WCF Tutorial

2013-05-04

Sources:

<http://msdn.microsoft.com/en-us/library/ms734712.aspx>

<http://wcftutorial.net/>

# Generalities

## Definitions

A WCF service exposes one or more endpoints.

Each endpoint exposes one or more service operations.

The endpoint of a service specifies an **address** [where?] where the service can be found, a **binding** [how?] that contains the information that describes how a client must communicate with the service, and a **contract** [what?] that defines the functionality provided by the service to its clients.

A binding has several characteristics: **transport** (http, named pipes, tcp, msmq, …), **encoding** (text, binary, MTOM [Message Transmission Optimization Mechanism]), **protocol** (security, transaction, reliable message capability).

Services can be self-hosted in a console application, or hosted under IIS.

Services can be configured using code or using a configuration file.

Services publish metadata that define the information a client application needs to communicate with the service. VS2012 automates the process of accessing this metadata and configure client application for the service, or use ServiceModel Metadata Utility Tool (Svcutil.exe) to configure client app for the service.

## Contract

In WCF, all services are exposed as contracts. Contract is a platform-neutral and standard way of describing what the service does. Mainly there are four types of contracts available in WCF.

### Service Contract

Service contracts describe the operation that service can provide. For example, a Service provide to give the temperature of the city based on the zip code, this service is called as Service contract. It will be created using Service and Operational Contract attribute.

### Data Contract

Data contract describes the custom data type which is exposed to the client. This defines the data types that are passed to and from service. Data types like int, string are identified by the client because it is already mention in XML schema definition language document, but custom created class or data types cannot be identified by the client e.g. Employee data type. By using DataContract we can make client to be aware of Employee data type that are returning or passing parameter to the method.

### Message Contract

Default SOAP message format is provided by the WCF runtime for communication between Client and service. If it is not meeting your requirements then we can create our own message format. This can be achieved by using Message Contract attribute.

### Fault Contract

Suppose the service I consumed is not working in the client application. I want to know the real cause of the problem. How I can know the error? For this we are having Fault Contract. Fault Contract provides documented view for error occurred in the service to client. This helps us to easy identity, what error has occurred.

## Service Host

Service Host object is in the process of hosting the WCF service and registering endpoints. It loads the service configuration endpoints, apply the settings and start the listeners to handle the incoming request. System.ServiceModel.ServiceHost namespace hold this object. This object is created while self-hosting the WCF service.

Example of a service self-hosted in a console application:

//Creating uri for the hosting the service

Uri uri = new Uri("http://localhost/CategoryService");

//Creating the host object for MathService

ServiceHost host = new ServiceHost(typeof(CategoryService), uri);

//Adding endpoint to the Host object

host.AddServiceEndpoint(typeof(ICategoryService),new WSHttpBinding(), uri);

host.Open(); //Hosting the Service

Console.WriteLine("Waiting for client invocations");

Console.ReadLine();

host.Close();

## Message and Channel

### Message

WCF Message is the unit of data exchange between client and service. It consists of several parts, including a body and headers.

### WCF Runtime

WCF runtime is the set of objects responsible for sending and receiving message. For example formatting the message, applying security and transmitting and receiving message using various protocol.

### Channels

Channels are the core abstraction for sending message to and receiving message from an Endpoint. Broadly we can categories channels as:

Transport Channels: Handle sending and receiving message from network. Protocols like HTTP, TCP name pipes and MSMQ.

Protocol Channels: Implement SOAP based protocol by processing and possibly modifying message, for instance, WS-Security and WS-Reliability.

