Getting Started Exercises for WCF session01

1. Remote Hello World Exercise:

In this exercise we will step by step go through the creation of a very simple distributed multi-tier application using WCF.

Content:

1. Create the back-end (HelloServer).
2. Implement a service (WcfHelloService) that exposes the back-end
    a. Declare service interface
    b. Implement the interface in a class
3. Implement the client (HelloClient).

Create the back-end

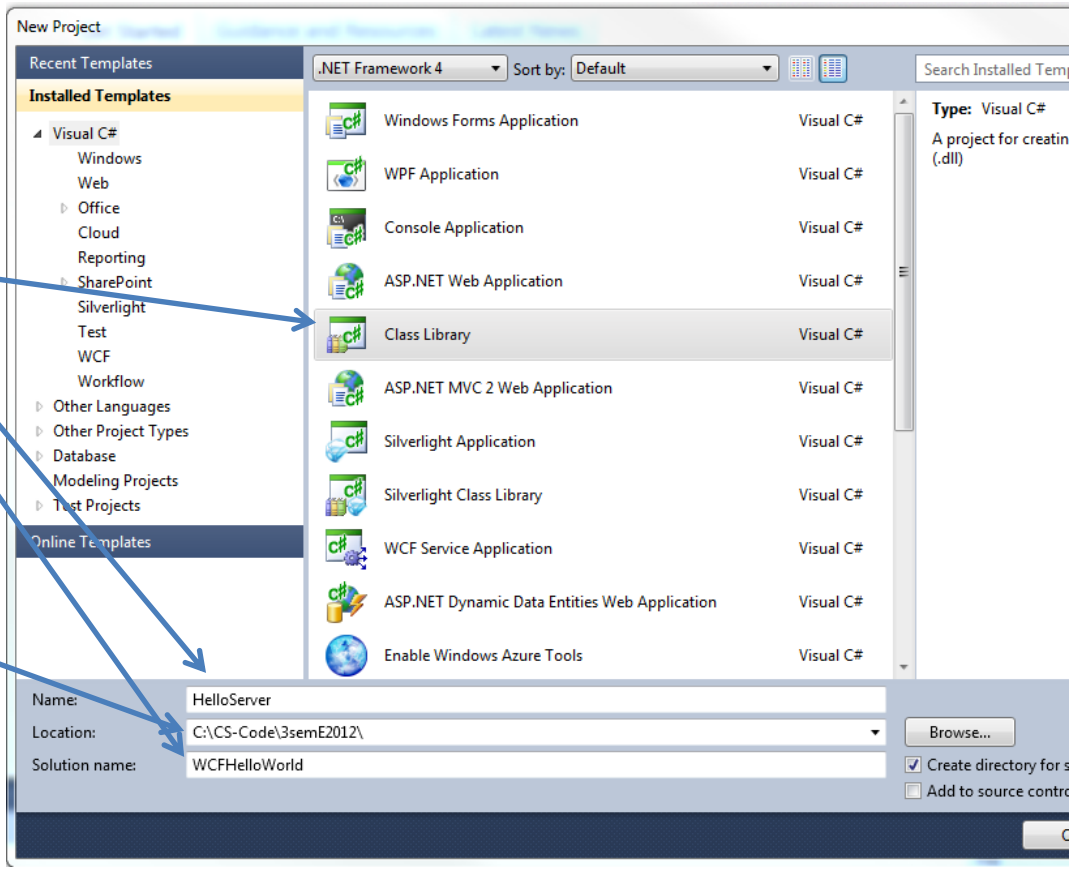
We will do it the right way: Create a Hello interface and an implementing class:

Open VS and create a new project and solution:

Choose “Class Library”.

Pick a good name for the project and the solution (there will come more projects, so don’t pick the same).

Pick a location.



And GO:

Remove the auto-generated “Class1.cs” and add an interface: IHelloWorld:

```
public interface IHelloWorld
{
    string SayHello();
}
```

Add a class that implements the interface:

```
public class HelloWorld: IHelloWorld
{
    public string SayHello()
    {
        return "Hello Remote World!";
    }
}
```

Your back-end is now ready.

