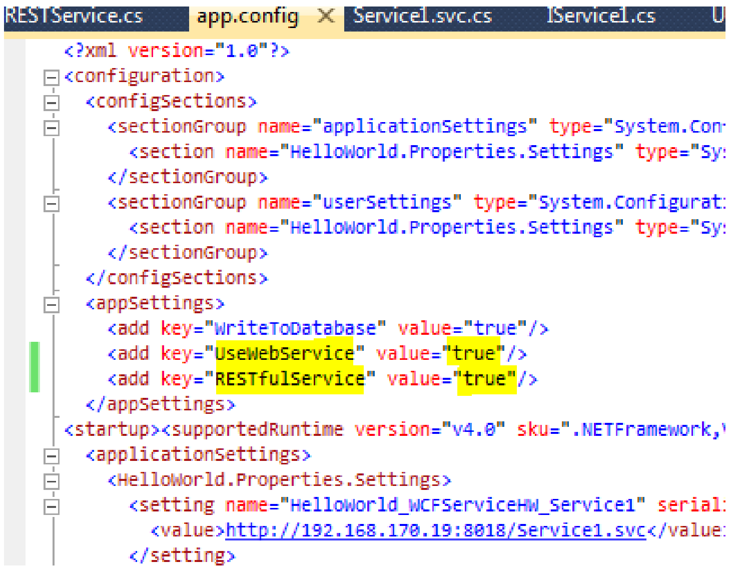
There are 5 projects in this solution:

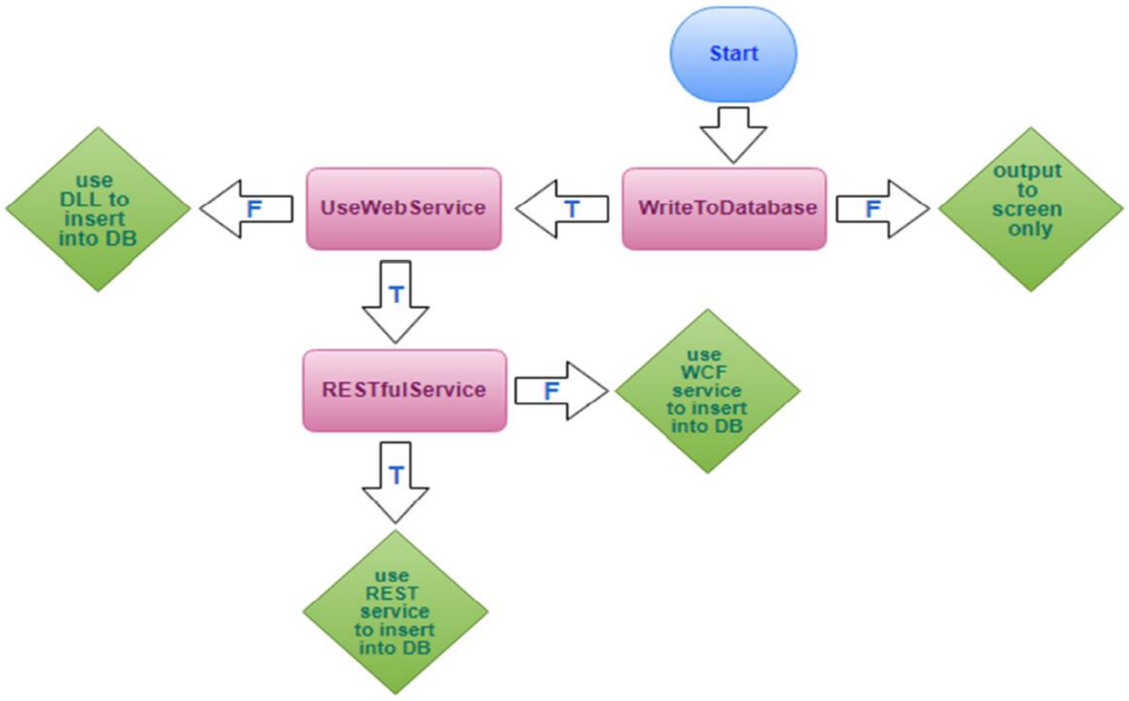
1. Console app HelloWorld
2. DllTest
3. RESTFulDemo
4. WcfService1
5. UnitTestHelloWorld

I added 3 different APIs for this console app to use. There are options to select which API (REST, SOAP, DLL) to use, in app.config of HelloWorld project.



