

Лабораторна робота 4

Розробка WCF-сервісу

Мета роботи: Розробка WCF сервісу для платформ Windows та Linux.

Використання WCF передбачає наявність трьох взаємозв'язаних компонентів:

1. Збірка служби (сервісу). Ця бібліотека класів .NET містить множину контрактів WCF та їх реалізацій. Контрактні інтерфейси оснащені різноманітними атрибутами, які визначають представлення типів, порядок їх взаємодії з середовищем WCF тощо. Класи та інтерфейси збірки сервісу забезпечують функціональність, що надається клієнтам.
2. Хост сервісу WCF. Це застосунок, який надає клієнтам доступ до збірки сервісу. Хост може бути застосунком довільного типу (Windows Forms, служба Windows, застосунок WPF тощо). При побудові спеціального хоста застосовується тип ServiceHost та зв'язаний з ним файл *.config.
3. Клієнт WCF. Це застосунок довільного типу, який звертається до функціональності сервісу через проміжний проксі. Налаштування взаємодії клієнта з сервісом здійснюється або програмно – у коді, або за допомогою конфігураційних файлів *.config. Якщо служба WCF використовує прив'язки на основі HTTP, то клієнт може бути реалізований на іншій платформі (наприклад, Java).

Розглянемо такі приклади створення служб (сервісів) WCF:

1. хостинг - спеціальний (консольна програма), інструмент - Visual Studio, цільова платформа – Windows;
2. хостинг - спеціальний (консольна програма), інструмент - Xamarin Studio, цільові платформи – Windows та Linux (.Net/ та Mono);
3. хостинг – керовані служби Windows, інструмент - Visual Studio, цільова платформа – Windows.

Розглянемо приклад (Приклади 1 та 2) створення служби (сервісу) WCF з назвою MagicBall, яка буде виконувати роль Магічної Кулі, а саме – робити жартівливе передбачення майбутнього (Приклади 1 і 2), а також сервісу арифметичних операцій (Приклад 3) .

Порядок виконання роботи

Приклад 1

1. Створити службу WCF як проект C# типу Class Library з ім'ям MagicBallServiceLib.

Перейменувати файл Class1.cs на MagicBallService.cs.
Додати посилання на збірку System.ServiceModel.dll.
Вказати використання простору імен System.ServiceModel.
Додати до проекту інтерфейс IMagicBall.

Вихідний код IMagicBall.cs

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

namespace MagicBallServiceLib
{
    [ServiceContract(Namespace = "http://localhost")] //MyCompany.com
    public interface IMagicBall
    {
        // Отримати відповідь на питання!
        [OperationContract]
        string ObtainAnswerToQuestion(string userQuestion);
    }
}
```

Вихідний код MagicBallService.cs

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

namespace MagicBallServiceLib
{
    public class MagicBallService : IMagicBall
    {
        public MagicBallService()
        {
            Console.WriteLine("Магічна куля відповідає...");
        }
        public string ObtainAnswerToQuestion(string userQuestion)
        {
            string[] answers = { "Майбутнє непевне", "Так", "Ні",
                                "Може статись", "Запитайте пізніше", "Безумовно" };

            // Return a random response.
            Random r = new Random();
            return answers[r.Next(answers.Length)];
        }
    }
}
```

Побудувати збірку сервісу MagicBallService.

2. Створити спеціальний хост служби WCF як консольний проект C# з ім'ям MagicBallServiceHost.

Додати посилання на збірки System.ServiceModel.dll та MagicBallServiceLib.dll, а також додати до коду імпорт простору імен System.ServiceModel и MagicBallServiceLib.

Код хоста:

```
using System;
```

```

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

using System.ServiceModel;
using MagicBallServiceLib;

namespace MagicBallServiceHost
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("***** Console Based WCF Host *****");

            using (ServiceHost serviceHost = new ServiceHost(typeof(MagicBallService)))
            {
                // Відкрити хост і слухати вхідні повідомлення.
                foreach (System.ServiceModel.Description.ServiceEndpoint se in
serviceHost.Description.Endpoints)
                {
                    Console.WriteLine("Address: {0}", se.Address);
                    Console.WriteLine("Binding: {0}", se.Binding.Name);
                    Console.WriteLine("Contract: {0}", se.Contract.Name);
                    Console.WriteLine();
                }
                serviceHost.Open();

                // Для виходу - натиснути Enter.
                Console.WriteLine("The service is ready.");
                Console.WriteLine("Press the Enter key to terminate service.");
                Console.ReadLine();
            }
        }
    }
}

```

Додати конфігураційний файл App.config для хоста з таким змістом:

```

<?xml version="1.0"?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="MagicBallServiceLib.MagicBallService"
behaviorConfiguration="BallServiceMEXBehavior">
        <endpoint address=""
binding="basicHttpBinding"
contract="MagicBallServiceLib.IMagicBall"/>

        <!-- Дозволити кінцеву точку MEX -->
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

        <!-- Повідомляємо MEX адресу сервісу -->
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8080/MagicBallService"/>
          </baseAddresses>
        </host>
      </service>
    </services>

    <!-- Визначення поведінки для MEX -->
    <behaviors>
      <serviceBehaviors>
        <behavior name="BallServiceMEXBehavior">
          <serviceMetadata httpGetEnabled="true"/>

```

```

        </behavior>
    </serviceBehaviors>
</behaviors>
</system.serviceModel>
<startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
</startup>
</configuration>

```

Скомпілювати хост.

