Thin Books

Licence Agreement
1) You are not allowed to distribute this book in any format.
2) You may not reproduce any part of this book
3) You add a link to your site to http://www.RealSoft.co.uk.
4) RealSoft reserves the right to remove this book without notice.
5) RealSoft reserves the right to change the terms of this agreement without notice.
6) The Term "Thin Books"™ is a trademark of RealSoft Limited
7) The Term "Thin Book"™ is a trademark of RealSoft Limited

The idea behind "Thin Books" is to provide you with the basics in any given topic, they do not try to dig into the depths but simply provide you with a quick start and from there you can decide to, or not to, read more details by searching the rest of the sprawling web.

Note this book does assume basic coding / c# knowledge

This is a Draft

Download Source Code : http://www.RealSoft.co.uk/files/RealSoft.WCF.BasicSample.zip

Thin Books

Table of Contents

Thin Books

1 What Is WCF – Windows Communication Foundation

WCF is a communication framework that allows inter-process communication, whether that be on the same machine or on different machines. It provides a unified approach to using different communication protocols such as HTTP , tcp, named pipes, MSMQ, WS* (Security enabled), and so on (it is extendible).
See http://msdn.microsoft.com/en-us/library/ms731082.aspx for further details.

1.1 WCF Terminology

WCF Terminology	Description
Message	This is a construct of one or more data items, the data items are bound by the message The message normally has a Header & Body where the header contains additional information about the message, such as encoding, security ect.
EndPoint	This is a concept and refers to the idea that communication occurs between points. A service can expose 1 or more EndPoints An EndPoint is made up of the following: 1) An Address: This is the location 2) A Binding: The communication protocol 3) A Service Contract: What can be sent
Address	This is the location where messages can be sent the following are examples of WCF Addresses: 1) http://www.RealSoft.co.uk/MyService //basic http endpoint 2) net.tcp://
Binding	This is the protocol that will be used.
Service Contract	This is the contract for the service.
Data Contract	This is the message contract.
Operation Contract	This defines what operations will be exposed as part of the Service
Data Member	This explicitly indicates a part of the message that is visible to the outside world
Host	What is going to host the service, the container the service will in.
Client	
Proxy	A impersonation of the service, this is what the client interacts with locally but the actual service can be anywhere.

2 Creating A WCF Service

This section will quickly step you through the basics of creating a WCF service using IIS as the Host

2.1 Service Contract

Before you can create a service you need to be able to outline what the service is going to do (Functional Specification)

In the good old tradition of computing we shall create a simple Hello service that simply replies Hello and the name of the person making the request and the current date.

From that simple description we can deduce the following requirements for the service Interface.

1) The service will have an Operation called Hello
2) The input to this operation is the name of the person
3) The output from this is the "Hello " + the name of the person, it will also return the current date

Input : string
Output : string , DateTime

2.2 DataContracts

Now that we know what the operation(s) are supported and what they expect we can write the DataContracts for the Service

Input : string , nothing special here

Output : string , DateTime → for this we will create a DataContrart called HelloResponse

```
using System;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace RealSoft.WCF.Sample{
   [DataContract]
   public class HelloResponse
   {
       [DataMember]
      public DateTime CurrentDate{get ;   set ; }

      [DataMember]
      public string  Speak{get ; set ;}
   }
}
```

Attributes:

[DataContract]: This indicates that this class is a DataContract, you must have this attribute to expose the data contract.
[DataMemeber]: This indicates that the property is to be exposed in the data contract, you must have this attribute.

2.3 Build The Service Contract

Now that you have the DataContract(s) you can create the ServiceContract

```
using System;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace RealSoft.WCF.Sample{
   [ServiceContract(Namespace="www.RealSoft.co.uk/2009/09/09")]
   public interface IHelloService
   {
      [OperationContract]
      HelloResponse  Hello(string personsName);
   }
}
```

Attributes:

[ServiceContract]: This indicates that the interface is the contract that will be exposed by the service, it is the operations in the Service contract that are exposed by the WCF service.