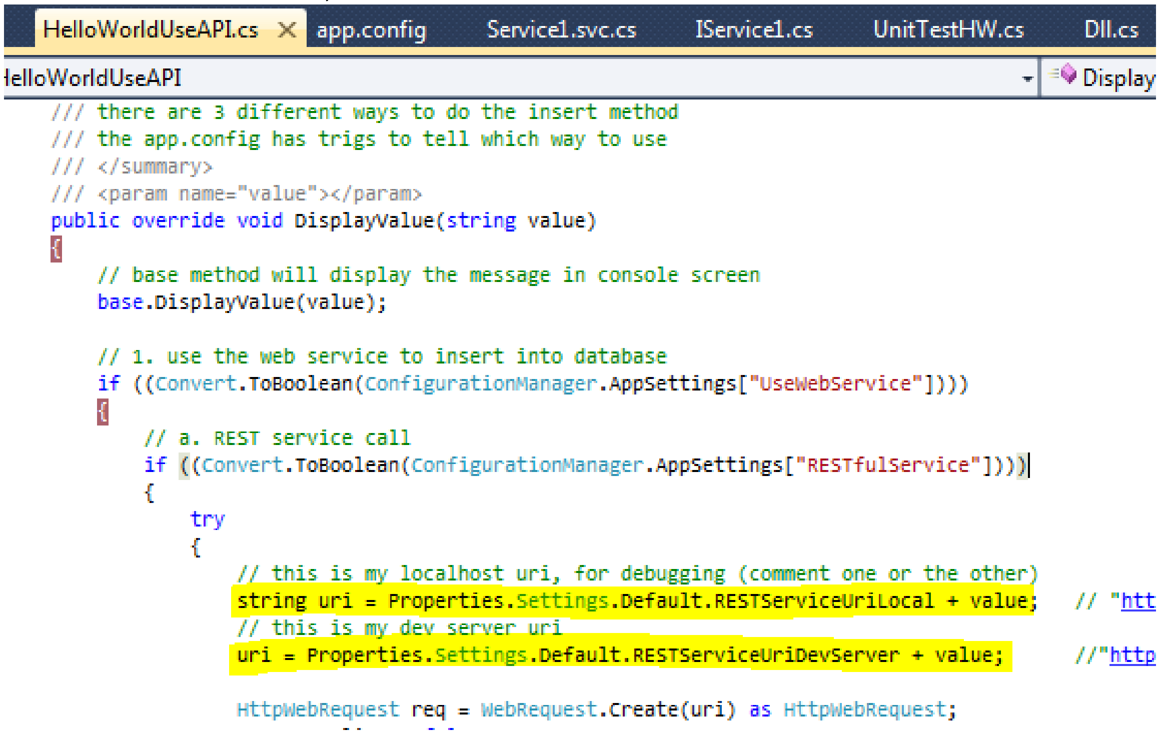
```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Con
      <section name="HelloWorld.Properties.Settings" type="Sy:
    </sectionGroup>
    <sectionGroup name="userSettings" type="System.Configurat:
      <section name="HelloWorld.Properties.Settings" type="Sy:
    </sectionGroup>
  </configSections>
  <appSettings>
    <add key="WriteToDatabase" value="true"/>
    <add key="UseWebService" value="true"/>
    <add key="RESTfulService" value="true"/>
  </appSettings>
<startup><supportedRuntime version="v4.0" sku=".NETFramework,'
  <applicationSettings>
    <HelloWorld.Properties.Settings>
      <setting name="HelloWorld_WCFServiceHW_Service1" serial:
        <value>http://192.168.170.19:8018/Service1.svc</value:
      </setting>
```

Flowchart below shows how to set the key values in app.config to call different APIs


Start
WriteToDatabase
F
output to screen only
T
UseWebService
F
use DLL to insert into DB
T
RESTfulService
F
use WCF service to insert into DB
T
use REST service to insert into DB

Some notes here:

1. In HelloWorldUseAPIs.cs, when the code calls REST service:

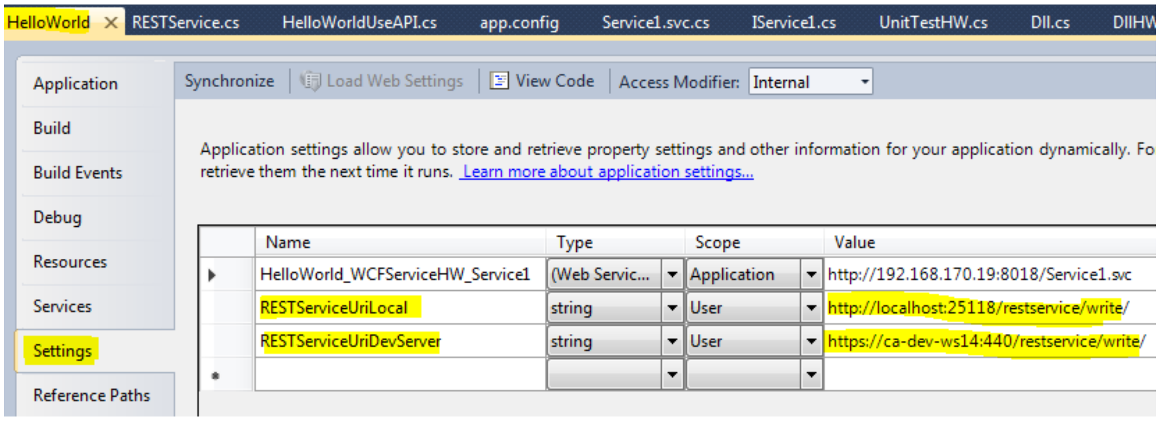


```
    /// there are 3 different ways to do the insert method
    /// the app.config has trigs to tell which way to use
    /// </summary>
    /// <param name="value"></param>
    public override void DisplayValue(string value)
    {
        // base method will display the message in console screen
        base.DisplayValue(value);

        // 1. use the web service to insert into database
        if ((Convert.ToBoolean(ConfigurationManager.AppSettings["UseWebService"])))
        {
            // a. REST service call
            if ((Convert.ToBoolean(ConfigurationManager.AppSettings["RESTfulService"])))
            {
                try
                {
                    // this is my localhost uri, for debugging (comment one or the other)
                    string uri = Properties.Settings.Default.RESTServiceUriLocal + value;   // "htt
                    // this is my dev server uri
                    uri = Properties.Settings.Default.RESTServiceUriDevServer + value;      //"http

                    HttpWebRequest req = WebRequest.Create(uri) as HttpWebRequest;
```