## Comparison with Web Services

|  |  |  |
| --- | --- | --- |
| Features | Web Service | WCF |
| Hosting | It can be hosted in IIS | It can be hosted in IIS, windows activation service, Self-hosting, Windows service |
| Programming | [WebService] attribute has to be added to the class | [ServiceContract] attribute has to be added to the class |
| Model | [WebMethod] attribute represents the method exposed to client | [OperationContract] attribute represents the method exposed to client |
| Operation | One-way, Request- Response are the different operations supported in web service | One-Way, Request-Response, Duplex are different type of operations supported in WCF |
| XML | System.Xml.serialization name space is used for serialization | System.Runtime.Serialization namespace is used for serialization |
| Encoding | XML 1.0, MTOM(Message Transmission Optimization Mechanism), DIME, Custom | XML 1.0, MTOM, Binary, Custom |
| Transports | Can be accessed through HTTP, TCP, Custom | Can be accessed through HTTP, TCP, Named pipes, MSMQ,P2P, Custom |
| Protocols | Security | Security, Reliable messaging, Transactions |

## Some protocols supported by WCF binding

|  |  |
| --- | --- |
| Binding | Description |
| BasicHttpBinding | Basic Web service communication. No security by default |
| WSHttpBinding | Web services with WS-\* support. Supports transactions |
| WSDualHttpBinding | Web services with duplex contract and transaction support |
| WSFederationHttpBinding | Web services with federated security. Supports transactions |
| MsmqIntegrationBinding | Communication directly with MSMQ applications. Supports transactions |
| NetMsmqBinding | Communication between WCF applications by using queuing. Supports transactions |
| NetNamedPipeBinding | Communication between WCF applications on same computer. Supports duplex contracts and transactions |
| NetPeerTcpBinding | Communication between computers across peer-to-peer services. Supports duplex contracts |
| NetTcpBinding | Communication between WCF applications across computers. Supports duplex contracts and transactions |

## Config files examples

### Service

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint

address="http://localhost:8090/MyService/MathService.svc"

contract="IMathService"

binding="wsHttpBinding" />

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

### Binding

Consider a scenario say, I am creating a service that has to be used by two types of clients. One of the client will access SOAP using http, and other client will access Binary using TCP. How it can be done? With Web service it is very difficult to achieve, but in WCF it's just we need to add extra endpoint in the configuration file.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint address="http://localhost:8090/MyService/MathService.svc"

contract="IMathService"

binding="wsHttpBinding"/>

<endpoint address="net.tcp://localhost:8080/MyService/MathService.svc"

contract="IMathService"

binding="netTcpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

### Scope

Definitions are applicable at a scope and below: common behaviors affect all endpoints globally, service behaviors affect only service-related aspects, endpoint behaviors affect only endpoint-related properties, and operation-level behaviors affect particular operations.

In the below configuration information, I have mentioned the Behavior at Service level. In the service behavior I have mention the serviceMetadata node with attribute httGetEnabled='true'. This attribute will specifies the publication of the service metadata. Similarly we can add more behavior to the service.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint address="" contract="IMathService"

binding="wsHttpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

Note: Application can be controlled either through coding, configuring or through combination of both. Specification mention in the configuration can also be overwritten in code.

# MS Tutorial

Note: Need to run VS2012 as an administrator

## Define a Windows Communication Foundation Service Contract

<http://msdn.microsoft.com/en-us/library/ms731835.aspx>

Create a new project WCF Library

The project contains 3 files: App.config, IService1.cs (default service contract) and Service1.cs (default implementation of the service contract). The App.config file contains configuration needed to load the default service with the Visual Studio WCF Service Host.

In IService1.cs, IService1 interface is marked with ServiceContract attribute

Each member of IService1 is marked with OperationContract attribute

## Implement a Windows Communication Foundation Service Contract

<http://msdn.microsoft.com/en-us/library/ms734686.aspx>

## Host and Run a Basic Windows Communication Foundation Service

<http://msdn.microsoft.com/en-us/library/ms730935.aspx>

Create a new console application to host the service, add a reference to WCF dll and System.ServiceModel.