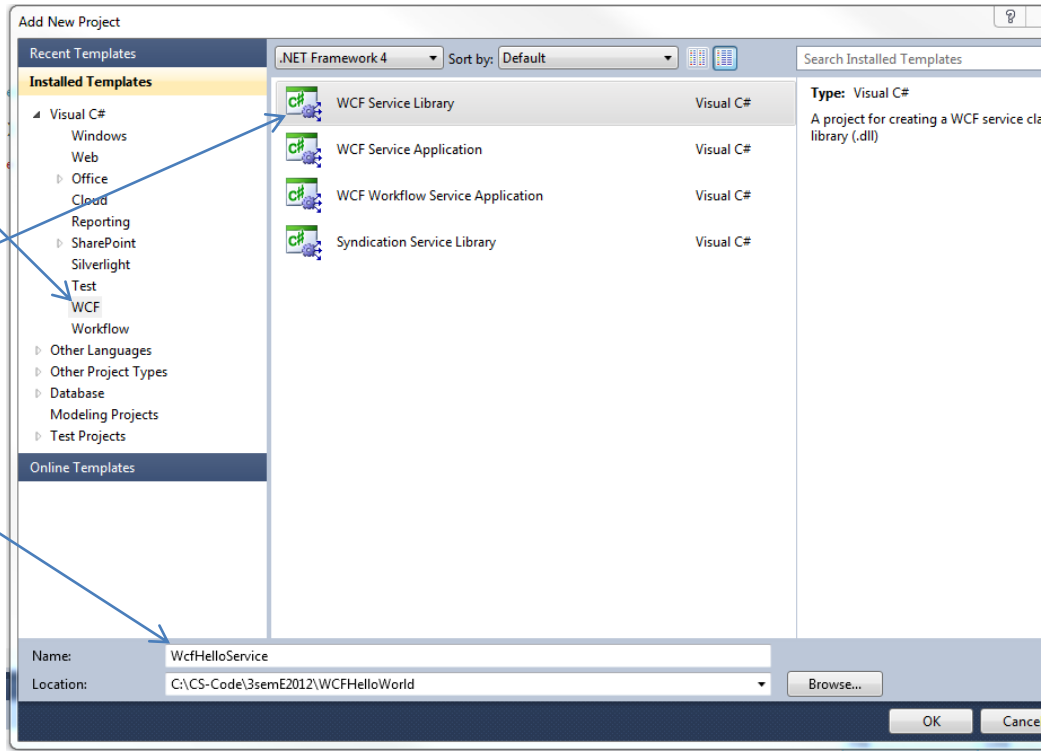
Create the Web-service

Right-click on the solution and choose “Add -> New Project”:

Choose WCF templates.

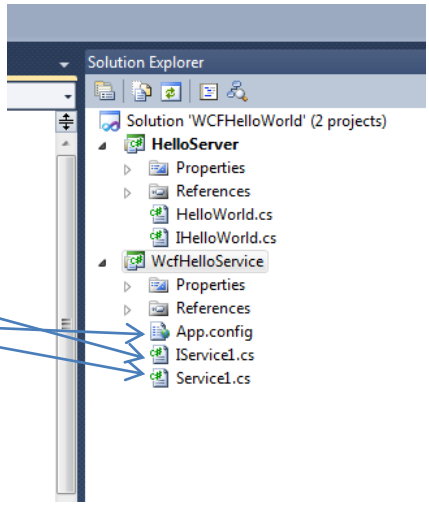
Choose “WCF Service Library”.

Give the project a suitable name.



And GO:

Now your solution should look like this:



An interface,

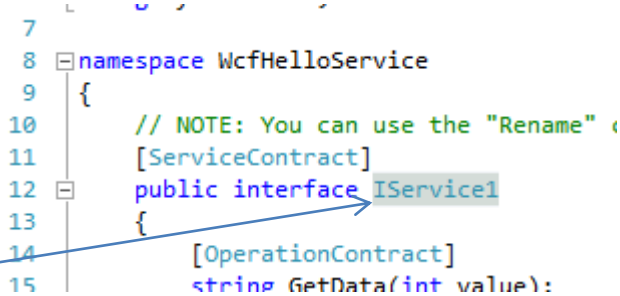
a class

and a config file

are added

and a lot of code is generated in “IService1.cs” and

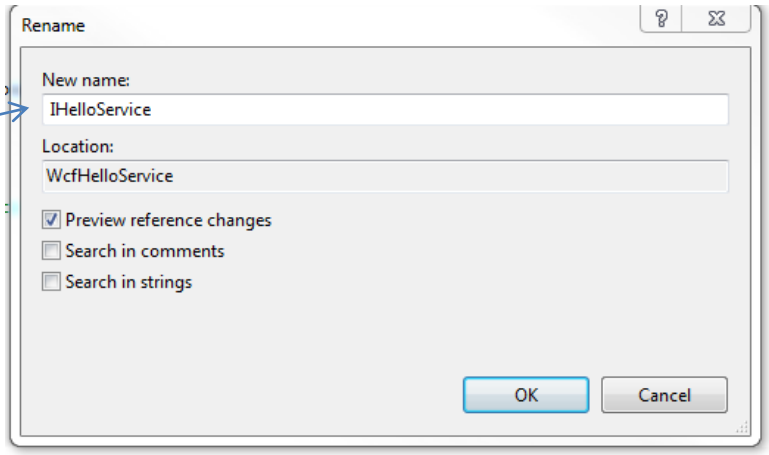
“Service1.cs”.