The uri values for localhost and dev server are set in:

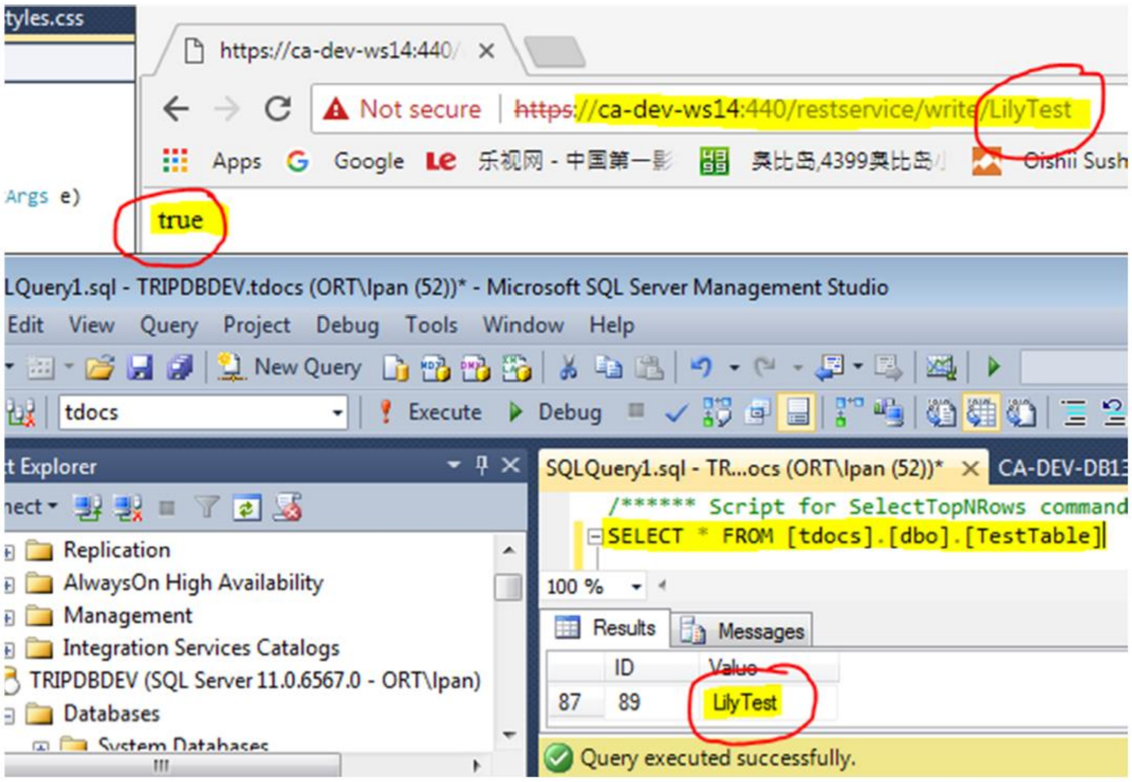


Application settings allow you to store and retrieve property settings and other information for your application dynamically. Fo
retrieve them the next time it runs. Learn more about application settings...

| | Name | Type | Scope | Value |
|---|---|---|---|---|
| ▸ | HelloWorld_WCFServiceHW_Service1 | (Web Servic... | Application | http://192.168.170.19:8018/Service1.svc |
| | RESTServiceUriLocal | string | User | http://localhost:25118/restservice/write/ |
| | RESTServiceUriDevServer | string | User | https://ca-dev-ws14:440/restservice/write/ |
| * | | | | |

These uri's are for my localhost and my dev server, please change to your uri values, and then comment one or the other to test it either on localhost or on server. My dev server is internal, so you will not be able to access it from internet.

