**===WCF Session Points to remember**

**==What is proxy in WCF?**

A proxy is a class by which a service client can interact with the service. By the use of proxy in the client application we can call different methods in the service (i.e. in WCF).

==**KnownType Attribute in WCF**

According to MSDN the KnownTypeAttribute class allows you to specify, in advance, the types that should be included for consideration during deserialization. The WCF service generally accepts and returns the base type. If you expect the service to accept and return an inherited type then we use the knowntype attribute.

**==Whats the diff between Message Contract and Data Contract.**

Data contracts are used to describe data types used by service.They can be either parameter or return type.

Message Contract are used to describe SOAP message format.They allow us to cotnrol the detail is SOAP header and body.

**==Exception handling of WCF Service**

<serviceBehaviors>

<behavior name="">

<serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

We can use FaultException class to catch exception

To report exceptions from a service to its caller, the exception too must be serialized to a standard format and wrapped in a message. A SOAP fault is a standard format for transfer of exception between distributed applications.

From my understanding if you are using wsHttpBinding and if you throw a .Net exception rather than a FaultException the Server-Client channel will be at faulted state, and the proxy class object will have to be recreated as the existing proxy object will be unusable. So the best practice might be to use FaultException.

They're not same. In a WCF service, if it throws an exception inside the service, the client will not get the details. In order to get the formatted exception details on client side, you need to use FaultException instead to let the client know the details. The FaultException information can be serialized as expected.

In summary, if you want the client to know the exception details in WCF service, you need to enable fault contract with FaultException.

**==Centralized error handling WCF**

To handle centralized error in WCF we Implement IErrorHandler which includes

ProvideFault() & HandleError() Methods which automatically gets called when Fault or un-handled exception occurs.

In Asp.net App use Application\_error Event to log all the exception and redirect to custom page.

Create a custom Service Behaviour Attribute

Decorate Service with GlobalErrorHandler Attribute

It is good practice to handle the exceptions at the methods that raises the exception. But sometimes, we may need a global solution for handling un-handled exceptions to avoid client objects going into fault state

**==Choosing the Appropriate WCF Binding**

<http://www.c-sharpcorner.com/UploadFile/dhananjaycoder/abc-of-an-endpoint-in-wcf/>

===When to Use the HTTP Transport

In WCF, the HTTP transport binding is optimized for interoperability with legacy non-WCF systems. If all communicating parties are using WCF, the TCP-based or named pipes-based bindings are faster.

If we are going to develop a WCF service that will be consumed by non-WCF client applications, then we can expose our service using BasicHttpBinding or WsHttpBinding.

If you are required to create a binding that is globally inter-operable, use http binding.

===When to Use the NetTCP Transport

TCP is a connection-based, stream-oriented delivery service with end-to-end error detection and correction. Connection-based means that a communication session between hosts is established before exchanging data.

For internal use, but consumers are in separate servers use tcp binding. It less inter-operable, but using binary encoding which is faster.

If we need to communicate across computers with same .NET technology on intranet, then netTcpBinding or netPeerTcpBinding options are available. It’s basically the replacement or enhancement of earlier .NET Remoting technology.

The WCF TCP transport is optimized for the scenario where both ends of the communication are using WCF. This binding is the fastest WCF binding for scenarios that involve communicating between different machines. The message exchanges use the BinaryMessageEncodingBindingElement for optimized message transfer. TCP provides duplex communication and so can be used to implement duplex contracts, even if the client is behind network address translation (NAT).

NetTcpBinding can be useful where IIS services are not needed.

Check for the network and firewall settings, as for NetTcp binding access is required for the server network where the service is hosted.

==When to Use the Named Pipe Transport\netNamedPipeBinding

When communication is required between different WCF applications on a single computer, and you want to prevent any communication from another machine, then use the named pipes transport. An additional restriction is that processes running from Windows Remote Desktop may be restricted to the same Windows Remote Desktop session unless they have elevated privileges.

Use named pipe binding for intra-server communication that is for consumers hosted in same servers. Named pipe binding is the fastest one in wcf allowed bindings.

