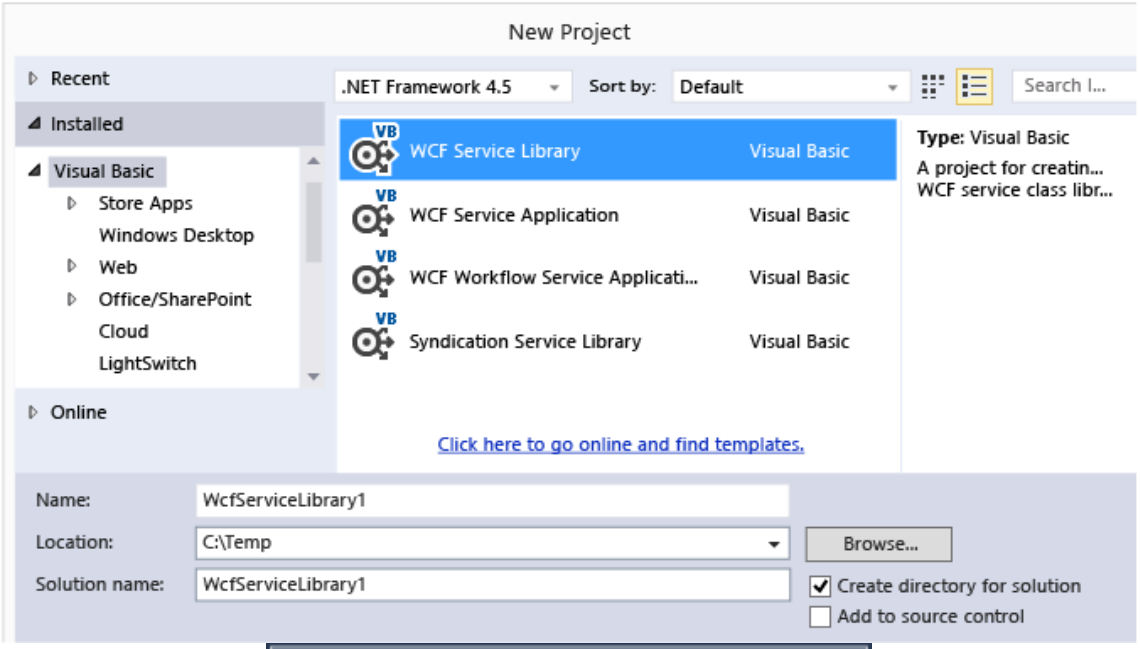
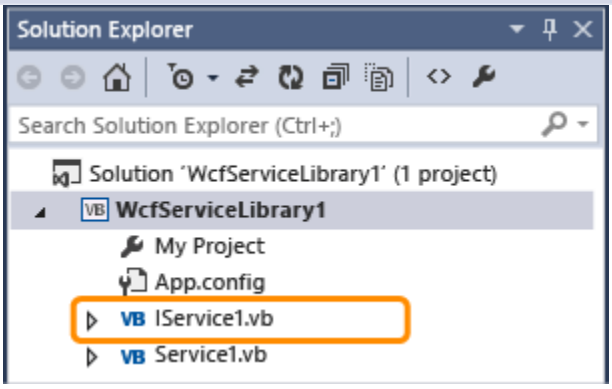
Crearea unui serviciu WCF

Pentru a crea un serviciu WCF urmati urmatorii pasi:

1. **File**-> **New-> Project**.
2. **New Project-> Visual C#-> WCF-> WCF Service Library**. Click **OK** pentru a deschide proiectul.





