

Ćwiczenie 4.1

WCF – podstawy i konfigurowanie usługi

Autor: Mariusz Fraś

1 Cele ćwiczenia

Celem ćwiczenia jest:

1. Poznanie podstawowej architektury aplikacji WCF
 2. Zapoznanie się z podstawami tworzenia usługi z opisem WSDL i dostępnej protokołem SOAP (tu: usługi WCF), i klienta takiej usługi (tu: klienta WCF).
 3. Poznanie opcji konfigurowania usług – punktów końcowych, transportu, sposobu działania usługi.
- **Pierwsza część zadania** jest do wykonania według podanej instrukcji i ewentualnych poleceń prowadzącego zajęcia laboratoryjne.
 - **Druga część zadania** jest do przygotowania i oddania lub do wykonania wg. poleceń prowadzącego na kolejnych zajęciach.

2 Zadanie – część I

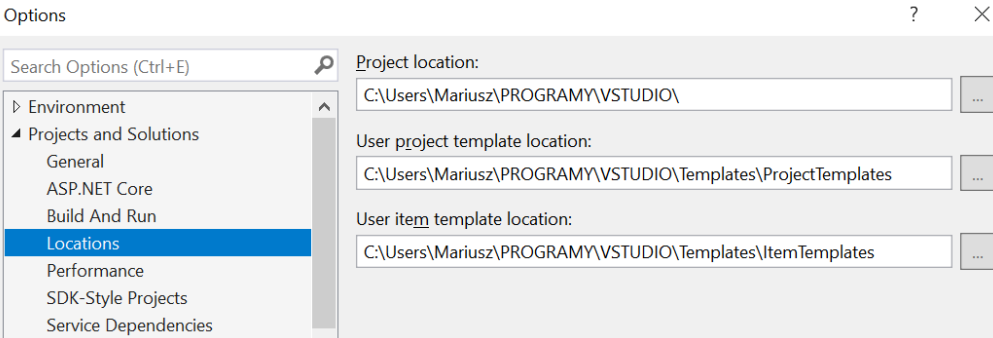
Konstrukcja podstawowej aplikacji WCF

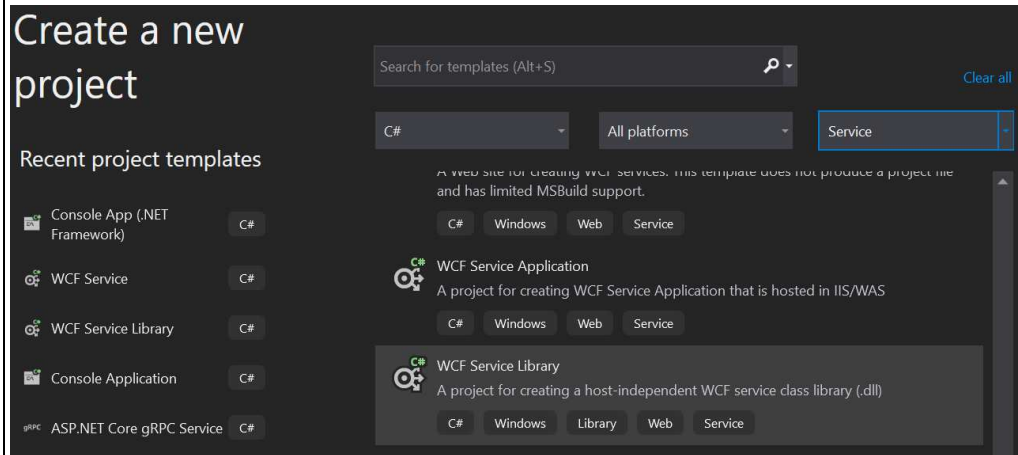
W zadaniu zostanie zrealizowane rozwiązanie obejmujące: **a)** usługę uruchamianą jako odrębna aplikacja i **b)** klienta korzystającego z tej usługi. Usługa i klient zostaną zrealizowane w narzędziu Visual Studio (dalej w skrócie VS) jako usługa WCF. Zadanie realizacji podstawowej aplikacji WCF klient-serwer składa się z kilku etapów:

1. Zdefiniowanie kontraktu usługi.
2. Implementacja kontraktu usługi (implementacja usługi).
3. Utworzenie aplikacji hostującej usługę
– tu: jest to aplikacja konsolowa – tzw. self-hosting service.
4. Utworzenie aplikacji klienckiej i w niej proxy klienta (ang. *proxy client*).
5. Skonfigurowanie proxy klienta.
6. Implementacja aplikacji klienckiej.