1. BasicHttpBinding is designed to replace ASMX Web services. It supports both HTTP and Secure HTTP. As far as encoding is concerned, it provides support for Text as well as MTOM encoding methods. BasicHttpBinding doesn’t support WS-\* standards like WS-Addressing, WS-Security and WS-ReliableMessaging.
2. WsHttpBinding also supports interoperability. With this binding, the SOAP message is, by default, encrypted. It supports HTTP and HTTPS. In terms of encoding, it provides support for Text as well as MTOM encoding methods. It supports WS-\* standards like WS-Addressing, WS-Security and WS-ReliableMessaging. By default, reliable sessions are disabled because it can cause a bit of performance overhead.
3. WsDualHttpBinding has all features of WsHttpBinding with addition that it supports Duplex MEP (Message Exchange Pattern). In this MEP, service can communicate with client via callback. It's basically a two way communication.
4. WsFederationHttpBinding is a specialized form of WS Binding that offers support for federated security.
5. NetNamedPipeBinding is secure and reliable binding on a single WCF computer across process communication. It provides support for binary encoding which is the best choice in this scenario and uses named pipes as transport for SOAP messages.
6. NetTcpBinding supports reliability, transactions and security. It also supports TCP protocol and binary as encoding method. We can say that it’s the most optimized or fastest binding because both client and service are on the same WCF technology.
7. NetPeerTcpBinding supports features as that of netTcpBinding but it provides secure binding for peer-to-peer environment with WCF Services.
8. NetMsmqBinding is required in a cross machine environment with secure and reliable queued communication. This uses MSMQ as transport.

**==When do asp.net developer\WCF uses a windows Service?**

Generally we use Windows Service to run the code in the background all the time, without any Sort of User Reaction

When you need the application to start running even when no one has physically logged into the machine,

This makes our WCF Service always available for clients.

To host Windows Service we need to use windows service project type &

on service.cs file add project installer and configure it then,

using installutil -i ..path of windows service host.exe

Windows Service WCF hosted Service is difficult to debug, we need to attach process windowservice.exe to debug that service.

**==WCF with IIS**

To host using IIS we need to add wcf service file(i.e. .svc file) and then

<%@ ServiceHost Language="C#" Debug="true" Service="HelloService.HelloService" %> on .svc file directive

which contains relative name of the service.

then publis in IIS.

Advantages of IIS hosting

No code required to host the service as .svc file responsible to create a instance of service

**==WAS => Windows Activation Service**

WAS is a Process Activation Service that comes with Windows Vista and IIS 7. WAS was designed to support non-HTTP protocols with the help of IIS 7 whereas IIS 6 supports only HTTP protocols. If you want to use Non-HTTP Protocols like NetTcpBinding and NetMSMQBinding over IIS, WAS hosting is the only option.

The biggest limitation in the WCF architecture before IIS 7 version was no support for non-HTTP protocols over IIS. To enable the usage of a complete list of WCF-Supported protocols including non-HTTP, such as net.tcp, net.pipe, net.msmq - WAS (Windows Activation Service) is the platform to be used.

**==One Way Call**

they are look like asynchronous call, but they are not caz one way call can still block the client, if the no. of messages waiting to be processed

has exceeded the server queue limit.

**==Duplex call**

Specify the CallBack Contract & associate with Service Contract

[ServiceContract(CallbackContract=typeof(IReportServiceCallBack))]

public interface IReportService

{}

**==MTOM in WCF**

to preferred approch to large binary messages in wcf is to use MTOM message encoding.

like downloading file using wcf service use wshttp binding and messageencoding=MTOM instead of text encoding which is default in WCF

**==InstanceContextMode in WCF**

Percall\PerSession\Single

[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single)]

public class Service1 : IService1

Use of per call InstanceContextMode mode

Better Memory usage \ Concurrency not an issue \ State is not maintained \ application scalable \

**==How to design WCF service like per session or per call**

if you are using object oriented style then per session else if you preferred SOA then Per Call.

all binding except BasicHTTPBinding per session InstanceContextMode work as per call as it does not maintain state

For Single instance context mode concurrency is an issue,we set-up to access service instance by single thread instance at a time.