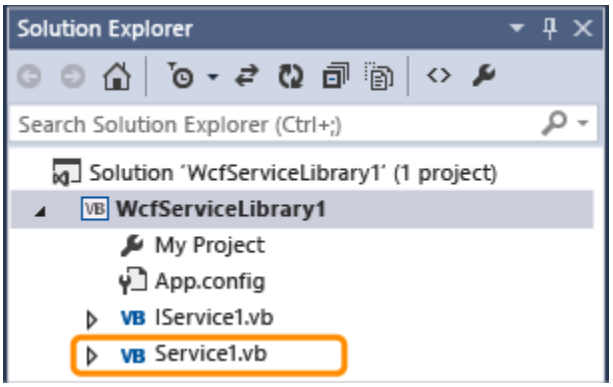
3. In **Solution Explorer**, dublu-click pe IService1.cs:

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

Schimbati tipul parametrului din int in string

```
string GetData(string value);
```

`[OperationContract]` este un atribut al metodei. Numai metodele decorate cu aceste attribute vor fi expuse de catre serviciu (accesibile din exterior)



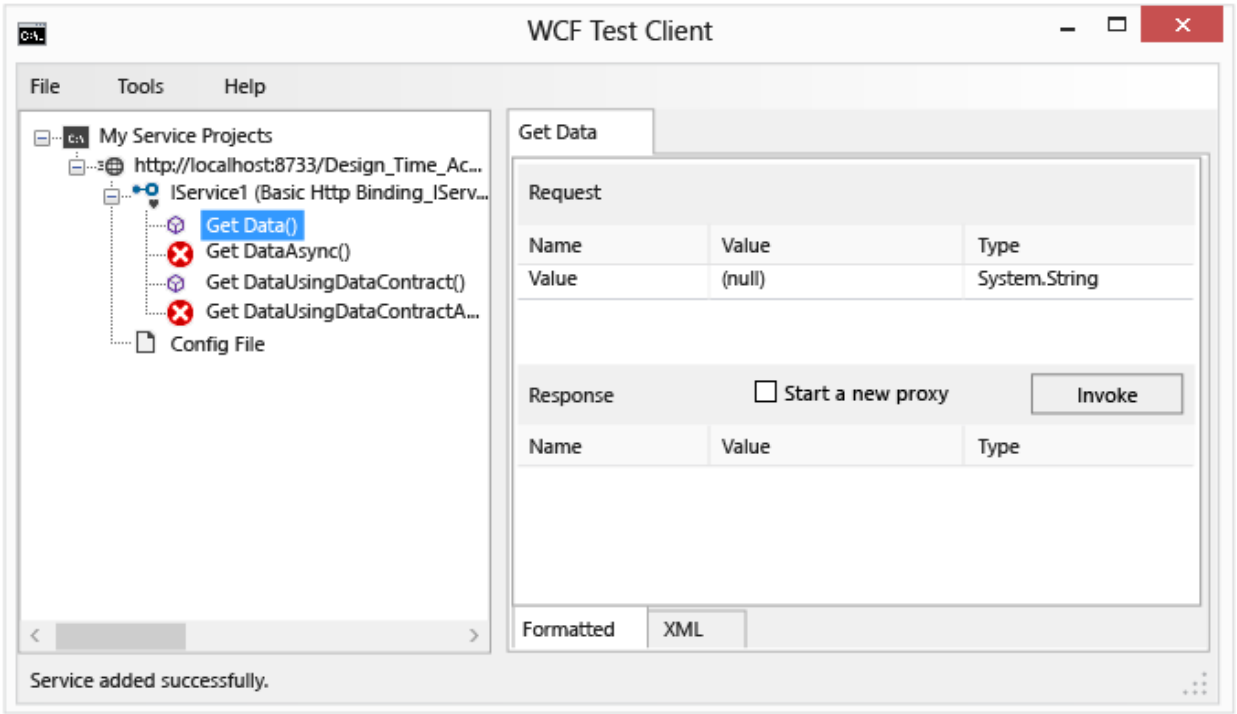
In **Solution Explorer**, dublu-click Service1.cs si modificati urmatoarea linie dupa cum este indicat mai jos:

```
        public string GetData(int value)
        {
            return string.Format("You entered: {0}", value);
        }
-------------------------------------------------------------------------
        public string GetData(string value)
        {
            return string.Format("You entered: {0}", value);
        }
```