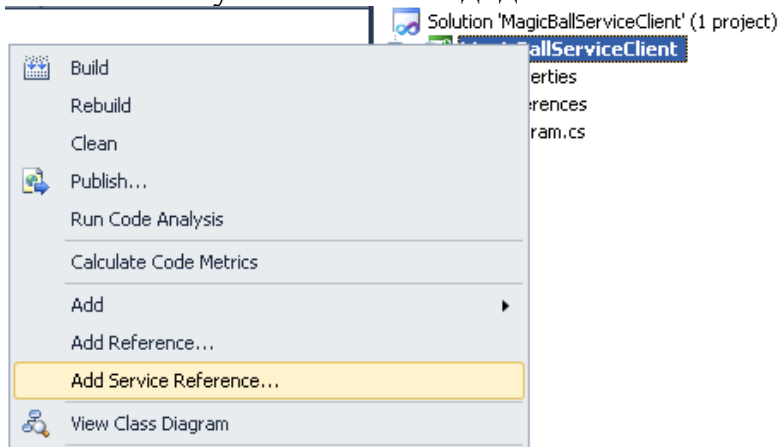
Звернути увагу на згенерований файл MagicBallServiceHost.exe.config.

Запустити MagicBallServiceHost.exe.

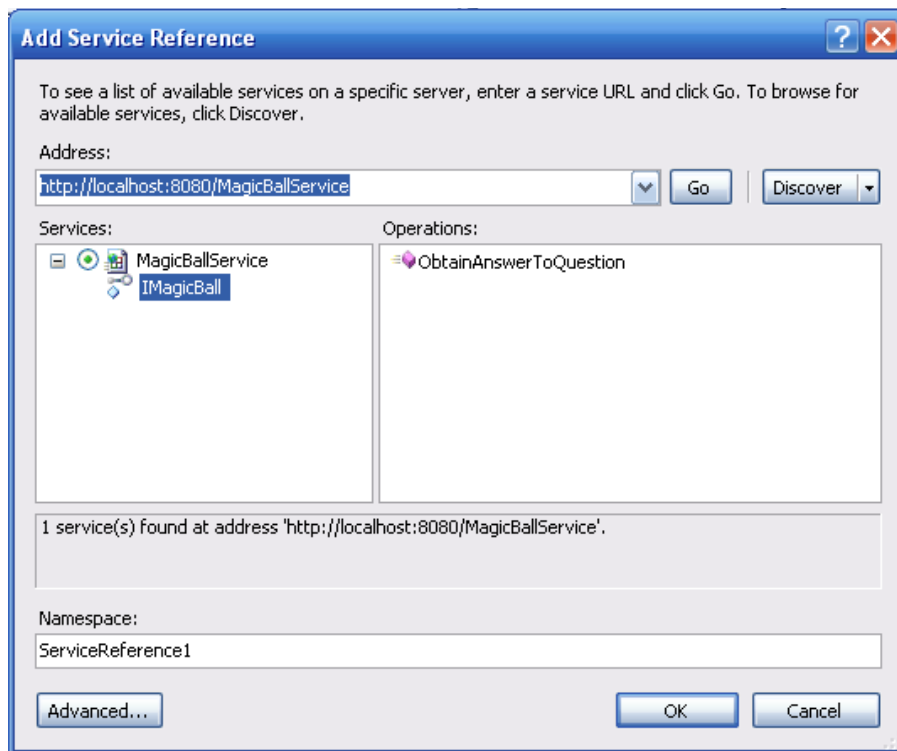
Звернутись за адресою <http://localhost:8080/MagicBallService> з веб-браузера.

3. Створити клієнта служби WCF як консольний проект C# з ім'ям MagicBallServiceClient.

Конфігураційний файл та файл проксі можна згенерувати за допомогою утиліти svcutil, або засобами VS 2010. Останній варіант виконується так: запустити хост та додати посилання на службу



Вказати Uri



Переглянути файл проксі Reference.cs та файл конфігурації клієнта app.config.