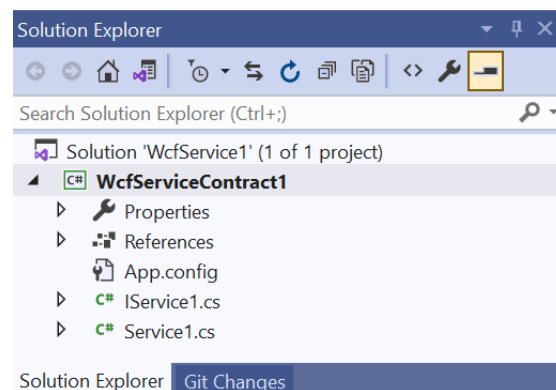
UWAGA: wszelkie zmiany (np. nazw klas itp.) w już automatycznie wygenerowanym przez platformę kodzie najlepiej realizować poprzez opcję refaktoryzacji narzędziami platformy

- W VS2013 **opcja Refactor w menu kontekstowym** (po kliknięciu prawego klawisza myszy).
- W VS2015÷2019 **odpowiednia opcja w menu kontekstowym** (np. Rename) - brak jest wyróżnionej opcji Refactor.

<p>1. Wstępne przygotowanie środowiska</p>	<ul style="list-style-type: none"> • Przygotuj katalog roboczy, w którym będą tworzona aplikacja WCF. • Uruchom platformę Visual Studio. • Sprawdź i ewentualnie ustaw foldery w których będą tworzone projekty w opcji TOOLS→Options...→Projects and Solutions→Locations (lub General we wcześniejszych wersjach VS). 
<p>2. Definiowanie kontraktu usługi WCF</p>	<ul style="list-style-type: none"> • Utwórz nową solucję i projekt aplikacji z szablonu Visual C# WCF Service Library nadając własne nazwy solucji i projektowi (tu: WcfService1 i WcfServiceContract1).



- Przeglądnij zawartość projektu. Zwróć uwagę na następujące elementy:
 - **IService1.cs** – plik deklaracji (interfejs) kontraktu,
 - **Service1.cs** – implementacja kontraktu,
 - **App.config** – konfiguracja własności i dostępności implementacji kontraktu.



Uwaga: w najnowszej wersji VS nazwy plików mogą się zmienić po zmianie nazwy interfejsu lub klasy z domyślnej (jak wyżej) na własną – na taką jak własna nazwa klasy lub interfejsu. Takie zachowanie jest konfigurowalne (można to wyłączyć).

- Otwórz plik **IService1.cs** i zdefiniuj kontrakt usługi – interfejs **ICalculator** zawierający metody Add, Sub, i Multiply:
 - Usuń nieużywany kod.
 - Dopisz lub przerób kod do postaci:

Uwaga: nazwa przestrzeni nazw (namespace) jest ustawiana taka jak nazwa projektu –nie trzeba jej zmieniać.