Mark the interface name in the code.

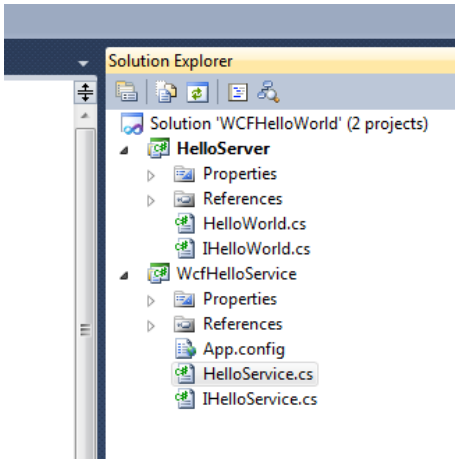
Choose “Refactor -> Rename”.

Give the interface a good name.



Do the same with the class Service1. Now the names are changed both in the code and in the App.config file. You may want to rename the files as well: Do this by right-clicking on the file in the solutions explorer and chose “Rename”. Be careful: Don’t rename the App.config file. That will confuse VS!

Now your solution should look like this:



Next step is to delete all the generated code

in the interface.

Instead this code should be inserted:

```
[ServiceContract]
public interface IHelloService
{
    [OperationContract]
    string SayHello();
}
```

Note that we define a service contract stating that this interface is an interface for a WCF service and an operation contract stating that this method may be called remote.

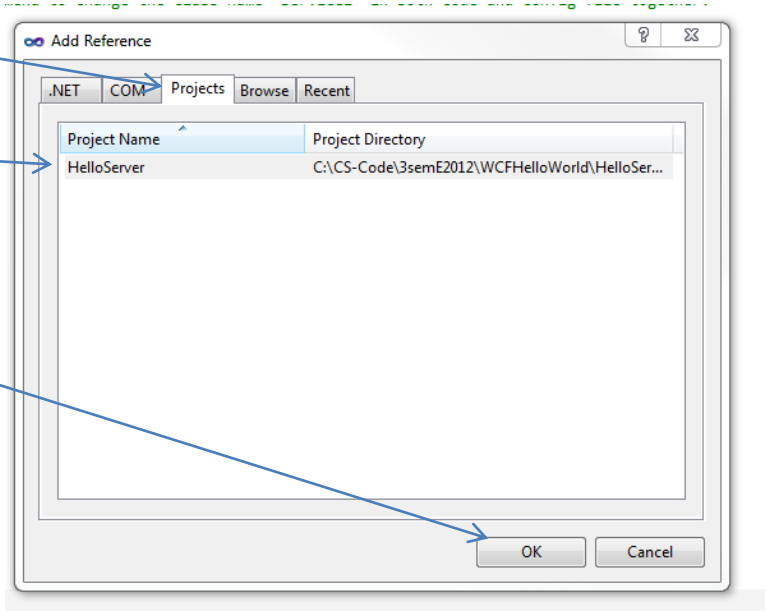
We are now going to implement the interface with a class that calls our back-end – this class becomes a proxy for the back-end. In order to do this, we need to add a reference from the WcfHelloService project to the back-end (the HelloWorld project) and a using declaration in the top of the code:

Right-click on the WcfHelloService project and choose: “Add Reference”:

Choose “Projects”

and the one you want

and “OK”.



Remember the using:

```
using HelloServer;
```

Your WCF service may now be implemented using the code in the back-end:

Using the back-end:

```
public class HelloService : IHelloService
{
    private static IHelloWorld helloObj = new HelloWorld();
    public string SayHello()
    {
        return helloObj.SayHello();
    }

}
```

Now your WCF service should be ready for use.

Create a client

As the first client add a new project (console application) to the solution. Call it HelloClient. Set the project as StartUp Project (right click on the project).