In the hosting app:

Step 1 - Creates an instance of the Uri class to hold the base address of the service. Services are identified by a URL which contains a base address and an optional URI. The base address is formatted as follows: [transport]://[machine-name or domain][:optional port #]/[optional URI segment]

Step 2 – Creates an instance of the ServiceHost class to host the service. The constructor takes two parameters, the type of the class that implements the service contract, and the base address of the service.

Step 3 – Creates a ServiceEndpoint instance. A service endpoint is composed of an address, a binding, and a service contract. The ServiceEndpoint constructor therefore takes the service contract interface type, a binding, and an address. The address is appended to the base address to identify the endpoint. The address specified in this code is “Calculator” so the fully qualified address for the endpoint is “http://localhost:8000/ServiceModelSamples/Service/Calculator”

Note: Adding a service endpoint is optional when using .NET Framework 4 or later. In these versions, if no endpoints are added in code or configuration, WCF adds one default endpoint for each combination of base address and contract implemented by the service.

Step 4 – Enable metadata exchange. Clients will use metadata exchange to generate proxies that will be used to call the service operations. To enable metadata exchange create a ServiceMetadataBehavior instance, set its HttpGetEnabled property to true, and add the behavior to the Behaviors collection of the ServiceHost instance.

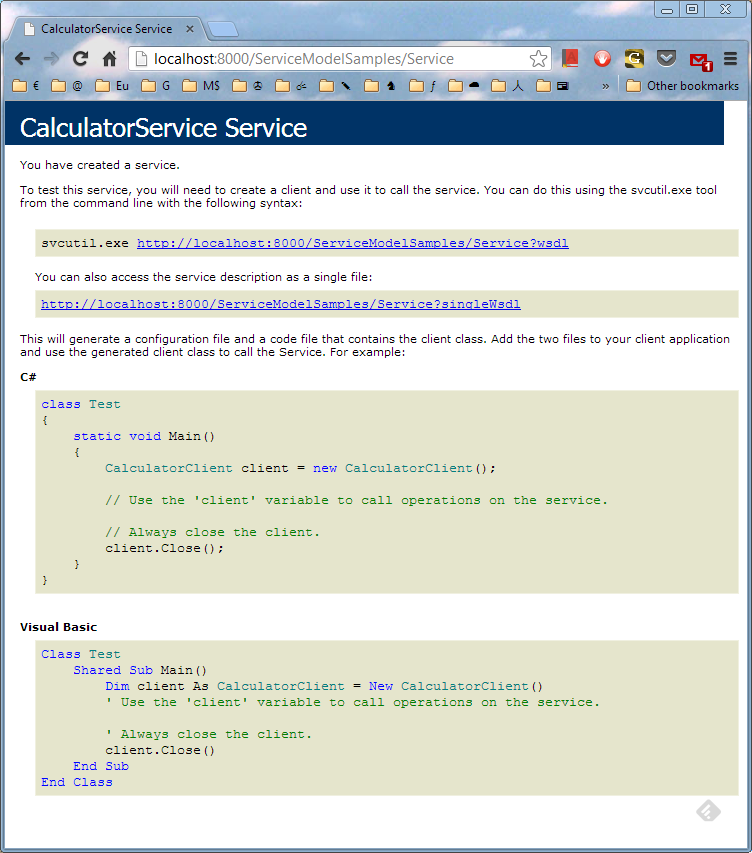
Step 5 – Open the ServiceHost to listen for incoming messages. Notice the code waits for the user to hit enter. If you do not do this, the app will close immediately and the service will shut down.

