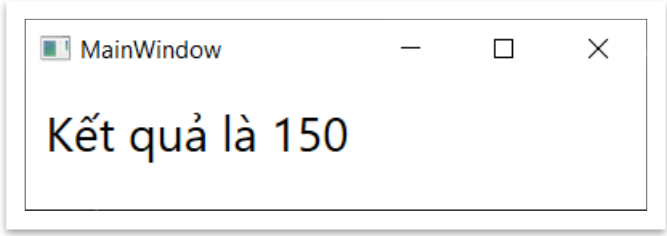


Tạo dự án Client VD WPF App và Add Service Reference

Tìm đến link host đã cấu hình và khởi động

Chọn dịch vụ cần add vào

Nhớ đặt tên namespace đúng như bên dự án service

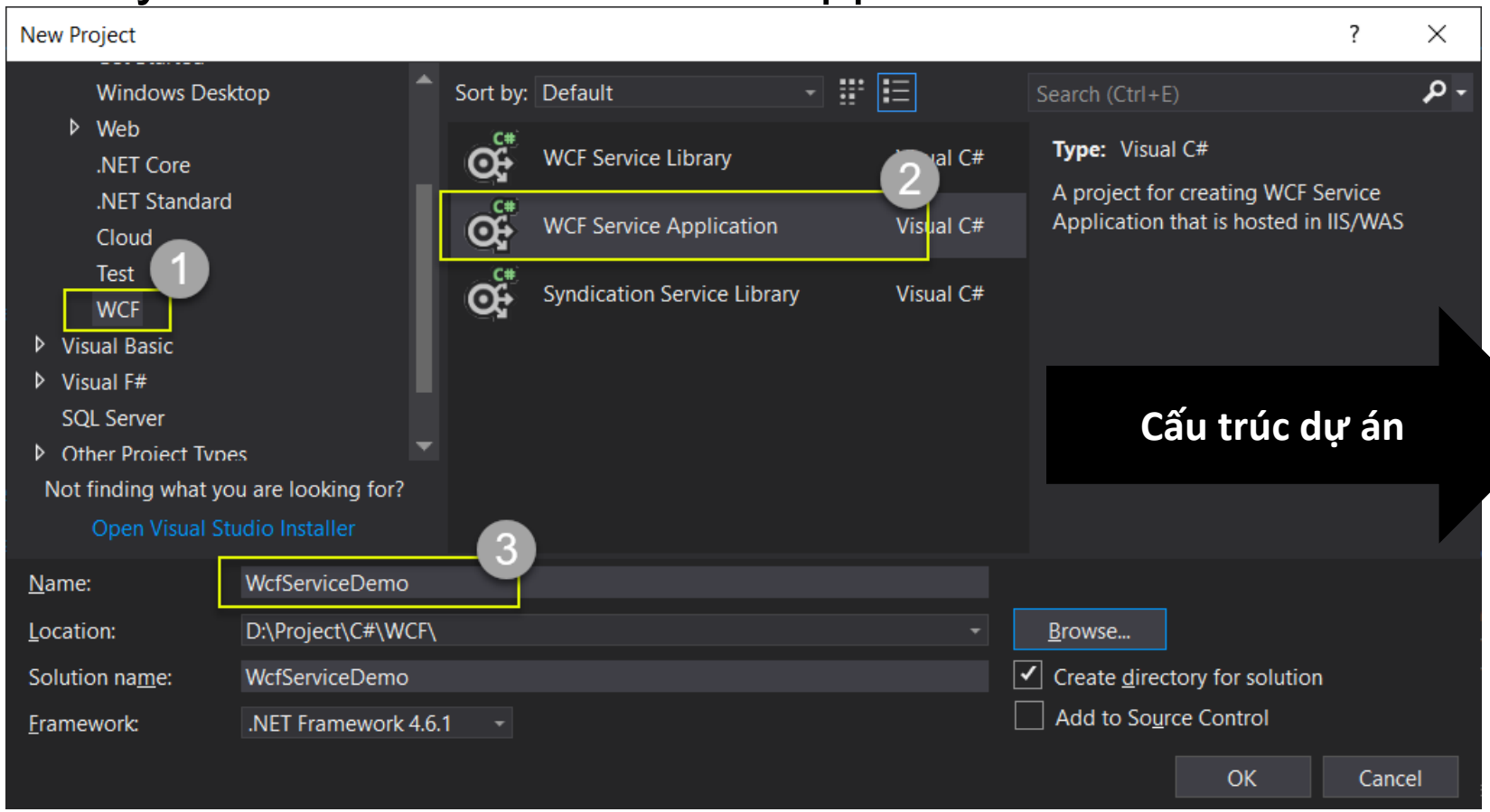


Code C# tại client

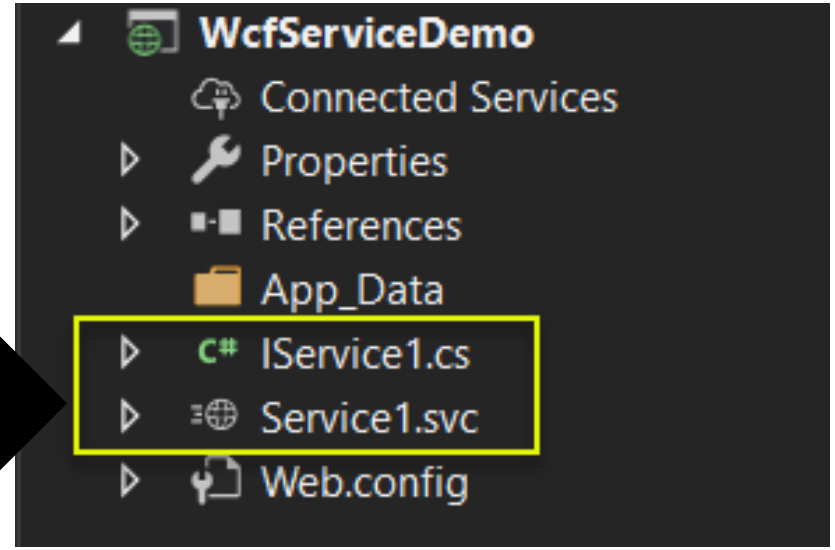
```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        WcfServiceDemo.CaculatorServiceClient client =
            new WcfServiceDemo.CaculatorServiceClient("BasicHttpBinding_ICaculatorService");
        TxtResault.Text = "Kết quả là " + client.Add(50, 100).ToString();
    }
}
```

Tạo dự án WCF đầu tiên

- Mở Visual Studio và tạo một project với tên WcfServiceDemo
- Chú ý: Tìm đến WCF Service Application như hình dưới