Додати код клієнта:

```
using MagicBallServiceClient.ServiceReference1;

namespace MagicBallServiceClient
{
    class Program
    {
        static void Main(string[] args)
        {
            string question;
            Console.WriteLine("***** Запитуйте магичну кулю *****\n");

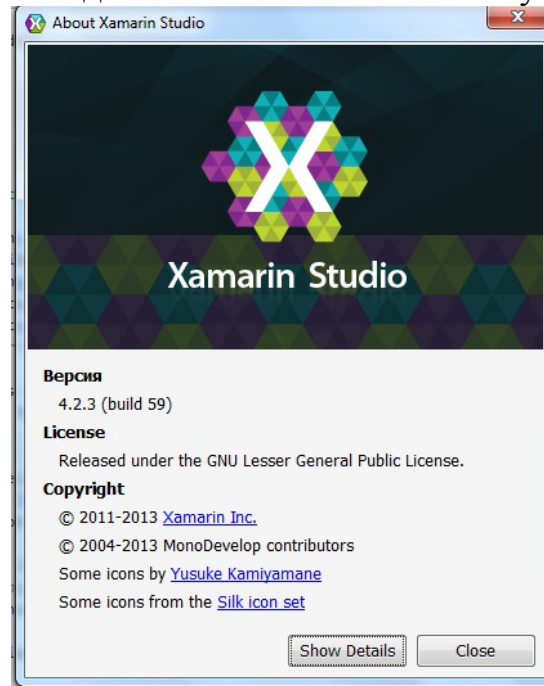
            using (MagicBallClient ball = new MagicBallClient())
            {
                do
                {
                    Console.Write("Ваше питання: ");
                    question = Console.ReadLine();
                    if (question != "")
                    {
                        string answer = ball.ObtainAnswerToQuestion(question);
                        Console.WriteLine("Магічна куля: {0}", answer);
                    }
                } while (question != "");
            }
        }
    }
}
```

Скомпілювати клієнта.

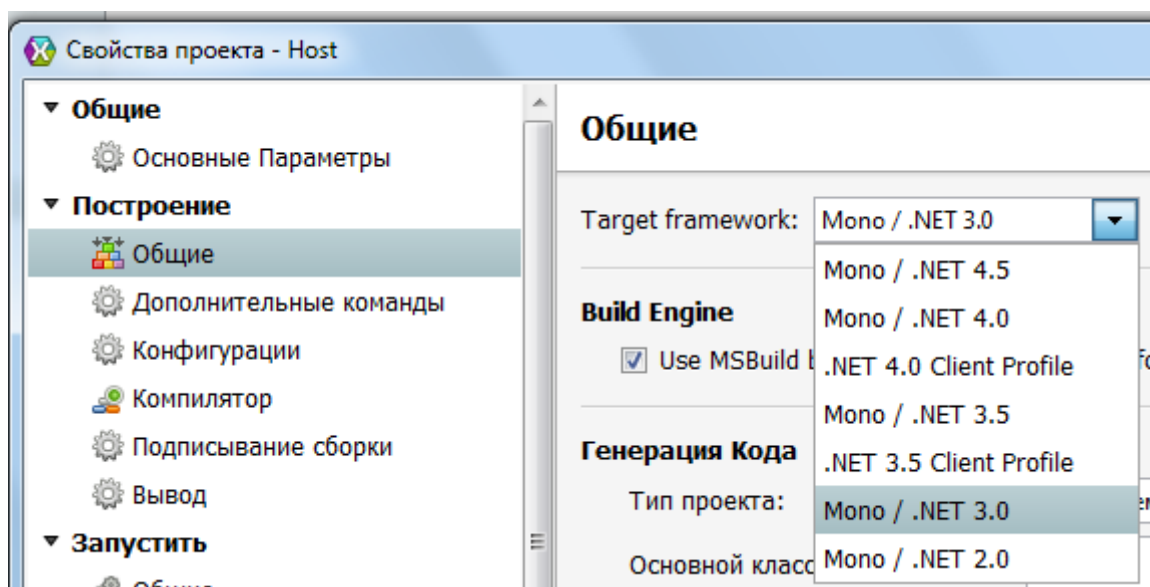
4. Перевірити взаємодію клієнта зі службою.

Приклад 2

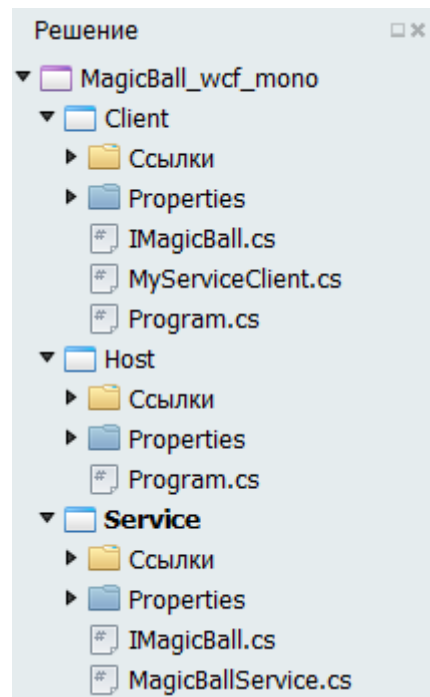
Застосування створимо за допомогою Xamarin Studio у середовищі Windows.