**==SessionMode Enumeration in WCF**

use SessionMode Enumeration with the service contract to require, allowed(Default) or NotAllowed to use sessions in WCF service

**==Method OverLoading in WCF**

[ServiceContract]

public interface IMethodOverload

{

[OperationContract(Name = "Intsum")]

int sum(int a, int b);

[OperationContract(Name = "Double\_sum")]

double sum(double a, double b);

[OperationContract(Name = "Str\_sum")]

string sum(string a, string b);

}

Set Attribute on OperationContract is Name

and while consuming in app. use that name to invoke service i.e. Intsum,Double\_sum

**==Concurrency Mode in WCF**

Single,Multiple & Reentrant

[ServiceBehavior(ConcurrencyMode = Con currencyMode.Multiple)]

Single=only one thread allowed to access the service instance of WCF at a time and set lock to resource

Multiple=multiple thread allowed to access service instance of WCF

Reentrant=Use to handle Concurrency in call back service

**== WCF throttling**

Controlling the number of instance and session for WCF Service, we have to use throttling. In throttling, you have to specify the number for the concurrent call and concurrent session, and the concurrent instance allows for the client to communicate with WCF Service. In throttling, you can specify it in two ways: in the app.config file of the Service or you can specify it at the hosting time programmatically.

WCF throttling provides some properties that you can use to limit how many instances or sessions are created at the application level. Performance of the WCF service can be improved by creating proper instance.

ServiceHost host = new ServiceHost(typeof(ThrottlingInWCF.ThrottlingInWCF));

ServiceThrottlingBehavior stb= new ServiceThrottlingBehavior();

stb.MaxConcurrentCalls=2;

stb.MaxConcurrentInstances=2;

stb.MaxConcurrentSessions=2;

host.Description.Behaviors.Add(stb);

host.Open();

<behaviors>

<serviceBehaviors>

<behavior name="ServiceBehavior">

<serviceMetadata httpGetEnabled="true"/>

<serviceDebug includeExceptionDetailInFaults="true "/>

<serviceThrottling maxConcurrentCalls="500"

maxConcurrentInstances ="100"

maxConcurrentSessions ="200"/>

</behavior>

</serviceBehaviors>

</behaviors>

**==Security in WCF**

Default Security For NetTCPBinding is Transport & For WSHTTPBinding is Message.

Basic HTTP Binding Doesn't Have Transport+ Message Security But We can implement using some Certificate

**==Difference between Transport security & Message security in WCF**

With transport use are securing the channel you are using, and with message the message (content) you are sending.

Transport level security happens at the channel level. Transport level security is the easiest to implement as it happens at the communication level. WCF uses transport protocols like TCP, HTTP, MSMQ etc and every of these protocols have their own security mechanisms.

Message level security is implemented with message data itself. Due to this it is independent of the protocol. Some of the common ways of implementing message level security is by encrypting data using some standard encryption algorithm.

<http://www.c-sharpcorner.com/blogs/difference-between-transport-and-message-level-security1>

**===There are four core security features that WCF addresses:**

**Confidentiality**: This feature ensures that information does not go in to the wrong hands when it travels from the sender to the receiver.

**Integrity**: This feature ensures that the receiver of the message gets the same information that the sender sent without any data tampering.

**Authentication**: This feature verifies who the sender is and who the receiver is.

**Authorization**: This feature verifies whether the user is authorized to perform the action they are requesting from the application

<https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/configuring-system-provided-bindings>

**Transport Level Security**

It secures the actual transport (i.e. the pipe) over which the message passes through from client to a service. For example it uses SSL (Secure Socket Layer) to ensure point-to-point protection.

**Message Level Security**

It secures the message itself that is being transported from client to a service and vice versa.

**===Message security using Protection Level parameter In WCF**

When you are using WsHttpBinding by the fault the message is encrypted and sign.

by using security mode=None we can disable security

protection level enum present in System.Net.Security;

Available Parameters for Protecting Data

None

Sign – It confirms the message has not been tampered or changed by anyone. It signs all the messages on the wire to provide the message integrity.

Encrypt and Sign – It conforms the message we send is confidential and it has not been tampered anywhere by anyone.

**==The Protection levels can be done at all the levels:**

Service Contract

Operation Contract

Message Contract

Message Header

Message Body

E.g. [OperationContract(ProtectionLevel=ProtectionLevel.None\ProtectionLevel.Sign\ProtectionLevel.EncryptAndSign)]

basicHttpBinding doesnot support ProtectionLevel parameter, if we set then exception will be thrown.