[OperationContract]: This indicates what operations are exposed by the Service, you must have this attribute otherwise the operation will not be exposed.

I've added the Namespace parameter for the ServiceContract as this is what allows you to version the service, the client will bind to this Namespace for this implementation.

2.4 Build The Concrete Service

At this stage you have the building blocks for the service all that remains is the implementation for the interface IHelloService, this is just a standard interface implementation, you do not have to do anything special here.

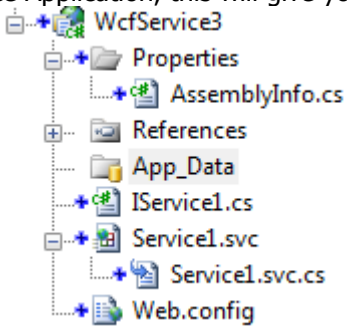
```
using System;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace RealSoft.WCF.Sample{
 public class HelloService : IHelloService
   {
      #region IHelloService Members
      public HelloResponse Hello(string personsName)
      {
         HelloResponse rep = new HelloResponse();
         rep.CurrentDate = DateTime.Now;
         rep.Speak =  "Hello " + personsName; //This is the expected output.
         return rep;
      }
      #endregion
   }
}
```

Build the DLL : RealSoft.WCF.Sample.dll

2.5 Hosting The Service In IIS

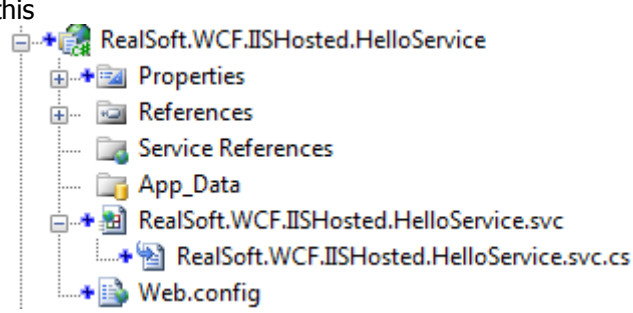
This section will detail how to build the hosting application.

1) Add a new Project of type WCF Service Application, this will give you the building blocks for the host.



2) Rename the Project to RealSoft.WCF.IISHosted.HelloService
3) Delete the Iservice.cs file
4) Rename Service1.svc to RealSoft.WCF.IISHosted.HelloService.svc
5) Right Mouse On RealSoft.WCF.IISHosted.HelloService.svc and select View Markup
   - You should have : <%@ ServiceHost Language="C#" Debug="true" **Service="WcfService3.Service1"** CodeBehind="RealSoft.WCF.IISHosted.HelloService.svc.cs" %> as the current markup
   - Change the **Service="WcfService3.Service1"** to **Service="RealSoft.WCF.IISHosted.HelloService"**

The Project should now look like this



6) Add a Refernce to the project to the dll you created for the service library **RealSoft.WCF.Sample.dll**
7) Open the RealSoft.WCF.IISHosted.HelloService.svc.cs file for edit.
8) Delete the class public class Service1 : Iservice1 and all its content
9) Create a class : public class HelloService : RealSoft.WCF.Sample.HelloService{}
10) Change the namespace from namespace WcfService3 to namespace RealSoft.WCF.IISHosted
11) Right mouse on the project RealSoft.WCF.IISHosted.HelloService and select Properties
12) Change the Assembly name to RealSoft.WCF.IISHosted.HelloService