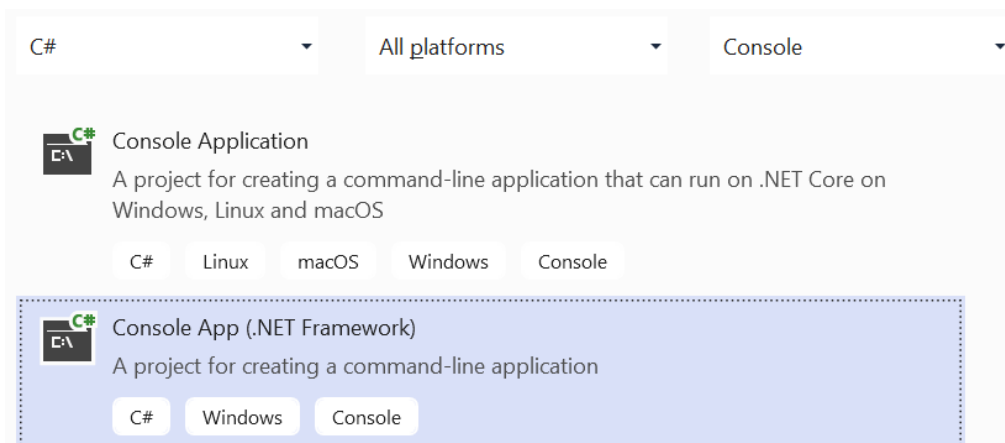
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
```

	<pre>namespace WcfServiceContract1 { [ServiceContract(ProtectionLevel = ProtectionLevel.None)] public interface ICalculator { [OperationContract] double Add(double n1, double n2); [OperationContract] double Sub(double n1, double n2); [OperationContract] double Multiply(double n1, double n2); } }</pre> <p>Uwaga: tu po zmianie nazwy z <i>Iservice1</i> na <i>ICalculator</i> (opcją menu – nie manualnie!) nazwa pliku może zmienić się w projekcie.</p> <ul style="list-style-type: none"> Własność (<i>behavior</i>) <code>ProtectionLevel</code> (ustawioną na <code>None</code>) dodano w celu uniknięcia kwestii oprogramowania autoryzacji przy dostępie do usług.
3. Implementacja kontraktu usługi WCF	<ul style="list-style-type: none"> Otwórz plik Service1.cs. Wpisz kod klasy MyCalculator implementującej interfejs ICalculator: Zaimplementuj każdą z wymaganych metod: <pre>using System; using System.Collections.Generic; using System.Linq; using System.Runtime.Serialization; using System.ServiceModel; using System.Text; namespace WcfServiceContract1 { public class MyCalculator : ICalculator { public double Add(double n1, double n2) { ... } public double Sub(double n1, double n2) { ... } public double Multiply(double n1, double n2) { ... } } }</pre> W miejsce kropek ... dopisz odpowiedni kod dla każdej metody: <ul style="list-style-type: none"> wykonanie odpowiedniego działania, wyświetlenie informacji w konsoli co jest wywołane, co otrzymano w wywołaniu i co jest zwracane, zwrócenie odpowiedniej wartości.

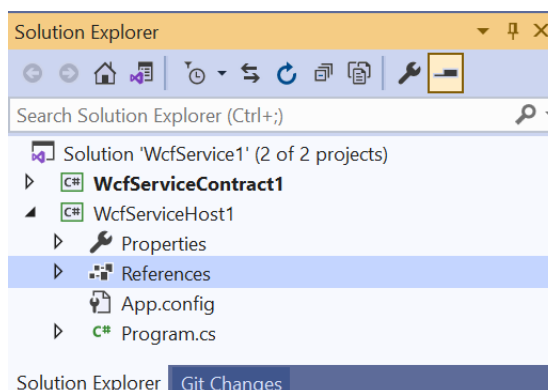
4. Hostowanie usługi WCF

Utwórz aplikację konsolową hostującą usługę WCF (Host usługi).

- Dodaj do dotychczasowej solucji drugi projekt aplikacji konsolowej nadając mu własną nazwę (tu: WcfServiceHost1)
– opcja: **Add... → New Project.**



- Sprawdź (i ewentualnie ustaw) wersję Framework'a aplikacji.
 - Przy tworzeniu lub zaznacz projekt w **Solution Explorer**, prawym klawiszem wywołaj menu kontekstowe Właściwości (**Properties**) i sprawdź wartość w opcji **Application** → **Target Framework**.