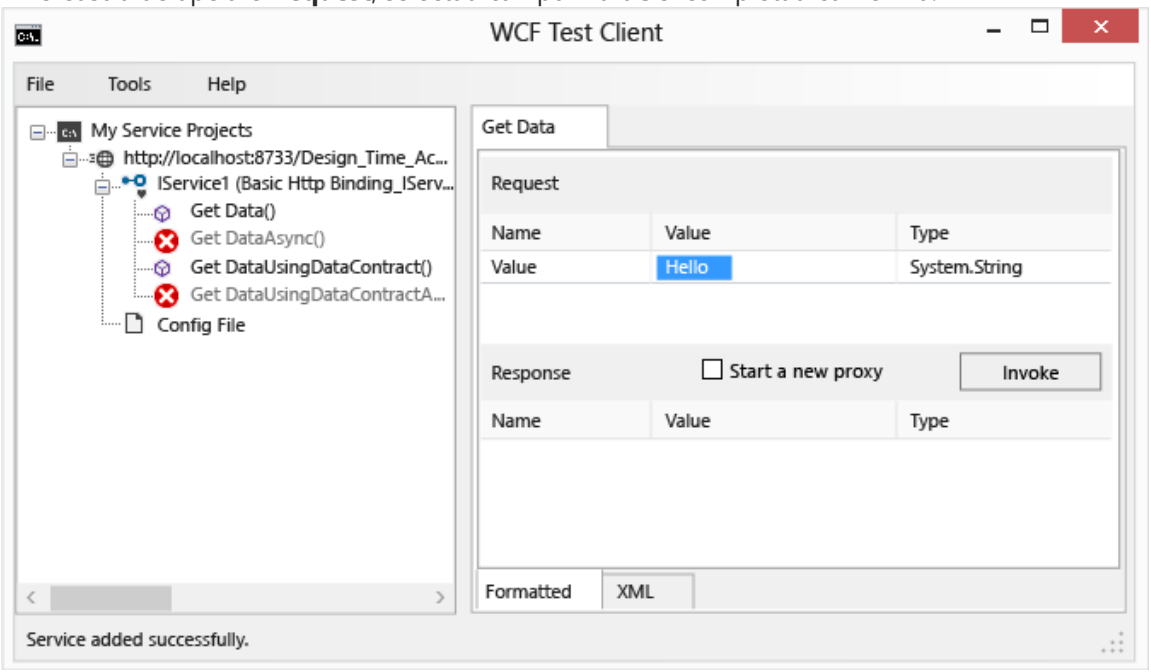
Testarea serviciului

Pentru a testa serviciul trebuie efectuati urmatorii pasi:

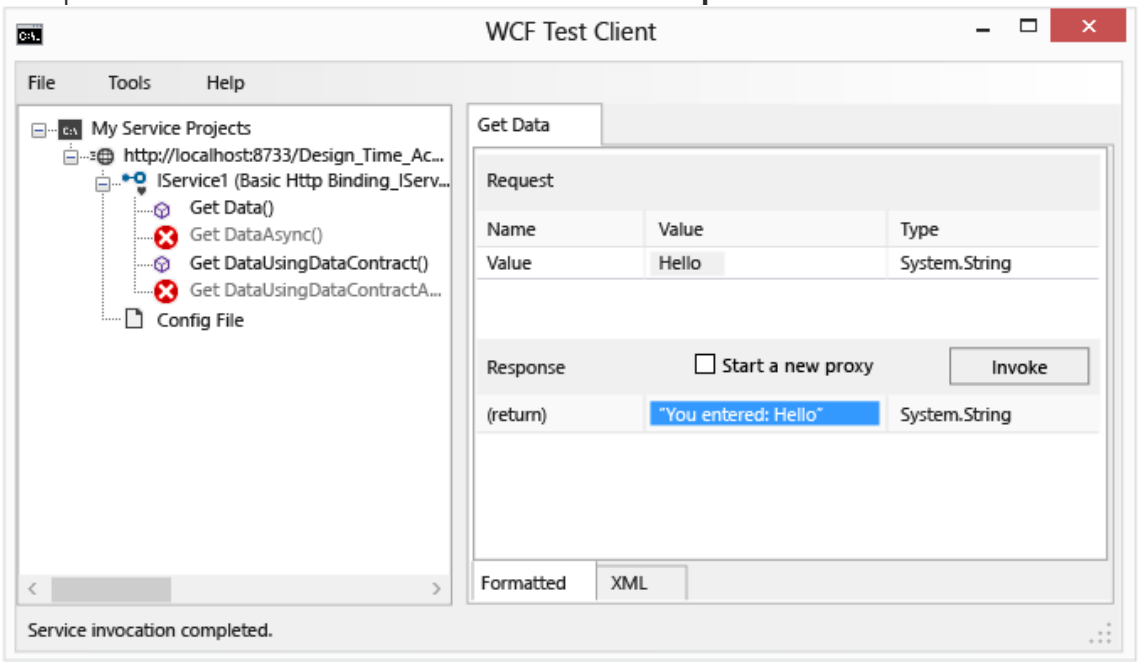
1. Apasati **F5** pentru a rula serviciul. Un client de test o sa se incarce in ecran, **WCF Test Client**.
2. In **WCF Test Client**, dublu-click pe metoda **GetData()** va deschide fereastra de apelare a metodei **GetData**.



3. In fereastra de apelare **Request**, selectati campul **Value** si completati cu `Hello`.



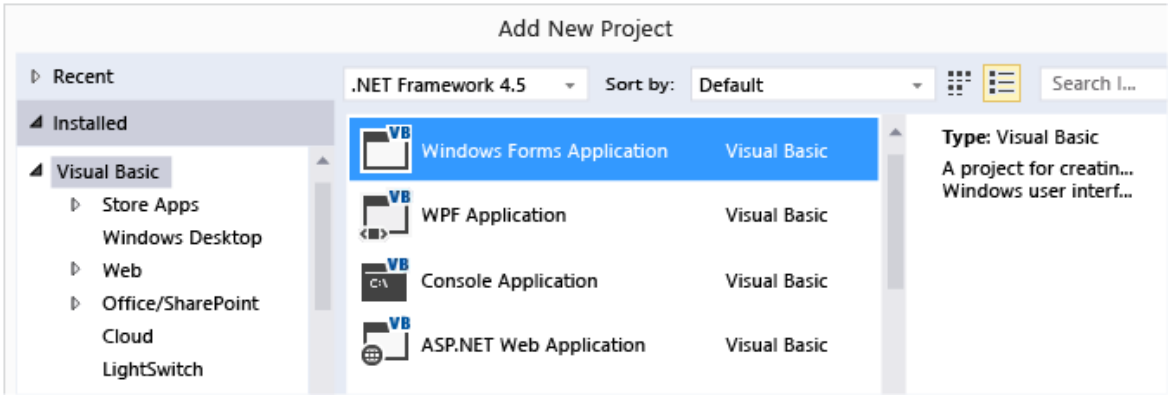
4. Click pe butonul **Invoke**. Rezultatul va fi afisat in casuta **Response**.