Cấu trúc dự án

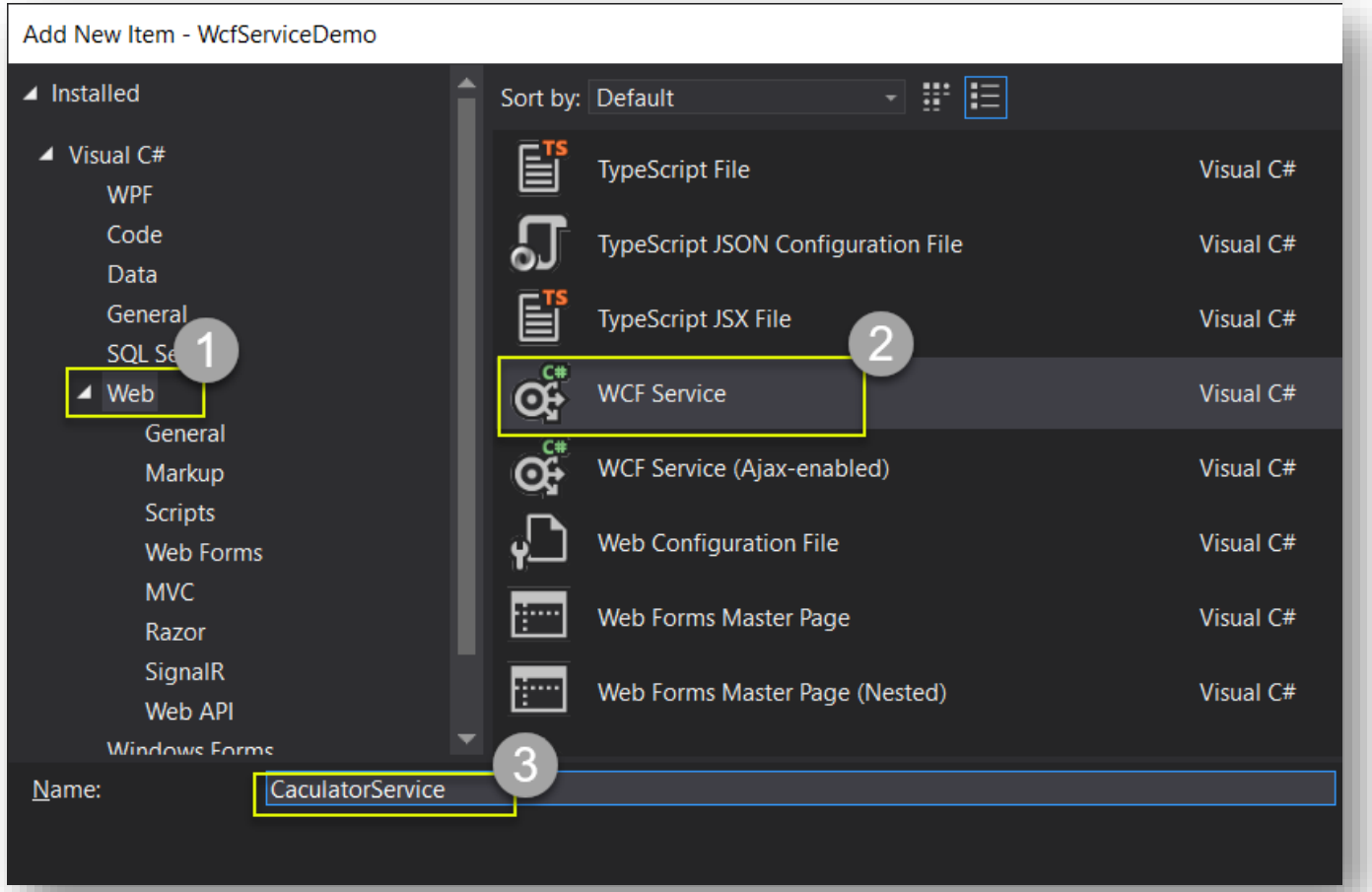


Mỗi một Service thường có 2 phần

Interface và code logic cho service

Tạo dự án WCF đầu tiên 1/3

- Xóa file Iservice1.cs và Service1.svc và tạo một WCF service mới đặt tên là **CaculatorService**
- Click phải chuột lên thư mục gốc, chọn **Add -> New Item** tìm đến **WCF Service**




```
using System.ServiceModel;
namespace WcfServiceDemo
{
    [ServiceContract]
    public interface ICaculatorService
    {
        [OperationContract]
        double Add (double a, double b);
    }
}
```