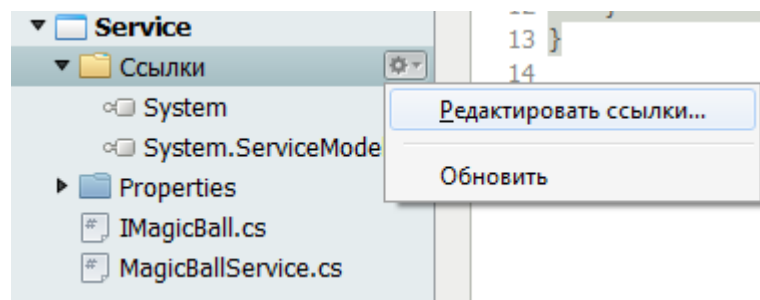
1. Встановити цільові платформи



2. Створити рішення, структура якого буде такою:



3. Створити службу



IMagicBall.cs

```
using System;
using System.ServiceModel;

namespace Service
{
    [ServiceContract]
    public interface IMagicBall
    {
        // Отримати відповідь на питання!
        [OperationContract]
        string ObtainAnswerToQuestion(string userQuestion);
    }
}
```

MagicBallService.cs

```
using System;
using System.ServiceModel;
```

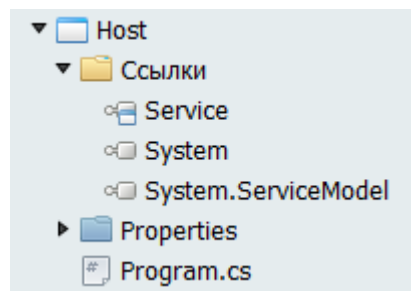
```

namespace Service
{
    public class MagicBallService: IMagicBall
    {
        public MagicBallService()
        {
            Console.WriteLine("Магічна куля відповідає...");
        }
        public string ObtainAnswerToQuestion(string userQuestion)
        {
            string[] answers = { "Майбутнє непевне", "Так", "Ні",
                                "Можливо", "Запитайте пізніше", "Безумовно" };

            // Return a random response.
            Random r = new Random();
            return answers[r.Next(answers.Length)]+"\n";
        }
    }
}

```

4. Створити хост



Program.cs

```

using System;
using System.ServiceModel;
using Service;

namespace Host
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("WCF Host for MagicBall!");
            BasicHttpBinding binding = new BasicHttpBinding ();
            Uri address = new Uri ("http://localhost:8080");
            ServiceHost host = new ServiceHost (typeof(MagicBallService));

            host.AddServiceEndpoint (
                typeof(IMagicBall), binding, address);
        }
    }
}

```



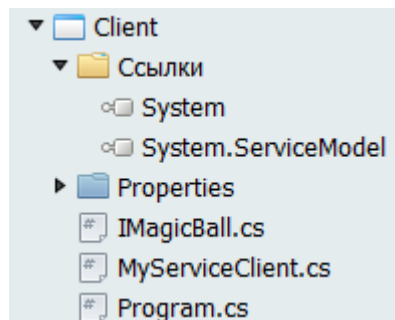
```

        host.Open ();

        Console.WriteLine ("Type Enter to stop ...");
        Console.ReadLine ();
        host.Close ();
    }
}

```

5. Створити клієнта



IMagicBall.cs

```

using System;
using System.ServiceModel;

namespace Client
{
    [ServiceContract]
    public interface IMagicBall
    {
        // Отримати відповідь на питання!
        [OperationContract]
        string ObtainAnswerToQuestion(string userQuestion);
    }
}

```

MyServiceClient.cs

```

using System;
using System.ServiceModel;
using System.ServiceModel.Channels;

namespace Client
{
    public class MyServiceClient: ClientBase<IMagicBall>, IMagicBall
    {
        public MyServiceClient (Binding binding, EndpointAddress address) : base (binding, address)
        {
        }
    }
}

```

```

        public string ObtainAnswerToQuestion(string userQuestion)
        {
            return Channel.ObtainAnswerToQuestion(userQuestion);
        }
    }
}

```

Program.cs

```
using System.ServiceModel;
```

```

namespace Client
{
    class MainClass
    {
        static void Main(string[] args)
        {
            string question;
            Console.WriteLine("*****Запитайте магичну кулю*****\n");
            BasicHttpBinding binding = new BasicHttpBinding ();
            var address = new EndpointAddress ("http://localhost:8080");
            var ball = new MyServiceClient (binding, address);

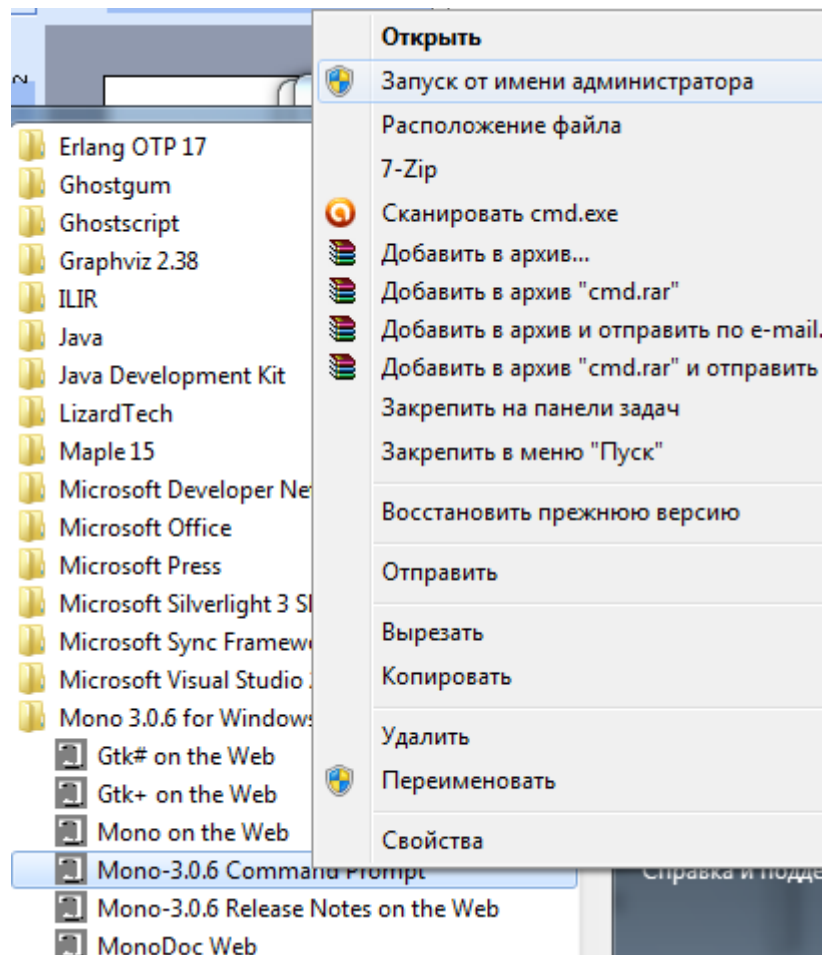
            do{
                Console.Write("Ваше питання: ");
                question = Console.ReadLine();
                if (question != "")
                {
                    Console.WriteLine("Реакція магичної кулі: {0} ...",
                                      ball.ObtainAnswerToQuestion(question));

                }
            } while (question!="");

            Console.ReadLine();
        }
    }
}