In general ProtectionLevel parameter is used to enforce the minimum level of protection required. If the binding does not provide that minimum level of protection then an exception will be thrown

**==To Customize Authentication Scheme we used clientcredentialType**

<wsHttpBinding>

<binding name="WShttp">

<security mode="Transport">

<transport clientCredentialType="Windows"></transport>

</security>

</binding>

</wsHttpBinding>

<netTcpBinding>

<binding name="NetTcp">

<security mode="Message">

<message clientCredentialType="Windows"/>

</security>

</binding>

</netTcpBinding>

**==NetTcp Binding and Transport Level Security**

netTcpBinding provides transport security. Even with transport security, all messages are encrypted and signed

netTcpBinding which provides transport security.

Run the WCF service and the client. When you inspect the logged messages, surprisingly they are in plain text. The reason for this is that, the messages are encrypted and signed at the transport layer. By the time the message is arrived at the log it is already decrypted. Hence, they appear in plain text

**==Configure wsHttpBinding to use transport security**

Create a self-signed SSL certificate

In IIS, add Https binding using the self-signed SSL certificate created

In IIS, configure SSL settings for WCFService

n web.config, include the following bindings section, to customize the wsHttpBinding to use Transport security and Basic client credential type.

<bindings>

<wsHttpBinding>

<binding name="wsHttp">

<security mode="Transport">

<transport clientCredentialType="Basic"></transport>

</security>

</binding>

</wsHttpBinding>

</bindings>

**==Configure netTcpBinding to use message security**

<binding name="NetTcp">

<security mode="Message">

<message clientCredentialType="Windows"/>

</security>

</binding>

</netTcpBinding>

**===Restful Service with WCF**

<http://www.c-sharpcorner.com/UploadFile/0c1bb2/creating-wcf-rest-service/>

<http://www.c-sharpcorner.com/UploadFile/0c1bb2/creating-wcf-rest-service/>

<https://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services>

REST stands for Representational state transfer which is a technique to communicate on cross platform application and exchange the data in JSON or XML format with the help of GET, POST, PUT, and DELETE methods of HTTP protocol.

Actually only the difference is how clients access our service. Normally, a WCF service will use SOAP, but if you build a REST service, clients will be accessing your service with a different architectural style (calls, serialization like JSON, etc.).

**Why and Where to Use REST**

Few days back, I was writing a service which was supposed to be accessed by heterogeneous language/platform/system. It can be used by iPhone, Android, Windows Mobile, .NET web application, JAVA or PHP. Using web service, it was bit complex for me to expose it to everyone using uniform system. Then we decided to use REST,

Let us briefly understand about the HTTP methods which is most commonly used to create WCF REST service:

GET : Get the resource (Records) from particular source such as SQL database.

POST : Used to insert the records into the particular source such as SQL, Oracle database.

PUT : Used to modify the resource or records.

DELETE : Used to delete the specific resource or record from particular source.

**==Transaction in WCF**

Let us say salary credited to your account. This transaction is having two activity i.e. your account will get ++ (with your salary amount) and your company account will get -- (with your salary amount).

So for above things in WCF Transaction is used.

There are only few bindings in WCF that can support transaction, these are NetTcpBinding, NetNamedPipeBinding, WSHttpBinding, WSDualHttpBinding, and WSFederationHttpBinding.

Steps

[OperationContract]

[TransactionFlow(TransactionFlowOption.Mandatory\Allowed\NotAllowed )]

void Deposit(int accountid,int amount);

[ServiceBehavior(InstanceContextMode=InstanceContextMode.PerSession,TransactionAutoCompleteOnSessionClose=true)]

public class Service1 : IService1

{

int counter;

int Amount;

[OperationBehavior(TransactionScopeRequired=true,TransactionAutoComplete=false)]

public void Deposit(int accountid, int amount)

{

}

}

Config in Host

<!--config for tranaction imp-->

<bindings>

<wsHttpBinding>

<binding name="WsHttpConfig" transactionFlow="true">

<reliableSession enabled="true"/>

</binding >

</wsHttpBinding>

</bindings>

<!--config for tranaction imp-->

At client, add reference to System.Transaction

using(TransactionScope scope = new TransactionScope())

{

IserviceClient client = new IserviceClient();

client.upload();

client.upload();

scope.Complete();

MessageBox.Show("All is well!!!");

}