We can also do a browser test for the REST service, and the result looks like this:
https://ca-dev-ws14:440/restservice/write/LilyTest has inserted the value into database table and returned "true" in the browser.
"Not secure" warning is because I made a self-signed certificate on my dev server which is not trusted by SSL.
This service works for HelloWorld console app too.


https://ca-dev-ws14:440/
Not secure
Apps
Google
Oishii Sush
LQuery1.sql - TRIPDBDEV.tdocs (ORT\lpan (52))* - Microsoft SQL Server Management Studio
Edit
View
Query
Project
Debug
Tools
Window
Help
New Query
tdocs
Execute
Debug
t Explorer
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
TRIPDBDEV (SQL Server 11.0.6567.0 - ORT\lpan)
Databases
System Databases
SQLQuery1.sql - TR...ocs (ORT\lpan (52))*
100 %
Results
Messages

https://ca-dev-ws14:440/restservice/write/LilyTest

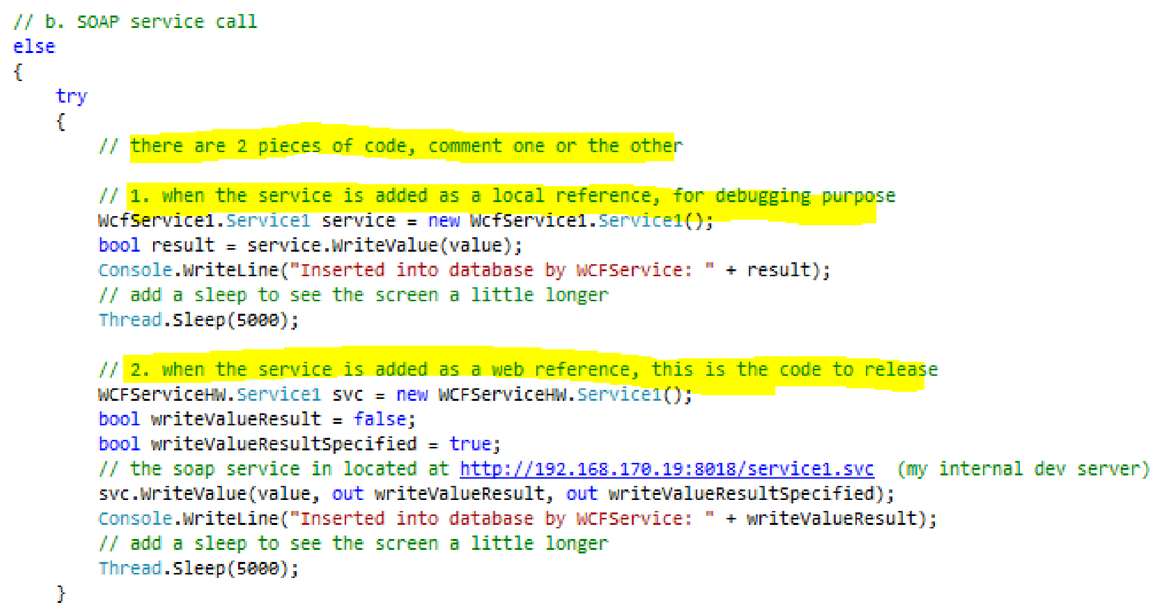
true

```
/****** Script for SelectTopNRows command
SELECT * FROM [tdocs].[dbo].[TestTable]
```

|    | ID | Value |
|----|----|-------|
| 87 | 89 | LilyTest |

Query executed successfully.

2. In HelloWorldUseAPI.cs, when the code calls WCF SAOP service, please comment one or the other, to try either the local reference (debug purpose) or the web reference (when service published on server).

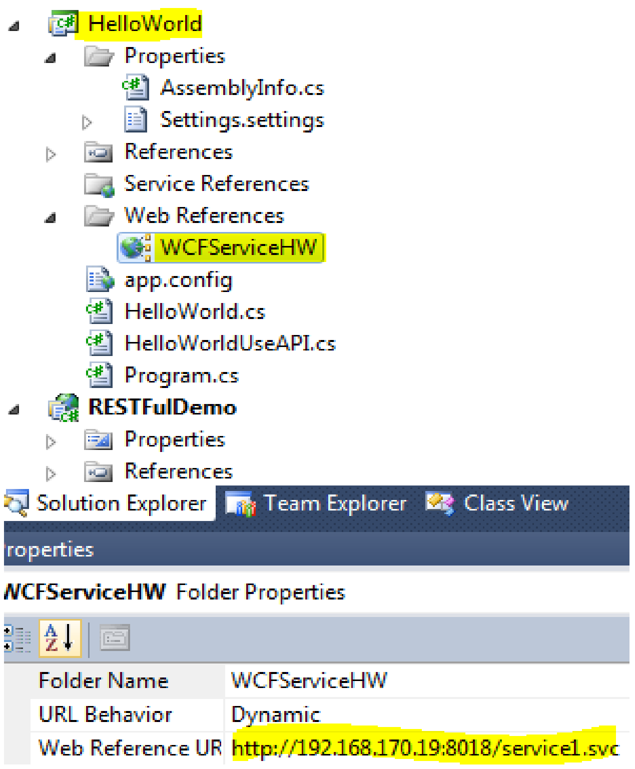


```
// b. SOAP service call
else
{
    try
    {
        // there are 2 pieces of code, comment one or the other

        // 1. when the service is added as a local reference, for debugging purpose
        WcfService1.Service1 service = new WcfService1.Service1();
        bool result = service.WriteValue(value);
        Console.WriteLine("Inserted into database by WCFService: " + result);
        // add a sleep to see the screen a little longer
        Thread.Sleep(5000);

        // 2. when the service is added as a web reference, this is the code to release
        WCFServiceHW.Service1 svc = new WCFServiceHW.Service1();
        bool writeValueResult = false;
        bool writeValueResultSpecified = true;
        // the soap service in located at http://192.168.170.19:8018/service1.svc  (my internal dev server)
        svc.WriteValue(value, out writeValueResult, out writeValueResultSpecified);
        Console.WriteLine("Inserted into database by WCFService: " + writeValueResult);
        // add a sleep to see the screen a little longer
        Thread.Sleep(5000);
    }
```