```

6. Запустити хост (через командний рядок Mono) від імені адміністратора



```

Администратор: Mono-3.0.6 Command Prompt - mono Host.exe

Mono version 3.0.6 Build 0
Prepending 'D:\PROGRA~2\MONO-3~1.6\bin' to PATH
C:\Windows\system32>cd C:\Users\User\Documents\Projects\MagicBall_wcf_mono\Host\bin\Debug

C:\Users\User\Documents\Projects\MagicBall_wcf_mono\Host\bin\Debug>mono Host.exe

WCF Host for MagicBall!
Type Enter to stop ...
  
```

7. Запустити клієнта через Mono

```

Mono-3.0.6 Command Prompt - mono Client.exe

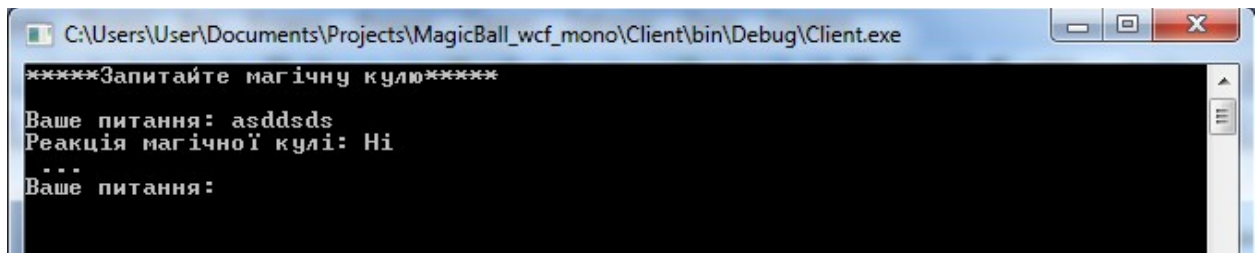
Mono version 3.0.6 Build 0
Prepending 'D:\PROGRA~2\MONO-3~1.6\bin' to PATH
C:\Windows\System32>cd C:\Users\User\Documents\Projects\MagicBall_wcf_mono\Client\bin\Debug

C:\Users\User\Documents\Projects\MagicBall_wcf_mono\Client\bin\Debug>mono Client.exe

*****Запитайте магичну кулю*****

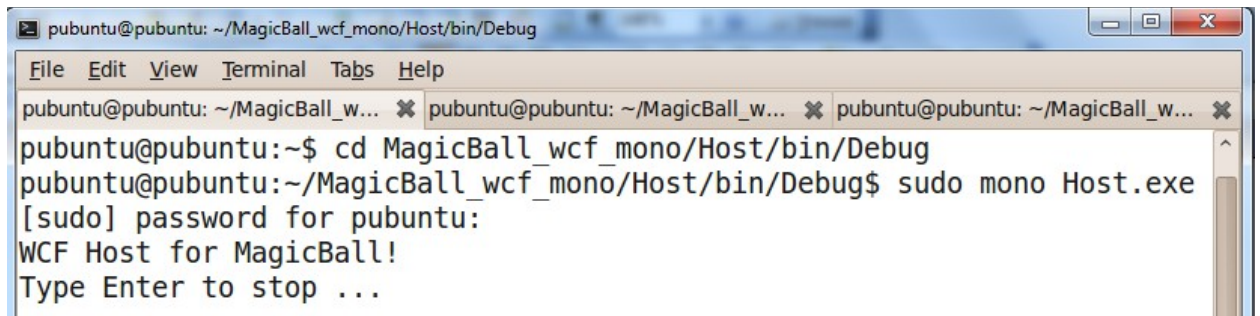
Ваше питання: Чи корисні найновіші інструменти?
Реакція магичної кулі: Можливо
...
Ваше питання: Тобто можна користуватись застарілими?
Реакція магичної кулі: Запитайте пізніше
...
Ваше питання: Тобто можна користуватись застарілими?
Реакція магичної кулі: Ні
...
Ваше питання:
  
