Overview of WCF 4.5

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
23rd June 2011	Version 1.0	Vaishali Kasture	Content Creation
22nd Sept 2011	Version 1.1	Vaishali Kasture	Modifications done in the Lab Book
19th Feb 2015	Version 1.1	Vaishali Kasture	Modifications based on 4.5

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning WCF version 4.0. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for Overview of WCF 4.5

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 8.0 or higher
- SQL Server 2010
- Visual Studio 2010

Please ensure that the following is done:

- Visual Studio 2010 and IE Explorer 8 is installed

Instructions

- All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a
- subdirectory WCF 4.0_assgn. For each lab exercise create a directory as lab
- <lab number>.
- You may also look up the on-line help provided in the MSDN library.

Learning More (Bibliography if applicable)

- WCF 4.0 Step by Step

Lab 1. Windows Communication Foundation Contracts and Bindings

Goals	At the end of this lab session you will be able to do the following: • Define Service Contract • Implement Service Contract • Consume WCF Service
Time	2 hours

In this exercise, you will define and implement the contract for a Windows Communication Foundation service. That service will calculate the Sum and Product of two numbers.

Solution:

Task 1: Create the Calculator Service Project

1. Open Visual Studio 20010, and create a new blank solution called, WCF1, in D:\Windows Communication Foundation\Labs\.
2. Add a new Project WCF Service Library give the name as Contract
3. Rename Service1.cs to Maths.cs and IService.cs to IMath.cs
4. Delete the exsisting code from IMath.cs file
5. Your IMath.cs file should have the following code.

```
namespace Contract
{
   // NOTE: You can use the "Rename" command on the "Refactor" menu to change
the interface name "IService1" in both code and config file together.
   [ServiceContract]
   public interface IMath
   {

           [OperationContract]
           int Add(int num1, int num2);

           [OperationContract]
           int Multiply(int num1, int num2);

           [OperationContract(IsOneWay = true)]
           void Logout();
       }

   }
```

Your Maths.cs file should have the following code delete the exsisting code from Maths.cs namespace Contract

```
{
   // NOTE: You can use the "Rename" command on the "Refactor" menu to change
the class name "Service1" in both code and config file together.
   public class Maths : IMath
   {

         public int Add(int num1, int num2)
         {
            return num1 + num2;
         }

         public int Multiply(int num1, int num2)
         {
            return num1 * num2;
         }


         public void Logout()
         {
            //do some activity.. it may take time .. but it is one way.. fire & forget
            System.Threading.Thread.Sleep(5000);
         }


      }

   }
```

6. Save the Project
7. Run the application
8. You will get the message your service has been hosted and WCF Host will be displayed
9. You can view the WCF Test client click on the Add method specify 10 and 10 for values and click on Invoke the result will be displayed 20 Note:The WCF service that you have built runs the same way as a regular Web application and is hosted by a Web server. In this case, when you created the WCF service, you set the location to a folder in the file system, so when you run the service it will execute by using the ASP.NET Development Server provided with Visual Studio.
10. Create a new Console Application name it as Host

Note:

With WCF 4 and WCF 4.5, you can now use ServiceHost to host the CalculatorService service without any application configuration whatsoever. When using ServiceHost in custom hosting scenarios, you will need to specify one or more base addresses to use. The following shows how to host the MathService in a console application, and again, you can assume there's no app.config file associated with this program:

11. Add a reference of System.ServiceModel

```
using System.ServiceModel;
using System.ServiceModel.Description;
namespace Host{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceHost host = new ServiceHost(typeof(CalculatorService.Maths  ),
    new Uri("http://localhost:8080/Maths"),
    new Uri("net.tcp://localhost:8081/Maths"));
            host.Open();
            foreach (ServiceEndpoint se in host.Description.Endpoints)
                Console.WriteLine("A: {0}, B: {1}, C: {2}",
                    se.Address, se.Binding.Name, se.Contract.Name);
            Console.WriteLine("Press <Enter> to stop the service.");
            Console.ReadLine();
            host.Close();

        }
    }
}
```

This example configures the ServiceHost with two base addresses: one for HTTP and another for TCP. When you run this program, you'll see four endpoints printed to the console window .You'll get two for the HTTP base address, one per contract, and two for the TCP base address, again one per contract. This is all provided behind the scenes by the ServiceHost instance.

Notice how WCF chooses to use the BasicHttpBinding for the default HTTP endpoints and the NetTcpBinding for the default TCP endpoints. Remember, this default endpoint behavior only kicks in when the service has not been configured with any endpoints. If the console application is changed to configure the service with at least one endpoint, none of these default endpoints show up in the output.

12. Append this code after Host.open()

```
host.AddServiceEndpoint(typeof(CalculatorService.IMath ), new WSHttpBinding(),
"myendpoint");

host.AddDefaultEndpoints();
```

13. Add a Application Configuration file to the Host Application and write the following code

```
<configuration>
  <system.serviceModel>
    <protocolMapping>
      <add scheme="http" binding="webHttpBinding"/>
    </protocolMapping>
  </system.serviceModel>
</configuration>
```

Save and Run the application . The default HTTP-based endpoints will now show they're using the WebHttpBinding

14. Create a new Console Application named Client
    Add a reference of System.ServiceModel Browse through the App.config file
15. Add a Service reference of Calculator service
16. Click on Discovery
17. Give the namespace name as CalculatorServiceReference
18. Code should be as follows.

```
namespace Client
{
    class Program
    {
        static void Main(string[] args)
        {


            CalculatorServiceReference.MathClient proxy = new
CalculatorServiceReference.MathClient();
            Console.WriteLine(proxy.Add(10,10);
            Console.WriteLine(proxy.Multiply (10,10);
            Console.Read();
        }
    }
}
```

19. Look for the App.config file of the Client

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <wsHttpBinding>
                <binding name="WSHttpBinding_IMath" closeTimeout="00:01:00"
openTimeout="00:01:00"
                    receiveTimeout="00:10:00" sendTimeout="00:01:00"
bypassProxyOnLocal="false"
                    transactionFlow="false" hostNameComparisonMode="StrongWildcard"
                    maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
                    messageEncoding="Text" textEncoding="utf-8"
useDefaultWebProxy="true"
                    allowCookies="false">
                    <readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384"
                        maxBytesPerRead="4096" maxNameTableCharCount="16384" />
                    <reliableSession ordered="true" inactivityTimeout="00:10:00"
                        enabled="false" />
                    <security mode="Message">
                        <transport clientCredentialType="Windows"
proxyCredentialType="None"
```

```
                    realm="" />
                 <message clientCredentialType="Windows"
negotiateServiceCredential="true"
                    algorithmSuite="Default" />
              </security>
            </binding>
         </wsHttpBinding>
      </bindings>
      <client>
         <endpoint
address="http://localhost:8732/Design_Time_Addresses/CalculatorService/Service1/"
            binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IMath"
            contract="CalculatorServiceReference.IMath"
name="WSHttpBinding_IMath">
            <identity>
              <dns value="localhost" />
            </identity>
         </endpoint>
      </client>
   </system.serviceModel>
</configuration>
```

In Solution Explorer goto solution properties and set multiple startupproject –set host ,service and client as startup.

20. Run your application
21. Check for Single WSDL and Improved Intelisense feature of WCF 4.5