When you try the web reference, please not to use my web reference. Because I published it on my internal dev server which will not be connected from outside.

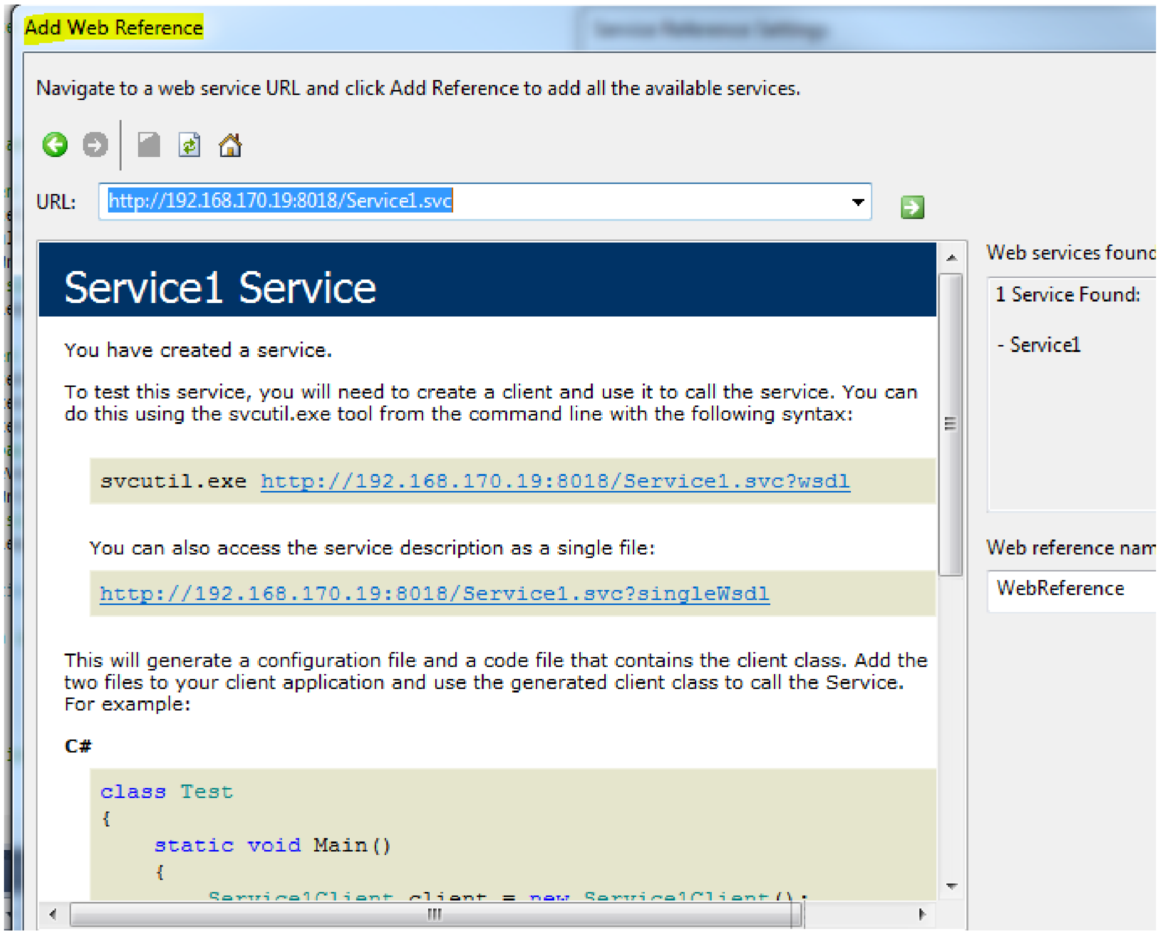


```
◢ HelloWorld
    ◢ Properties
          AssemblyInfo.cs
      ▷   Settings.settings
    ▷ References
      Service References
    ◢ Web References
          WCFServiceHW
      app.config
      HelloWorld.cs
      HelloWorldUseAPI.cs
      Program.cs
◢ RESTFulDemo
    ▷ Properties
    ▷ References
```

**NCFServiceHW** Folder Properties

| | |
|---|---|
| Folder Name | WCFServiceHW |
| URL Behavior | Dynamic |
| Web Reference UR | http://192.168.170.19:8018/service1.svc |

Please publish the WCF SOAP service on your server, and then go to "Add Web Reference" with your own service link.


Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
URL: http://192.168.170.19:8018/Service1.svc
Web services found
1 Service Found:
- Service1
Web reference name
WebReference

Service1 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://192.168.170.19:8018/Service1.svc?wsdl
```

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

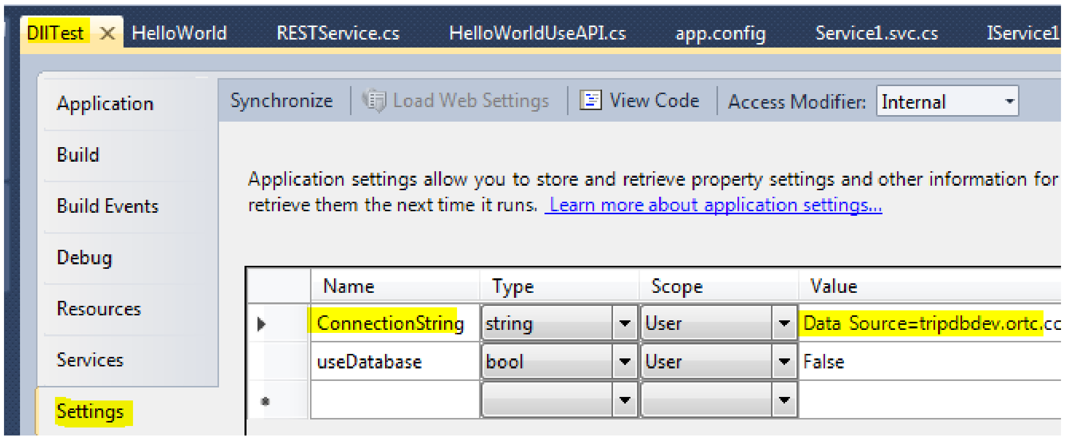
```
http://192.168.170.19:8018/Service1.svc?singleWsdl
```

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()
    {
        Service1Client client = new Service1Client();
```

3. The connection string for the DLL is set in here:
I will delete my connection string value, so you will put your own value in there.

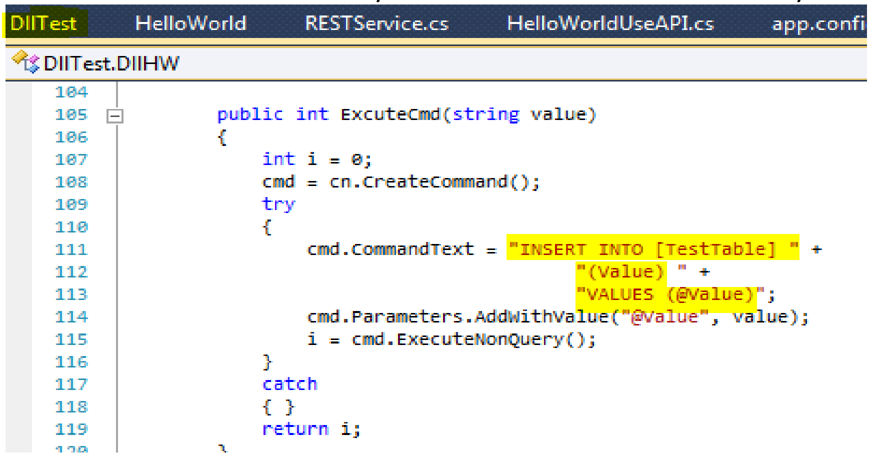


Access Modifier: Internal

Application settings allow you to store and retrieve property settings and other information for
retrieve them the next time it runs. Learn more about application settings...

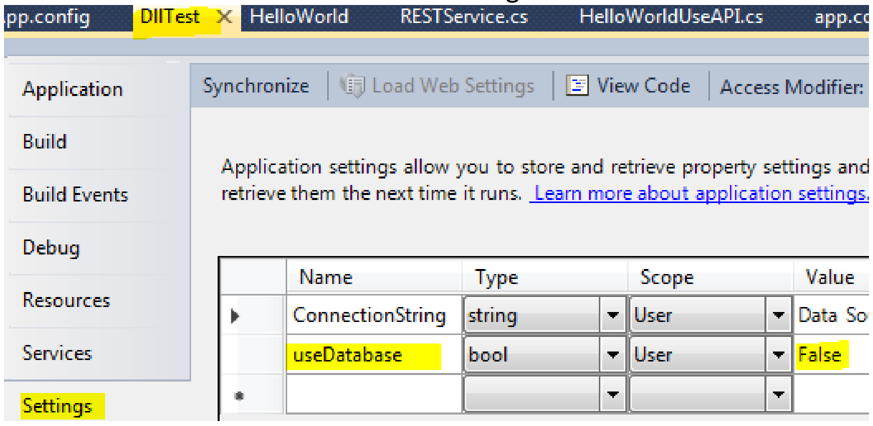
| Name | Type | Scope | Value |
|---|---|---|---|
| ConnectionString | string | User | Data Source=tripdbdev.ortc.cc |
| useDatabase | bool | User | False |
| | | | |