- Dodaj w projekcie referencję do projektu kontraktu usługi WCF:
 - Zaznacz w Solution Explorer folder **References** i wybierz opcję **Add Reference**.

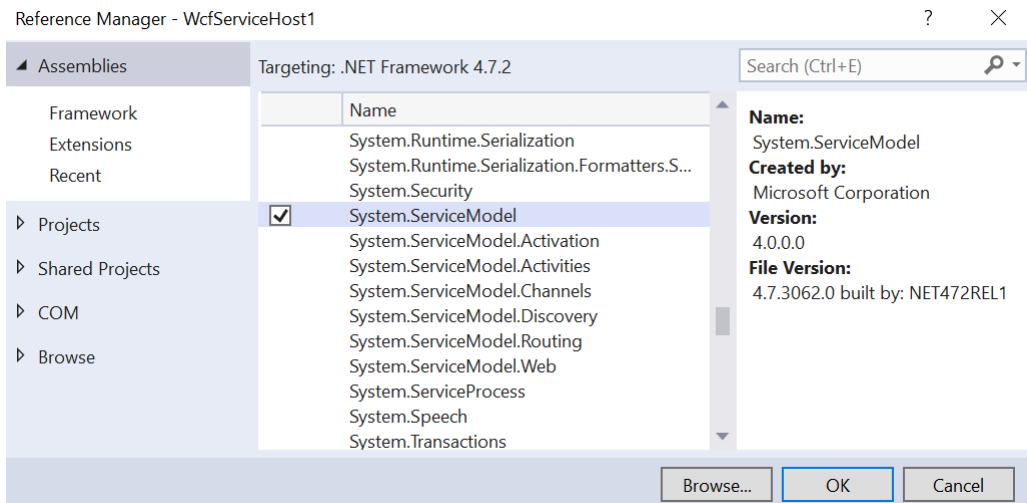


- W oknie menadżera referencji wybierz **Solution/Project**, zaznacz projekt kontraktu usługi WCF i zatwierdź:

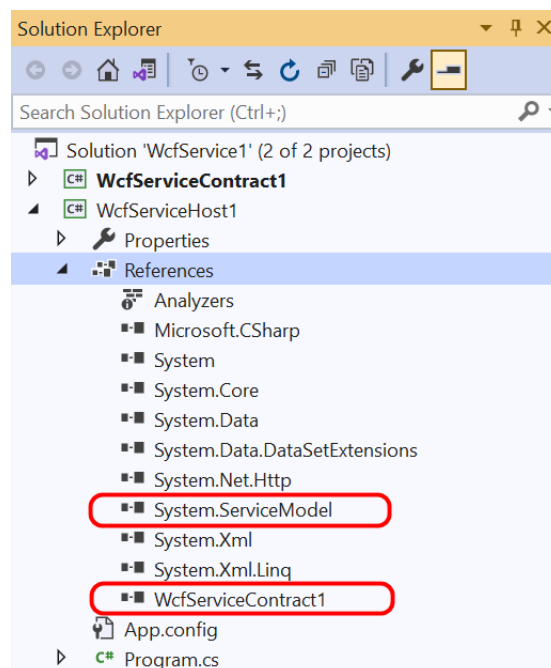


- Dodaj w projekcie referencję do **System.ServiceModel**:
 - Kliknij w Solution Explorer prawym klawiszem folder **References** i wybierz opcję **Add Reference**.

- o W oknie menadżera referencji wybierz **Assemblies/Framework**, zaznacz **System.ServiceModel** i zatwierdź.



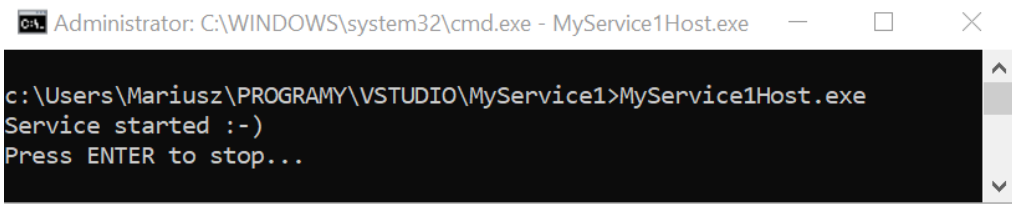
- Zweryfikuj pojawienie się dodatkowych referencji w projekcie jak na rysunku poniżej:



- Otwórz plik **Program.cs** i wpisz kod realizujący następujące funkcje:
 - o Utworzenie URI z bazowym adresem serwisu.
 - o Utworzenie instancji serwisu.
 - o Dodanie punktu końcowego serwisu.
 - o Umożliwienie wymiany metadanych (informacji o serwisie).
 - o Uruchomienie serwisu (i na koniec zamknięcie serwisu).

Zamiast **xxx** podaj port **10000+nr stanowiska w laboratorium**.
Zamiast **NazwaBazowa** (nazwa serwisu) wpisz własną nazwę swojej usługi.

	<pre> using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; namespace WcfServiceHost { class Program { static void Main(string[] args) { // Krok 1 Utworz URI dla bazowego adresu serwisu. Uri baseAddress = new Uri("http://localhost:xxx/NazwaBazowa"); // Krok 2 Utworz instancje serwisu. ServiceHost myHost = new ServiceHost(typeof(MyCalculator), baseAddress); // Krok 3 Dodaj endpoint. WSHttpBinding myBinding = new WSHttpBinding(); ServiceEndpoint endpoint1 = myHost.AddServiceEndpoint (typeof(ICalculator), myBinding, "endpoint1"); // Krok 4 Ustaw właczenie metadanych. ServiceMetadataBehavior smb = new ServiceMetadataBehavior(); smb.HttpGetEnabled = true; myHost.Description.Behaviors.Add(smb); try { // Krok 5 Uruchom serwis. myHost.Open(); Console.WriteLine("Serwis jest uruchomiony."); Console.WriteLine("Nacisnij <ENTER> aby zakonczyc."); Console.WriteLine(); Console.ReadLine(); // aby nie kończyć natychmiast: myHost.Close(); } catch (CommunicationException ce) { Console.WriteLine("Wystapil wyjatek: {0}", ce.Message); myHost.Abort(); } } } } </pre> <ul style="list-style-type: none"> • Usuń błędy poprzez dodanie importu odpowiednich bibliotek - po zaznaczeniu kursorem, opcja Quick Actions and Refactorings... w menu kontekstowym lub Show potential fixes. <ul style="list-style-type: none"> ○ Najczęściej będzie to dodanie dyrektywy importu nazw using.
5. Testowanie działania aplikacji	<p><i>UWAGA: aby uruchomić serwis trzeba mieć uprawnienia administratora w systemie Windows. W innym przypadku trzeba system dodatkowo odpowiednio skonfigurować.</i></p> <ul style="list-style-type: none"> • Przetestuj poprawność działania aplikacji

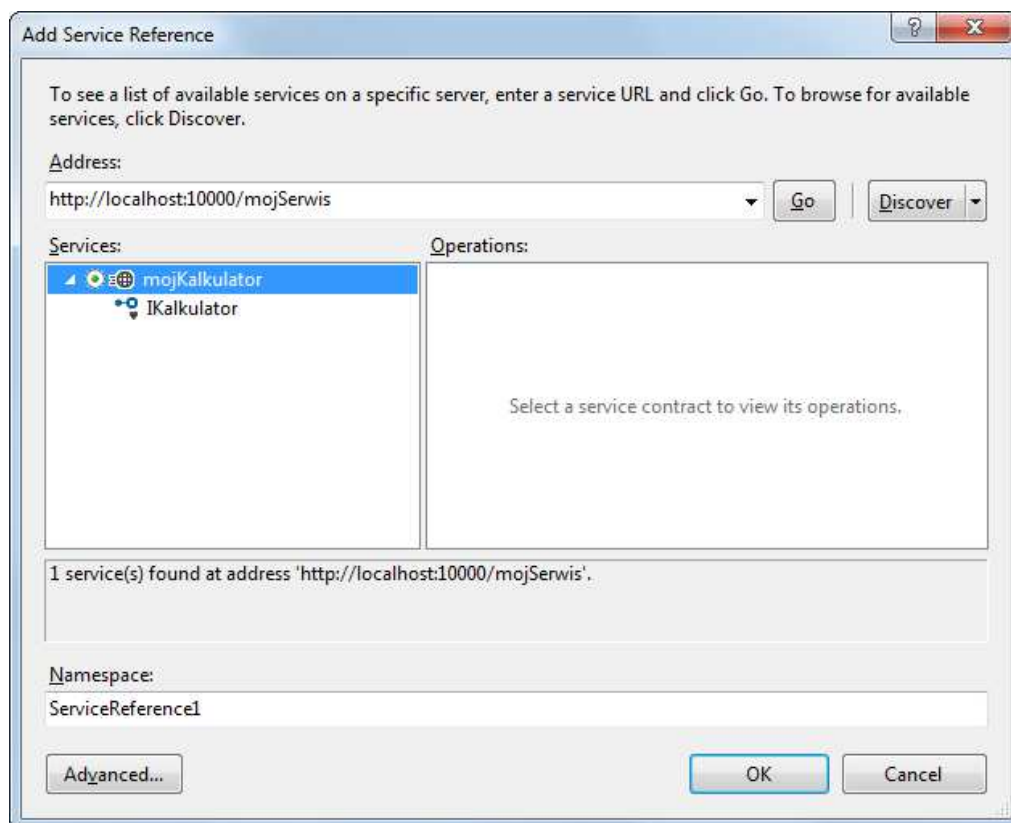
	<ul style="list-style-type: none"> ○ Zbuduj kod wykonywalny aplikacji – użyj jednej z opcji: <ul style="list-style-type: none"> – BUILD → Build <i>Nazwa_projektu</i> (zbudowanie aplikacji – zwykle za pierwszym razem). – BUILD → Rebuild <i>Nazwa_projektu</i> (kolejne powtórzenia – zwykle po modyfikacjach). – BUILD → Build Solution (dotyczy całej solucji / wszystkich projektów – zwykle za pierwszym razem) – BUILD → Rebuild Solution (zwykle kolejne powtórzenia). ○ Uruchom z linii komend aplikację hostującą serwis WCF  <ul style="list-style-type: none"> ○ Uruchom przeglądarkę i spróbuj połączyć się z adresem: http://localhost:xxx/<i>NazwaBazowa</i> <i>Połączenie z hostem i wyświetlenie strony z odpowiednim opisem oznacza poprawność działania aplikacji.</i> • Przejdź do strony z opisem WSDL usługi. Zamknij węzły <i>Policy</i> (są tu nieistotne) i zidentyfikuj ważne części opisu usługi: typy, wiadomości, operacje, punkt dostępu, itp. • Uruchomienie hosta usługi z poziomu VS automatycznie uruchamia wbudowanego klienta umożliwiającego przetestowanie działania usługi. Kliknij w tym kliencie którąś operację i sprawdź postać wysyłanego komunikatu XML (SOAP).