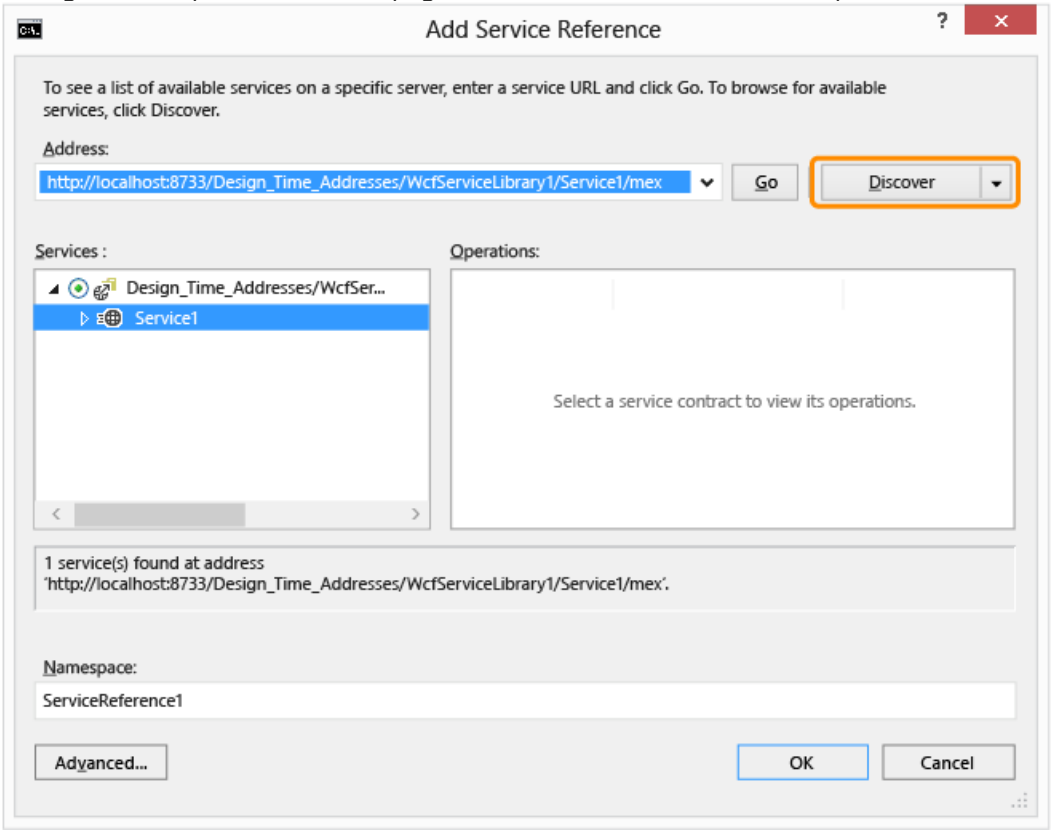
Accesarea serviciului

Pentru a crea o referinta catre serviciul crea trebuie sa executam urmatorii pasi:

1. **File-> Add-> New Project**.
2. In **New Project Visual C#** -> **Windows**,-> **Windows Forms Application -> OK**.



3. Click dreapta pe **WindowsApplication1** si click pe optiunea **Add Service Reference**.
4. Adaugati in campul adresa url-ul paginii unde este deschis serviciul si apoi click **Discover**.