```

8. Запустити клієнта без Mono

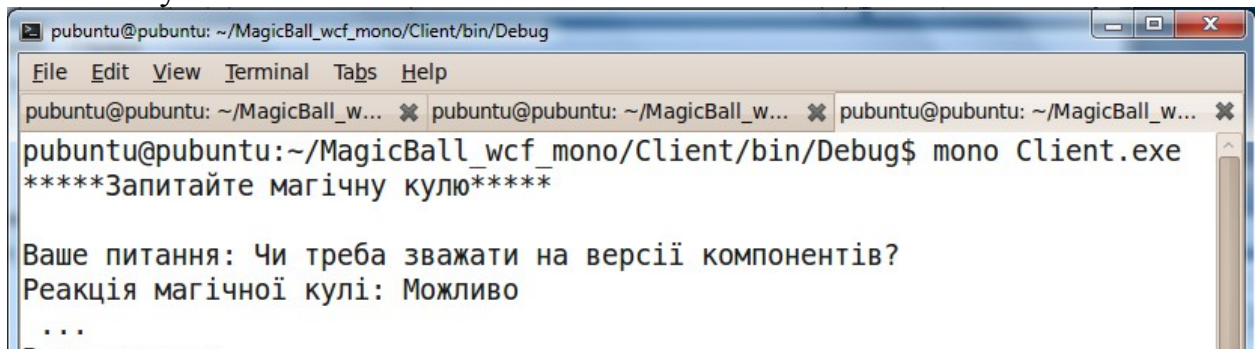


9. Скопіювати рішення в Ubuntu.

10. Запустити від хоста імені адміністратора



11. Запустити клієнта



Приклад 3

Створимо службу (сервіс) WCF для реалізації арифметичних операцій над дійсними числами.

Служба представимо однією збіркою, яка міститиме контракт служби `ICalculator`, клас реалізації контракту `CalculatorService` (є службою WCF), клас `CalculatorWindowsService` (є службою Windows) та клас інстальатора `ProjectInstaller`.

Клас `CalculatorWindowsService`, щоб бути службою Windows, успадковує клас `ServiceBase` та реалізує методи `OnStart` та `OnStop`. Метод `OnStart` створює об'єкт [ServiceHost](#) типа `CalculatorService` і відкривається для прослуховування. Метод `OnStop` зупиняє службу. Клас установника, успадкований від класу [Installer](#), дозволяє виконати встановлення програми як служби Windows за допомогою утиліти `Installutil.exe`.

Клієнт реалізується окремою збіркою.

Для реалізації служби треба виконати наступні дії.

1. Створити новий проект консольного застосунку Visual Studio з ім'ям Service у рішенні `myService`.
2. Додати посилання на збірки:
 - `System.ServiceModel.dll`

- System.ServiceProcess.dll
 - System.Configuration.Install.dll
- 3. Додати оператори using в файл Service.cs.

```
using System.ComponentModel;
using System.ServiceModel;
using System.ServiceProcess;
using System.Configuration;
using System.Configuration.Install;
```
- 4. Визначити контракт служб ICalculator:

```
[ServiceContract(Namespace = "http://myService")]
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);
}
```
- 5. Реалізувати контракт служби у класі CalculatorService:

```
public class CalculatorService : ICalculator
{
    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;
    }
}
```

6. Створити новий клас `CalculatorWindowsService`, похідний від класу [ServiceBase](#). Додати локальну змінну `serviceHost`, для посилання на екземпляр [ServiceHost](#). Визначте метод `Main`, що викликає `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());

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

            serviceHost =
                new ServiceHost(typeof(CalculatorService));

            serviceHost.Open();
        }

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

7. Створити новий клас `ProjectInstaller`, похідний від [Installer](#) з атрибутом [RunInstallerAttribute](#), встановленим у значення `true`. Це дозволяє встановлювати службу Windows програмою `Installutil.exe`

```
[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);
}

```

8. Видаліть клас Service, створений автоматично.
9. Додайте до проекту файл конфігурації App.config з таким змістом.

```

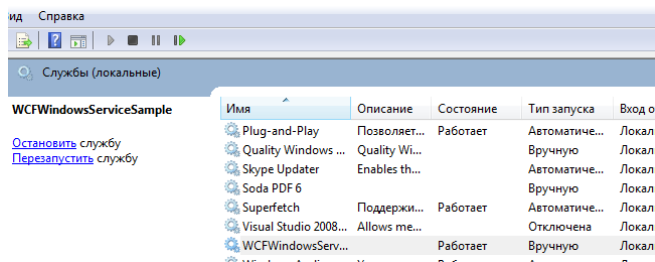
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <!-- This section is optional with the new configuration model
            introduced in .NET Framework 4. -->
      <service name="myService.CalculatorService"
        behaviorConfiguration="CalculatorServiceBehavior">
        <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="myService.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>

```

10. В редакторі властивостей файлу App.config надайте властивості *Копировать в выходной каталог* значення *Копировать более новые*.
11. Побудуйте рішення, щоб створити виконуваний файл myService.exe.