Give security access to current user:

In netsh utility, run in an administrator cmd window:

http add urlacl url=http://+:8000/ServiceModelSamples/Service/ user=THOR\Pierre

When the service is running, browser service debug page at http://localhost:8000/ServiceModelSamples/Service

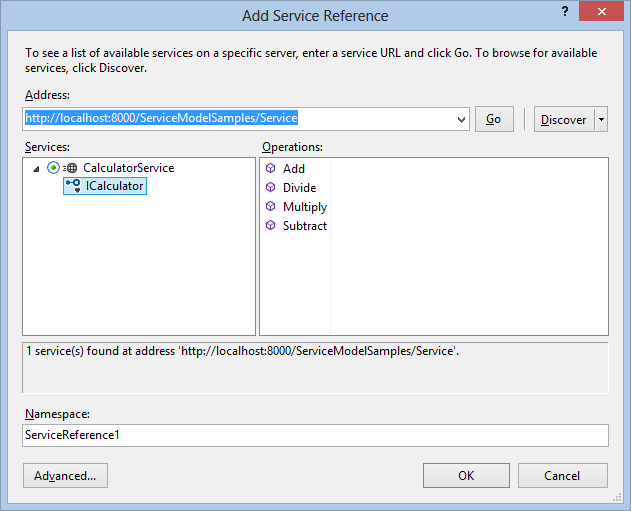


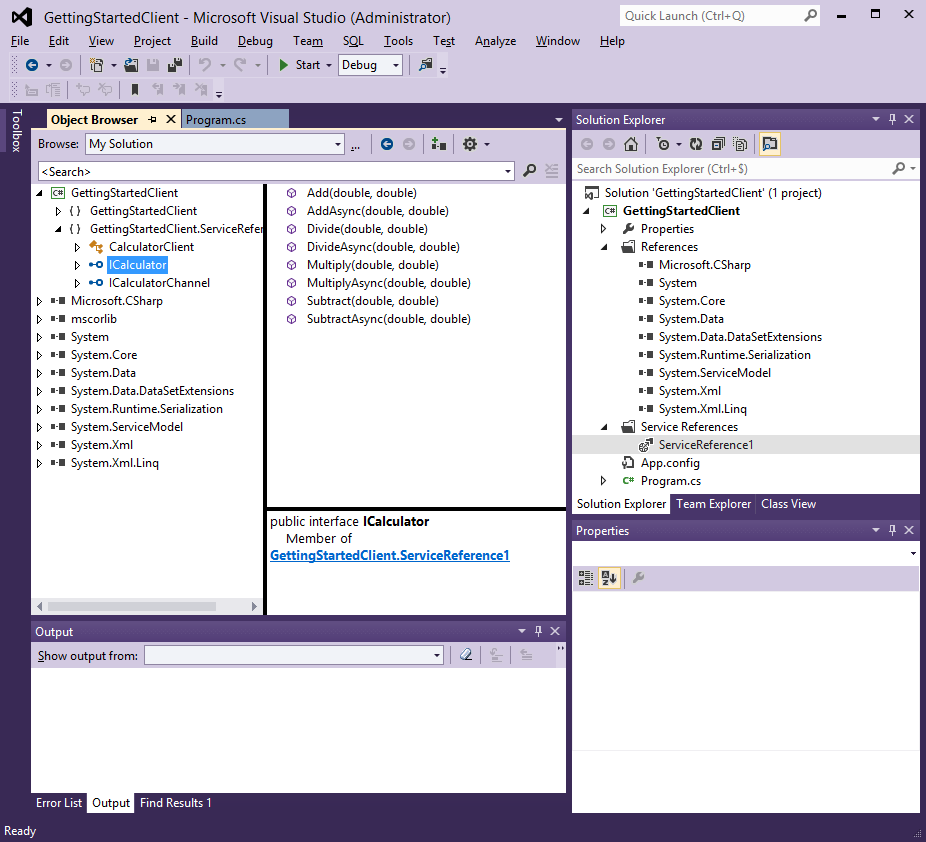
## Create a Windows Communication Foundation Client

<http://msdn.microsoft.com/en-us/library/ms733133.aspx>

When adding service reference, GettingStartedHost must be running.

Adding service reference:





## Configure a Basic Windows Communication Foundation Client

<http://msdn.microsoft.com/en-us/library/ms734663.aspx>

## Use a Windows Communication Foundation Client

<http://msdn.microsoft.com/en-us/library/ms730144.aspx>