6. Tworzenie klienta usługi.	<p>Utworzenie aplikacji klienta usługi i proxy klienta (<i>client proxy</i>) z użyciem funkcji Visual Studio: Add Service Reference.</p> <ul style="list-style-type: none"> • Dodaj do solucji trzeci projekt aplikacji z szablonu Visual C# Console Application nadając mu własną nazwę. • Sprawdź (i ewentualnie ustaw) wersję Framework'a aplikacji (tak samo jak w drugim projekcie). • Dodaj w projekcie referencję do System.ServiceModel (tak samo jak w drugim projekcie). • Dodaj referencję serwisową do zdefiniowanej usługi: (Ilustracja dalej na rysunku) <ul style="list-style-type: none"> ○ Uruchom najpierw aplikację hostującą usługę WCF! ○ Zaznacz w Solution Explorer prawym klawiszem folder References i wybierz opcję Add Service Reference. ○ W oknie Add Service Reference, w polu Address wpisz adres usługi (endpoint):

`http://localhost:xxx/NazwaBazowa`

Zamiast **xxx** podaj odpowiedni nr portu.

- Naciśnij przycisk **Go** i powinna pojawić się dostępna usługa na podanym punkcie dostępu (patrz rys. dalej). Wybranie kontraktu (interfejsu) dodatkowo pokaże dostępne operacje (metody).
- Zatwierdź wybór przyciskiem **OK**.

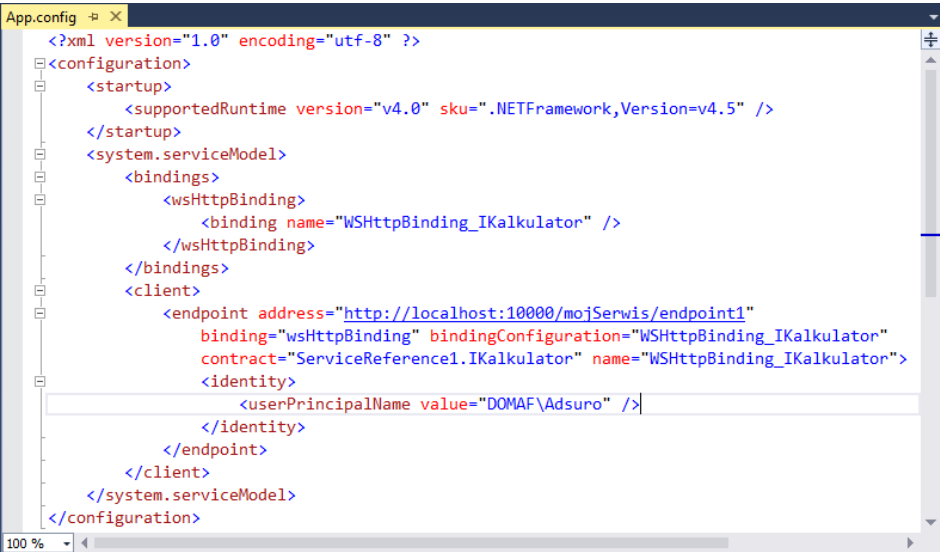
W powyższy sposób generowany jest kod proxy klienta (client proxy) – dołączanego modułu, realizującego wywołania usług.



7. Konfigurowanie klienta

Konfiguracja klienta zawarta jest, w utworzonym poprzez referencję serwisową, pliku konfiguracyjnym **App.config** projektu klienta.

- Otwórz plik **App.config** klienta i przeanalizuj jego zawartość.
- Zwróć uwagę zwłaszcza na sekcję i nazwę i typ wiązania (binding), sekcję i nazwę punktu dostępu (endpoint) oraz specyfikację kontraktu (contract).
- Jego zawartość powinna być podobna do tej na ilustracji.

	 <pre> App.config <?xml version="1.0" encoding="utf-8" ?> <configuration> <startup> <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" /> </startup> <system.serviceModel> <bindings> <wsHttpBinding> <binding name="WSHttpBinding_IKalkulator" /> </wsHttpBinding> </bindings> <client> <endpoint address="http://localhost:10000/mojSerwis/endpoint1" binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IKalkulator" contract="ServiceReference1.IKalkulator" name="WSHttpBinding_IKalkulator"> <identity> <userPrincipalName value="DOMAF\Adsuro" /> </identity> </endpoint> </client> </system.serviceModel> </configuration> </pre>