13) Open the web.config file for editing, you shlould have a section that looks a bit like this:

```
<system.serviceModel>
  <services>
   <service name="WcfService3.Service1" behaviorConfiguration="WcfService3.Service1Behavior">
    <!-- Service Endpoints -->
    <endpoint address="" binding="wsHttpBinding" contract="WcfService3.IService1">
     <!--
        Upon deployment, the following identity element should be removed or replaced to reflect the
        identity under which the deployed service runs.  If removed, WCF will infer an appropriate identity
        automatically.
      -->
     <identity>
      <dns value="localhost"/>
     </identity>
    </endpoint>
    <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
   </service>
  </services>
  <behaviors>
   <serviceBehaviors>
    <behavior name="WcfService3.Service1Behavior">
     <!-- To avoid disclosing metadata information, set the value below to false and
          remove the metadata endpoint above before deployment -->
     <serviceMetadata httpGetEnabled="true"/>
     <!-- To receive exception details in faults for debugging purposes, set the value below to true.
        Set to false before deployment to avoid disclosing exception information -->
     <serviceDebug includeExceptionDetailInFaults="false"/>
    </behavior>
   </serviceBehaviors>
  </behaviors>
 </system.serviceModel>
```

14) Make the following changes to this file
15) Replace WcfService3.Service1 with RealSoft.WCF.IISHosted.HelloService
16) Replace WcfService3.IService1 with RealSoft.WCF.Sample.IHelloService
17) Clean and do a Rebuild
18) Test the Service is hosted
    1. Right Mouse on RealSoft.WCF.IISHosted.HelloService.svc and select View In Browser.

3 Build The Client – Consumer

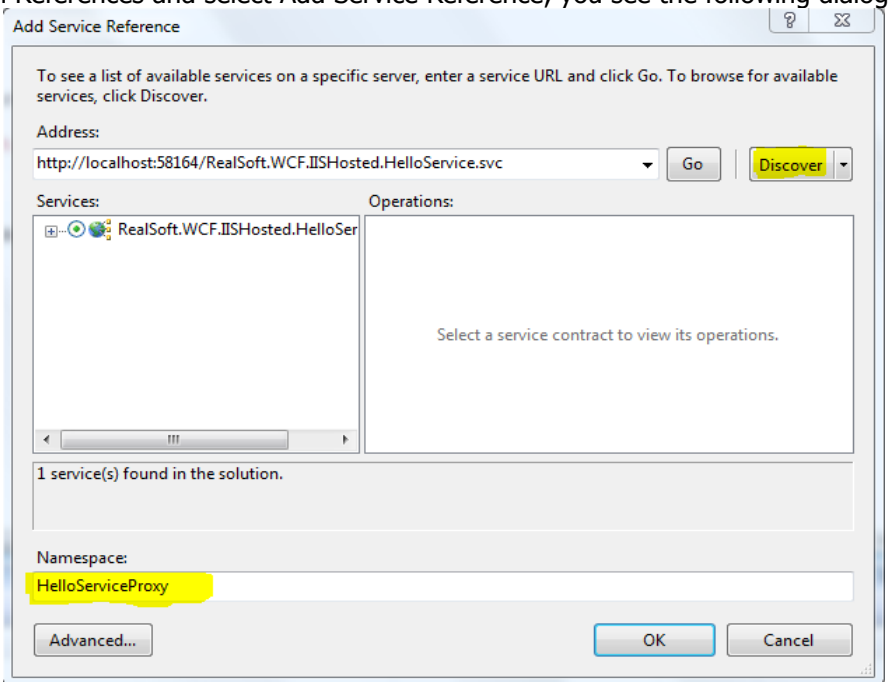
Now that you have a hosted WCF Service you need to build a client application.

3.1 Initial Project Set-up

1) Add a new project, let's use a console application, call this WCFClient

3.2 Add a Service References

1) Right mouse on References and select Add Service Reference, you see the following dialog box



2) Click the Discover Button
3) Change the Namespace to HelloServiceProxy
4) Click OK
5) Edit the Program.cs file so that it has the following

```
class Program
{
    static void Main(string[] args)
    {
        HelloServiceProxy.HelloServiceClient client = new WCFClient.HelloServiceProxy.HelloServiceClient();
        Console.WriteLine(client.Hello("Hitesh").Speak);
        Console.ReadKey();
    }
}
```

6) Now run the console application, this should show you that the client can talk to the service
7) We're done!