Встановлення і запуск служби

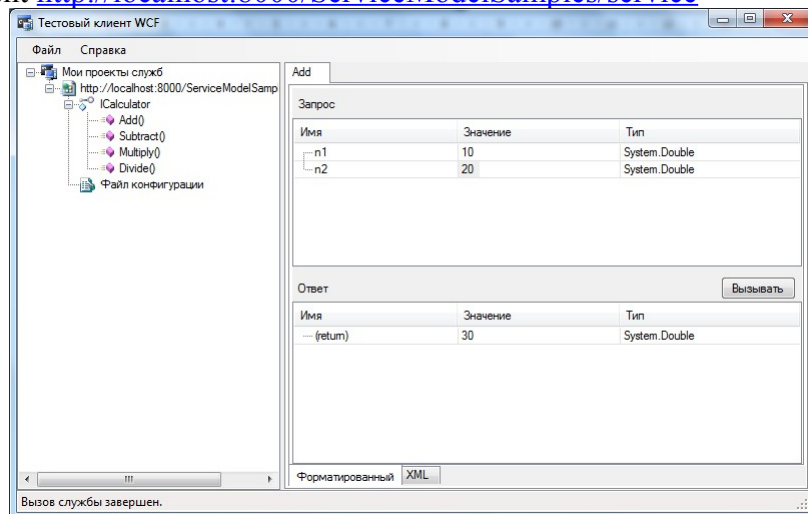
12. Встановити службу можна з командного рядка Visual Studio (з правами адміністратора) командою *installutil MathWindowsServiceHost.exe*. Після успішного встановлення можна переглянути Services (Службы) в теці Administrative Tools (Администрирование) панелі управління.



Запустить її можна посиланням (Запустити службу), або кнопкою . Запущену службу можна зупинити, або перезапустити. Видалити службу (для перевизначення) можна командою

installutil /u MathWindowsServiceHost.exe.

13. Перевірити функціонування служби можна за допомогою команди wcfestclient <http://localhost:8000/ServiceModelSamples/service>

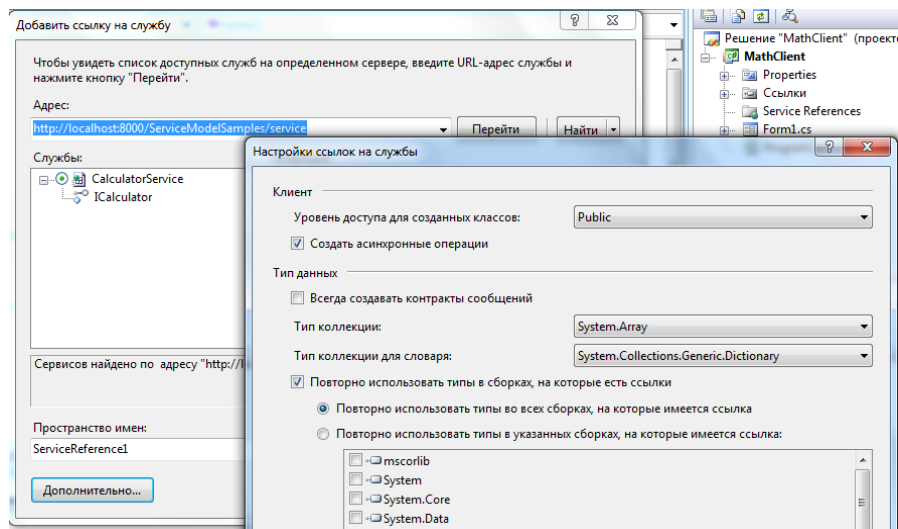


14. Повний текст коду наведено у Додатку.

Для використання сервісу реалізувати клієнта

15. Створити консольний застосунок MathConsoleClient.

16. Додати посилання на службу (служба має бути запущеною), вибрати підтримку асинхронних операцій



17. Код клієнта:

```
using System;
```



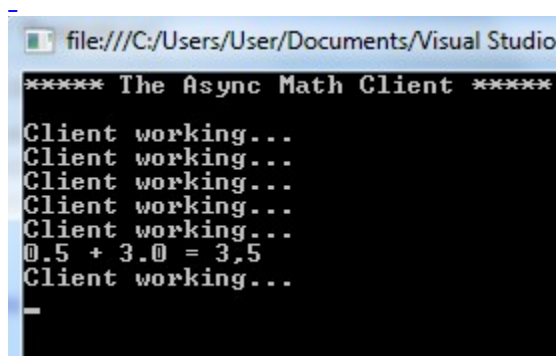
```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using MathConsoleClient.ServiceReferencel;
using System.Threading;

namespace MathConsoleClient
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("***** The Async Math Client
            *****\n");
            using (CalculatorClient proxy = new
                CalculatorClient())
            {
                proxy.Open ();
                // Додавання в асинхронному режимі з
                //використанням лямбда-виразів.
                IAsyncResult result = proxy.BeginAdd(0.5,3.0,      ar =>
{ Console.WriteLine("0.5 + 3.0 =
                    ,proxy.EndAdd(ar));
                                }, null) ;
                while (!result.IsCompleted)
                {
                    Thread.Sleep(200);
                    Console.WriteLine("Client working...");
                }
            }
            Console.ReadLine();
        }
    }
}