To that project we must add a reference to the WCF service: Right-click on the project and choose “Add Service Reference”:

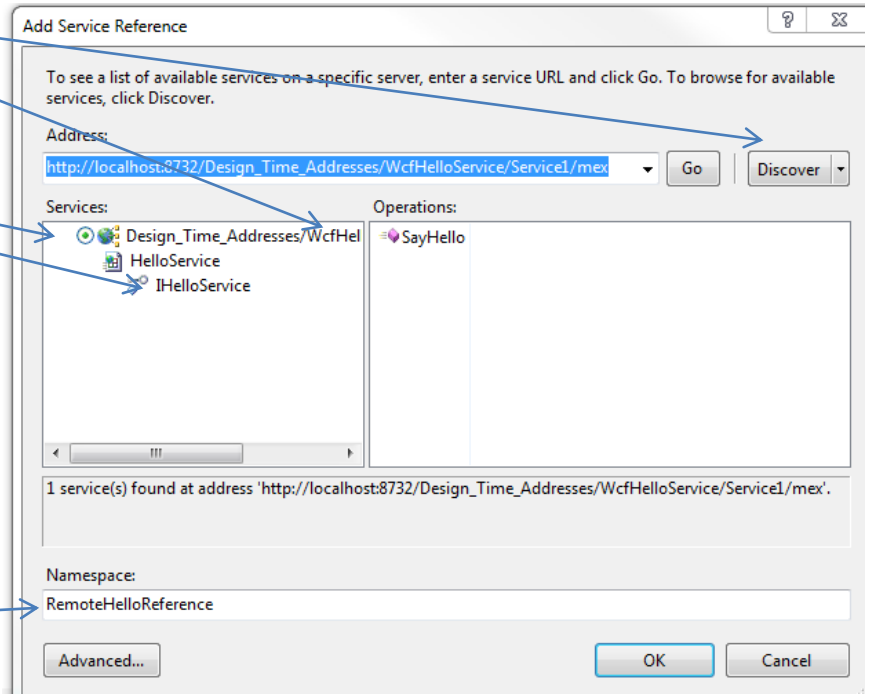
Click the “Discover” button:

Your WcfHelloService should appear.

Expand.

Pick the Service interface.

Give the reference namespace a good name.



In the back VS has generated a client side proxy for the WCF HelloService and implemented the necessary piping for communicating with the service.

So now you can implement the client using the proxy in the RemoteHelloReference namespace:

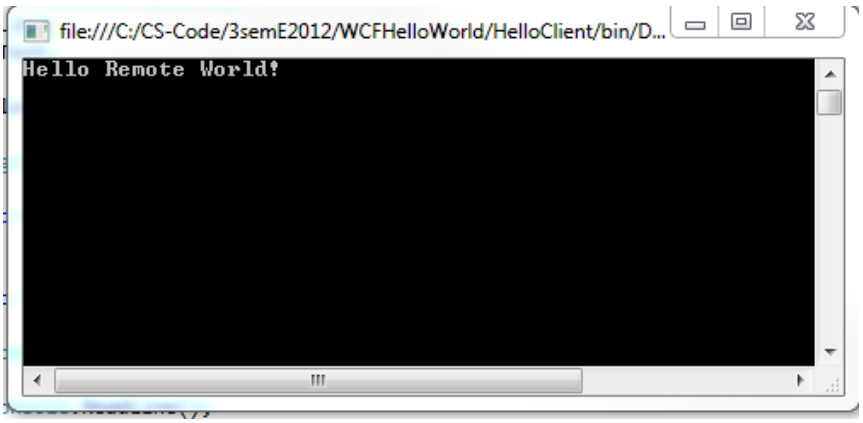
Here we reference the local (generated proxy):

Here we call the SayHello() method on the local proxy, which passes on the call to the remote WCF service, which again passes on the call to the backend and the answer is coming all the way back to the client again:

```
namespace HelloClient
{
    class Program
    {
        static RemoteHelloReference.IHelloService helloObj =
                    new RemoteHelloReference.HelloServiceClient();

        static void Main(string[] args)
        {
            Console.WriteLine(helloObj.SayHello());

            Console.ReadLine();
        }
    }
}
```



Fantastic!

2. Add another client

Create a WinForm application that says ´”Hello World” using the same WCF service. Go through the same steps as when the console application was created.

3 Make your own distributed application

Make a back-end (Calculator) with methods for adding, subtracting and multiplying two numbers in the same way as the HelloServer. Add a WCF service that uses the back-end. And finally, add a client (for instance a simple GUI).