In DllHW.cs, if you would like to test the database insert method, this is where you put your own query.
I have created a TestTable in my dev database and it works with my code.



```
        public int ExcuteCmd(string value)
        {
            int i = 0;
            cmd = cn.CreateCommand();
            try
            {
                cmd.CommandText = "INSERT INTO [TestTable] " +
                                  "(Value) " +
                                  "VALUES (@Value)";
                cmd.Parameters.AddWithValue("@Value", value);
                i = cmd.ExecuteNonQuery();
            }
            catch
            { }
            return i;
```

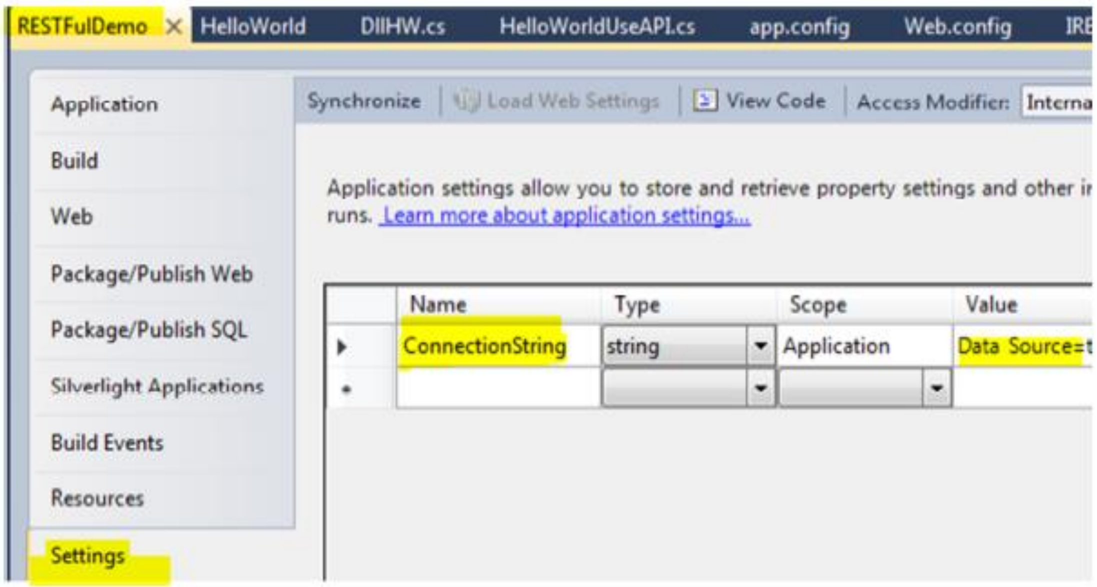
In the DLL I also added GetValue( ) method so we have options of either getting
value from the database or just hard coding “Hello World” for now. GetValue( ) is turned off
now and it could be turned on in the config file: useDatabase.



Application settings allow you to store and retrieve property settings and
retrieve them the next time it runs. Learn more about application settings.

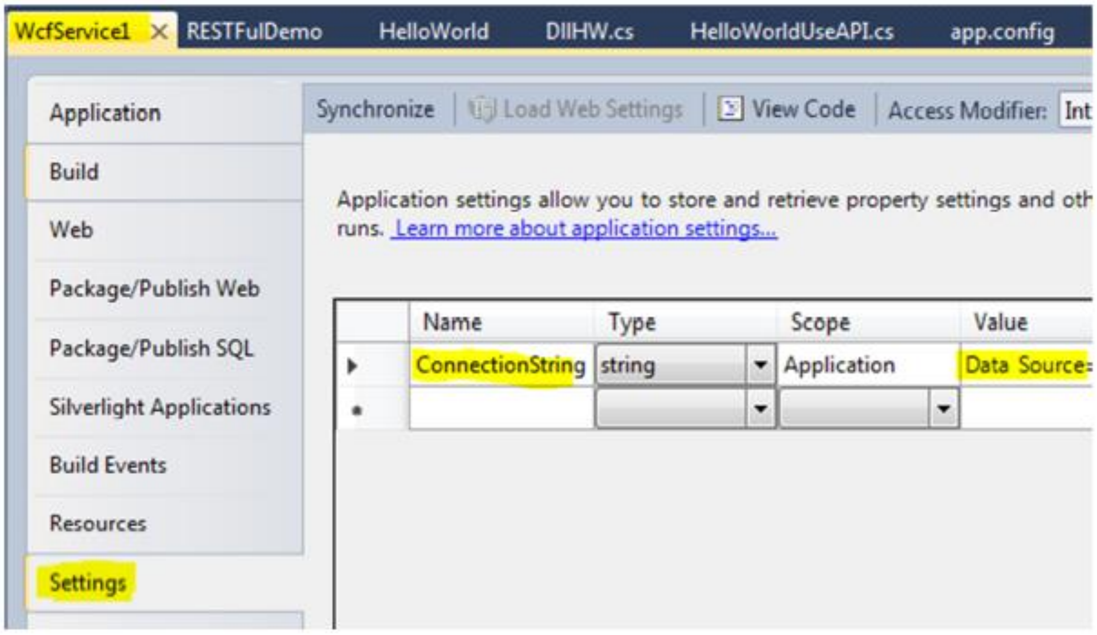
| Name | Type | Scope | Value |
|---|---|---|---|
| ConnectionString | string | User | Data So |
| useDatabase | bool | User | False |
| | | | |

4. I will also delete all connection strings in RESTFulDemo and in WcfService1. Please add your own connection strings in there if you will test the code.