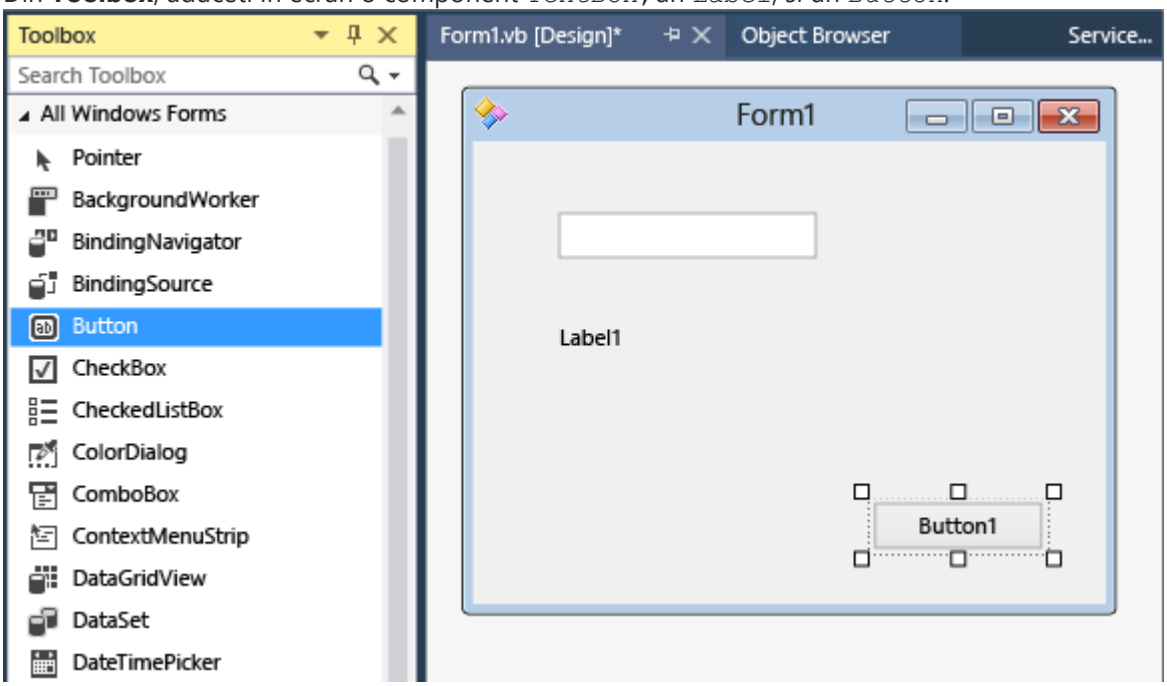


5. Click **OK** pentru a adauga referinta.

Construirea aplicatie client

1. In **Solution Explorer**, dublu-click **Form1.cs**

2. Din **Toolbox**, aduceti in ecran o component `TextBox`, un `Label`, si un `Button`.

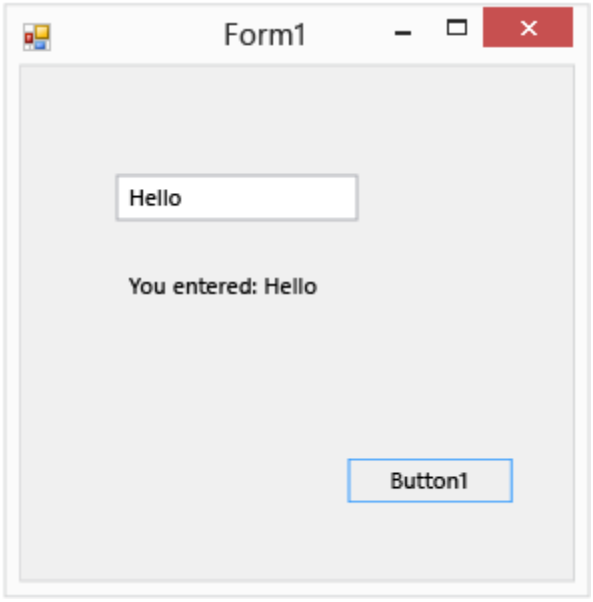


3. Duble-click pe `Button`, si adaugati urmatorul cod in interiorul evenimentului creat:

```
        private void button1_Click(System.Object sender,
System.EventArgs e)
        {
            ServiceReference1.Service1Client client = new
                ServiceReference1.Service1Client();
            string returnString;

            returnString = client.GetData(textBox1.Text);
            label1.Text = returnString;
        }
```

4. In **Solution Explorer**, click dreapta pe **WindowsApplication1** si click **Set as StartUp Project**.
5. Apasati **F5** pentru a rula aplicatia.


Form1
Hello
You entered: Hello
Button1

Hostarea unui serviciu WCF intr-un Managed Windows Service

Construirea serviciului

1. Creati o aplicatie consola numita "Service".
2. Redenumiti Program.cs -> Service.cs.
3. Schimbati namespace-ul in Microsoft.ServiceModel.Samples.
4. Adaugati urmatoarele referinte.
   - System.ServiceModel.dll
   - System.ServiceProcess.dll
   - System.Configuration.Install.dll
5. Adaugati urmatoarele comenzi in Service.cs.

   using System.ComponentModel;

   using System.ServiceModel;

   using System.ServiceProcess;

   using System.Configuration;

   using System.Configuration.Install;