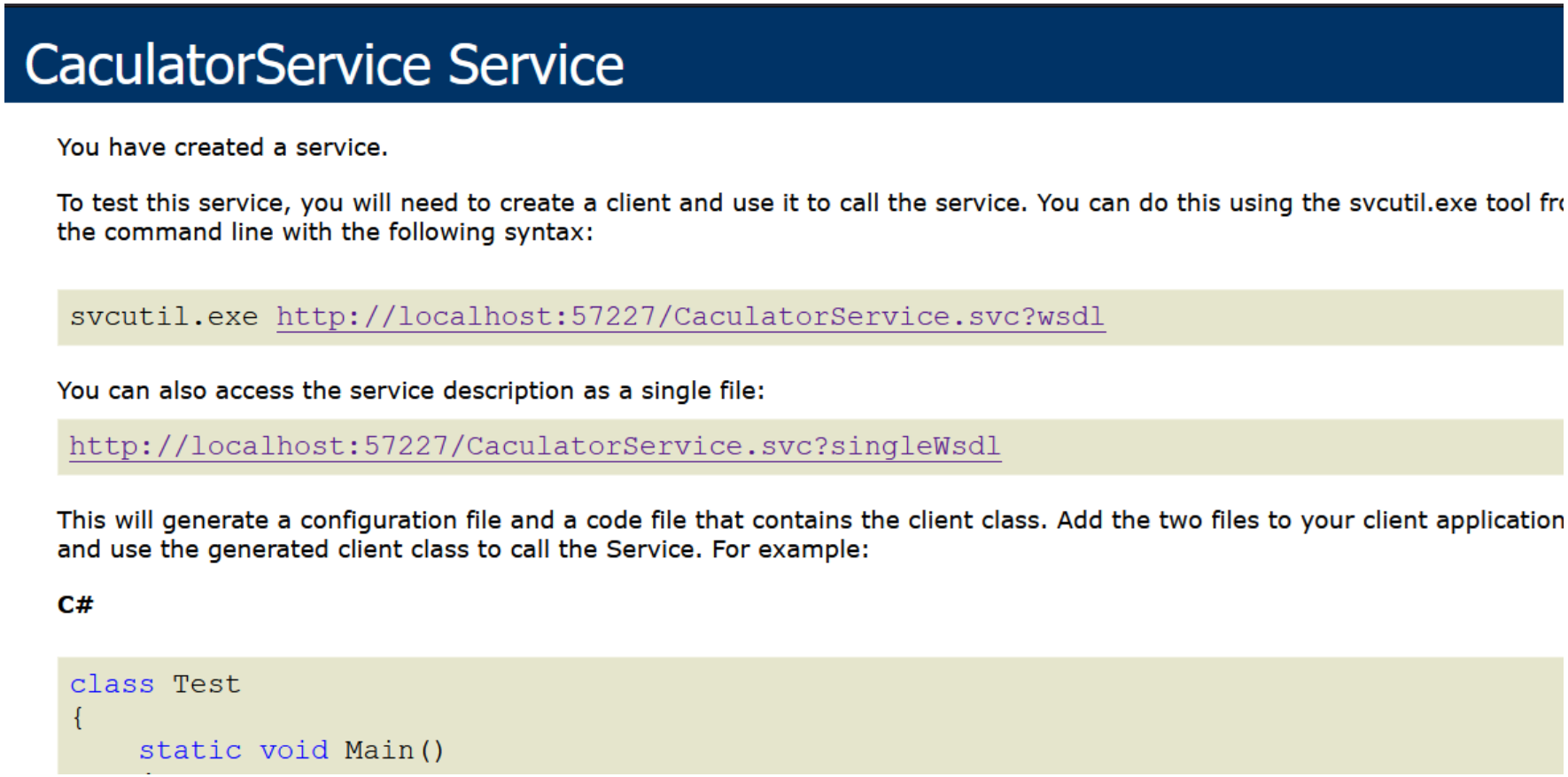
interface

```
namespace WcfServiceDemo
{
    public class CaculatorService : ICaculatorService
    {
        public double Add(double a, double b)
        {
            return a + b;
        }
    }
}
```

class

Tạo dự án WCF đầu tiên 2/3

- Test dịch vụ bằng cách, build lại dự án, click phải chuột vào file CaculatorService.svc chọn View in Browser, dịch vụ sẽ được kích hoạt và mở thông báo trên trình duyệt có dạng http://localhost:57227/CaculatorService.svc



CaculatorService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool fr the command line with the following syntax:

```
svcutil.exe http://localhost:57227/CaculatorService.svc?wsdl
```

You can also access the service description as a single file:

```
http://localhost:57227/CaculatorService.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
```

Tạo dự án WCF đầu tiên 3/3

- Quay lại code của dự án, nhân CTRL + F5 để khởi động WCF test client

Tạo ứng dụng đầu tiên

HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH

TRÂN TRỌNG CẢM ƠN!

238 Hoàng Quốc Việt, Bắc Từ Liêm, Hà Nội

0968.27.6996

tuyensinh@bachkhoa-aptech.edu.vn

www.bachkhoa-aptech.edu.vn