Application settings allow you to store and retrieve property settings and other i
runs. Learn more about application settings...

| | Name | Type | Scope | Value |
|---|---|---|---|---|
| ▸ | ConnectionString | string | Application | Data Source=t |
| * | | | | |



Application settings allow you to store and retrieve property settings and oth
runs. Learn more about application settings...

| | Name | Type | Scope | Value |
|---|---|---|---|---|
| ▸ | ConnectionString | string | Application | Data Source= |
| * | | | | |

5. I did unit test for the DLL, to check if the return value is “Hello World”, and the database insert method.

6. Interface, abstract class are used in the code.

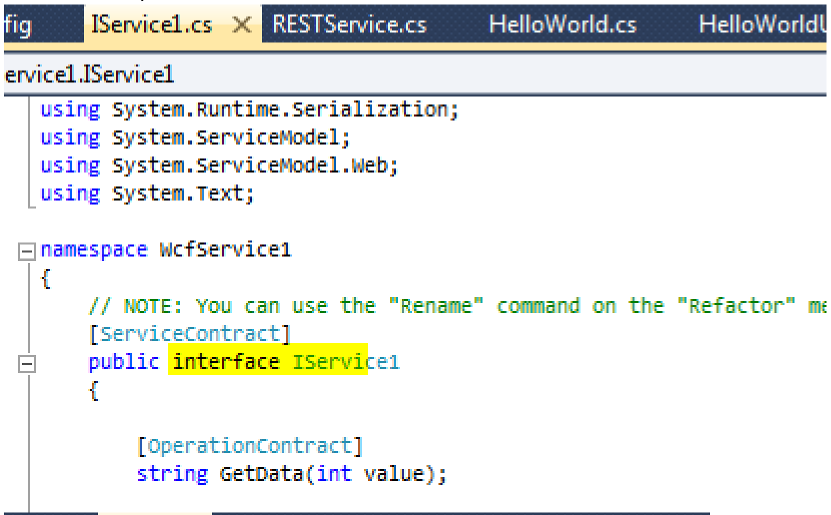


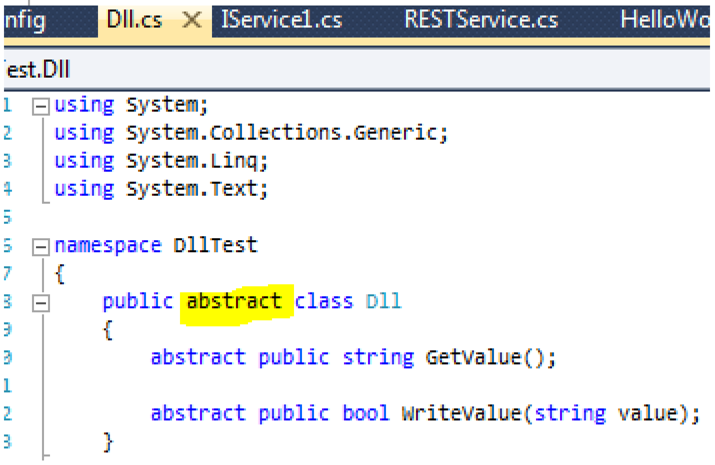
fig | IService1.cs | RESTService.cs | HelloWorld.cs | HelloWorldU

ervice1.IService1

```
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfService1
{
    // NOTE: You can use the "Rename" command on the "Refactor" me
    [ServiceContract]
    public interface IService1
    {

        [OperationContract]
        string GetData(int value);
```



nfig | Dll.cs | IService1.cs | RESTService.cs | HelloWo

est.Dll

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace DllTest
{
    public abstract class Dll
    {
        abstract public string GetValue();

        abstract public bool WriteValue(string value);
    }
```

Conclusion:

The DLL is easy to use, but it has to be added locally in each application, and if there are changes in the DLL then we will have to rebuild the DLL, also all applications that use the DLL will have to re-reference the DLL and rebuild too. The SOAP service can be consumed from the server on internet, and it’s stable, but if there are changes in the Service Contract of the service, the client app will also have to re-reference the service and rebuild the code. Also we have to make sure all the configurations match on the server side and on client side, such as endpoints, binding etc., otherwise it will not work. Especially when SSL is enabled on the server, the configurations could be a little complicated for the client side. The REST service is easier for client apps to use especially easy for mobil apps, because there is no reference required, if there are changes in the service, as long as the uri is not changed, the client side doesn’t have to do make any changes.