6. Definiti o interfata numita `ICalculator`:

```
// Define a service contract.
[ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples")]
public interface ICalculator
{
    [OperationContract]
    double Add(double n1, double n2);
    [OperationContract]
    double Subtract(double n1, double n2);
    [OperationContract]
    double Multiply(double n1, double n2);
    [OperationContract]
    double Divide(double n1, double n2);

}
```

7. Creati o clasa `CalculatorService` care implementeaza interfata creata anterior dupa cum urmeaza:

```
// Implement the ICalculator service contract in a service class.
public class CalculatorService : ICalculator
{
    // Implement the ICalculator methods.
    public double Add(double n1, double n2)
    {
        double result = n1 + n2;
        return result;
    }

    public double Subtract(double n1, double n2)
    {
        double result = n1 - n2;
        return result;
    }

    public double Multiply(double n1, double n2)
    {
        double result = n1 * n2;
        return result;
    }

    public double Divide(double n1, double n2)
    {
        double result = n1 / n2;
        return result;
    }

     }
```

8. Creati o clasa numita `CalculatorWindowsService` care mosteneste clasa ServiceBase. Adaugati o variabila de tip Servicehost numita `serviceHost` Definiti o metoda `Main` care apeleaza `ServiceBase.Run(new CalculatorWindowsService)`

```
public class CalculatorWindowsService : ServiceBase
{
    public ServiceHost serviceHost = null;
    public CalculatorWindowsService()
    {
        // Name the Windows Service
        ServiceName = "WCFWindowsServiceSample";
    }
    public static void Main()
    {
        ServiceBase.Run(new CalculatorWindowsService());
```

```
    }
```

9. Override the OnStart(String[]) method by creating and opening a new ServiceHost instance as shown in the following code.

```
// Start the Windows service.
protected override void OnStart(string[] args)
{
    if (serviceHost != null)
    {
        serviceHost.Close();
    }

    // Create a ServiceHost for the CalculatorService type and
    // provide the base address.
    serviceHost = new ServiceHost(typeof(CalculatorService));

    // Open the ServiceHostBase to create listeners and start
    // listening for messages.
    serviceHost.Open();

     }
```

10. Override the OnStop method closing the ServiceHost as shown in the following code.

```
protected override void OnStop()
{
    if (serviceHost != null)
    {
        serviceHost.Close();
        serviceHost = null;
    }

     }
```

11. Creati o noua clasa `ProjectInstaller` care mosteneste Installer si care este decorata cu atributul RunInstallerAttribute setat la `true`. Aceasta va permite ca serviciul Windows sa se instaleze cu Installutil.exe tool.

```
// Provide the ProjectInstaller class which allows
// the service to be installed by the Installutil.exe tool
[RunInstaller(true)]
public class ProjectInstaller : Installer
{
    private ServiceProcessInstaller process;
    private ServiceInstaller service;

    public ProjectInstaller()
    {
        process = new ServiceProcessInstaller();
        process.Account = ServiceAccount.LocalSystem;
        service = new ServiceInstaller();
        service.ServiceName = "WCFWindowsServiceSample";
        Installers.Add(process);
        Installers.Add(service);
    }

     }
```

12. Stergeti clasa `Service` care a fos generata cand ati creat proiectul.
13. Adaugati in proiect un fisier de configurare si inlocuiti tot ce este prezent in el cu codul de mai jos.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>    <services>
      <!-- This section is optional with the new configuration model
           introduced in .NET Framework 4. -->
      <service name="Microsoft.ServiceModel.Samples.CalculatorService"
               behaviorConfiguration="CalculatorServiceBehavior">
        <host>
          <baseAddresses>
            <add
baseAddress="http://localhost:8000/ServiceModelSamples/service"/>
          </baseAddresses>
        </host>
        <!-- this endpoint is exposed at the base address provided by host:
http://localhost:8000/ServiceModelSamples/service  -->
        <endpoint address=""
                  binding="wsHttpBinding"
                  contract="Microsoft.ServiceModel.Samples.ICalculator" />
        <!-- the mex endpoint is exposed at
http://localhost:8000/ServiceModelSamples/service/mex -->
        <endpoint address="mex"
                  binding="mexHttpBinding"
```