```

18. Після компіляції та запуску клієнта отримаємо результат, подібний наступному:



```

file:///C:/Users/User/Documents/Visual Studio
***** The Async Math Client *****
Client working...
Client working...
Client working...
Client working...
Client working...
0.5 + 3.0 = 3,5
Client working...
-

```

Завдання для самостійного виконання

Розробити, у відповідності до варіанта завдання:

1. WCF – службу зі спеціальним хостингом для Windows. Сконфігурувати сервіс для роботи з каналами http та tcp (див. також Додаток);
2. WCF – службу зі спеціальним хостингом для Windows та Linux;
3. WCF – службу з хостингом у керованих службах Windows (факультативне завдання).

Створити консольний застосунок - клієнта створеної служби. Продемонструвати працездатність застосунку.

№ варіант а	Зміст завдання
1	Користувач вводить рядок. Сервіс визначає чи містить воно повтори, (підрядок принаймні з двох літер, який зустрічається більше одного разу). Наприклад рядок abbak – не містить повторів, а abbbab – містить два повтори підрядка ab та два повтори підрядка bb.
2	Користувач вводить рядок. Сервіс знаходить його довжину K та визначає, чи є воно (і яким по порядку) простим числом.
3	Користувач вводить рядок та число K. Сервіс знаходить кількість N слів у рядку та визначає найбільший спільний дільник чисел N та K.
4	Користувач вводить рядок. Сервіс знаходить кількість літер N1 та пробілів N2 у ньому, знаходить суму чисел від $\min(N1, N2)$ до $\max(N1, N2)$ включно.
5	Користувач вводить рядок та число M. Сервіс знаходить довжину N найбільшого слова у рядку, визначає найменше спільне кратне чисел M і N.
6	Користувач вводить рядок. Сервіс визначає чи є воно, або будь-який його префікс паліндромом (напр. рядок abbak – не паліндром, але його префікс abba – паліндром).
7	Користувач вводить два рядки. Сервіс визначає їх найбільший спільний префікс (напр. найбільший спільний префікс у abcs та abfgr є ab. А у abcs та fghrt – порожній рядок).
8	Сервіс визначає чи є в тексті записи цілих та/або дійсних чисел.
9	Користувач вводить рядок. Сервіс визначає довжину кожного слова тексту.
10	Користувач вводить слово. Сервіс повертає всі його анаграми (всі перестановки літер).
11	Сервіс отримує два рядки та змішує їх (напр. для abc та FGHK повертає aFbGcHK).
12	Сервіс повідомляє кількість одиниць у двійковому представленні числа та саме двійкове представлення.
13	Сервіс повертає кількість та список парних цифр у отриманому від

	користувача числі.
14	Користувач вводить два рядки. Сервіс визначає їх найбільший спільний суфікс. Наприклад найбільший спільний суфікс у abcsdf та abfgrsdf є sdf. А у abcs та fghrt – порожній рядок.
15	Рядки Фібоначчі обчислюють так: $f_0 = b$, $f_1 = a$, $f_n = f_{n-1} + f_{n-2}$, $n \geq 2$, де знаком + позначена операція конкатенації. Наприклад $f_2 = ab$, $f_3 = aba$, $f_4 = abaab$ тощо. Сервіс знаходить перші N рядків Фібоначчі та довжину кожного з них.
16	Сервіс перевіряє, чи є N-значний номер квитка "щасливим". Сервіс повертає висновок та суму перших і останніх N/2 цифр ("щасливим" вважається номер, в якому сума перших N/2 цифр збігається з сумою останніх N/2 цифр).
17	Клієнт передає рядок та число N. Сервіс виконує циклічний зсув рядка вправо на N слів.
18	Клієнт передає рядок та число N. Сервіс виконує циклічний зсув рядка вліво на N символів.

Література

1. Троелсен, Эндрю Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд. : Пер. с англ. – М. : ООО "И.Д. Вильямс", 2011.
2. Джувел Леве - Создание служб Windows Communication Foundation. – СПб.: Питер, 2008.

Додаток

Приклади конфігурування клієнта та сервера для роботи з tcp каналом

```
<client>
  <endpoint address="net.tcp://localhost:8090/MagicBallService"
    binding="netTcpBinding"
    contract="ServiceReference1.IMagicBall"
    name="netTcpBinding_IMagicBall" />
</client>
```

```
<services>
  <service name="MagicBallServiceLib.MagicBallService">
    <endpoint address=""
      binding="netTcpBinding"
      contract="MagicBallServiceLib.IMagicBall"/>

    <host>
      <baseAddresses>
        <add baseAddress="net.tcp://localhost:8090/MagicBallService"/>
      </baseAddresses>
    </host>
  </service>
</services>
```