```
                contract="IMetadataExchange" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="CalculatorServiceBehavior">
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="False"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>

    </configuration>
```

Click dreapta pe App.config si selectati **Properties**. La optiunea **Copy to Output Directory** selectati **Copy if Newer**.

Instalarea serviciului si rularea lui

1. Build-uiti Solutia pentru a crea executabilul `Service.exe`.
2. Deschideti consola comanda a Visual Studio 20xx si navigate la directorul proiectului. Tastati mai apoi `installutil service.exe` pentru a instala serviciul.

   Pentru a vizualiza serviciul tastati in cmd `services.msc` Pentru a porni serviciul fie il startati din fereastra de servicii fie din command prompt ruland comanda **net start WCFWindowsServiceSample**.

3. Pentru a dezinstala serviciul folositi comanda **installutil /u service.exe**

Examplul de mai jos este tot codul explicat in cele de mai sus:

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

using System.ComponentModel;
using System.ServiceModel;
using System.ServiceProcess;
using System.Configuration;
using System.Configuration.Install;

namespace Microsoft.ServiceModel.Samples
{
    // Define a service contract.
```

```
    [ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples")]
    public interface ICalculator
    {
        [OperationContract]
        double Add(double n1, double n2);
        [OperationContract]
        double Subtract(double n1, double n2);
        [OperationContract]
        double Multiply(double n1, double n2);
        [OperationContract]
        double Divide(double n1, double n2);
    }

    // Implement the ICalculator service contract in a service class.
    public class CalculatorService : ICalculator
    {
        // Implement the ICalculator methods.
        public double Add(double n1, double n2)
        {
            double result = n1 + n2;
            return result;
        }

        public double Subtract(double n1, double n2)
        {
            double result = n1 - n2;
            return result;
        }

        public double Multiply(double n1, double n2)
        {
            double result = n1 * n2;
            return result;
        }

        public double Divide(double n1, double n2)
        {
            double result = n1 / n2;
            return result;
        }
    }

    public class CalculatorWindowsService : ServiceBase
    {
        public ServiceHost serviceHost = null;
        public CalculatorWindowsService()
        {
            // Name the Windows Service
            ServiceName = "WCFWindowsServiceSample";
        }

        public static void Main()
```

```
        {
            ServiceBase.Run(new CalculatorWindowsService());
        }

        // Start the Windows service.
        protected override void OnStart(string[] args)
        {
            if (serviceHost != null)
            {
                serviceHost.Close();
            }

            // Create a ServiceHost for the CalculatorService type and
            // provide the base address.
            serviceHost = new ServiceHost(typeof(CalculatorService));

            // Open the ServiceHostBase to create listeners and start
            // listening for messages.
            serviceHost.Open();
        }

        protected override void OnStop()
        {
            if (serviceHost != null)
            {
                serviceHost.Close();
                serviceHost = null;
            }
        }
    }

    // Provide the ProjectInstaller class which allows
    // the service to be installed by the Installutil.exe tool
    [RunInstaller(true)]
    public class ProjectInstaller : Installer
    {
        private ServiceProcessInstaller process;
        private ServiceInstaller service;

        public ProjectInstaller()
        {
            process = new ServiceProcessInstaller();
            process.Account = ServiceAccount.LocalSystem;
            service = new ServiceInstaller();
            service.ServiceName = "WCFWindowsServiceSample";
            Installers.Add(process);
            Installers.Add(service);
        }
    }
}
```

Like the "Self-Hosting" option, the Windows service hosting environment requires that some hosting code be written as part of the application. The service is implemented as a console application and contains its own hosting code. In other hosting environments, such as Windows Process Activation Service (WAS) hosting in Internet Information Services (IIS), it is not necessary for developers to write hosting code.

https://msdn.microsoft.com/en-us/library/bb386386.aspx