8. Implementacja klienta	<ul style="list-style-type: none"> Otwórz plik Program.cs i wpisz kod klienta realizujący działania: <ul style="list-style-type: none"> Utworzenie instancji klienta (WCF proxy). Wywołanie operacji usługi z klienta (proxy). Zamknięcia klienta <pre> using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; namespace WcfServiceClient1 { class Program { static void Main(string[] args) { //Krok 1: Utworzenie instancji WCF proxy. CalculatorClient myClient = new CalculatorClient(); // Krok 2: Wywołanie kolejnych operacji usługi. ... double result = myClient.Add(value1, value2); Console.WriteLine(...); ... result = myClient.Sub(value1, value2); Console.WriteLine(...); ... result = myClient.Multiply(value1, value2); Console.WriteLine(...); // Krok 3: Zamknięcie klienta zamyka połączenie i czyszczy zasoby. myClient.Close(); } } } </pre> <ul style="list-style-type: none"> W miejsce kropek wstaw wypisanie odpowiednich komunikaty W miejsce value1 i value2 wpisz jakieś liczby. Usuń błędy poprzez dodanie importu odpowiednich nazw.

<p>9. Testowanie działania aplikacji</p>	<ul style="list-style-type: none"> • Uruchom serwis (aplikację hostującą usługę WCF) w jednym oknie konsoli. • Uruchom klienta w drugim oknie konsoli. • Skontroluj wyniki działania. • Efekt działania klienta i serwisu powinien być podobny do zamieszczonych ilustracji. <div data-bbox="432 495 1441 1128"> </div> <ul style="list-style-type: none"> • Zakończ działanie wszystkich aplikacji.
<p>10. Modyfikacja kontraktu</p>	<ul style="list-style-type: none"> • Dopisz w interfejsie kontraktu jeszcze jedną operację: <code>[OperationContract]</code> <code>double Summarize(double n1);</code> • Dopisz w implementacji kontraktu zmienną globalną w klasie: <code>double sum=0;</code> • Zdefiniuj w implementacji kontraktu operację "trwałego" sumowania <code>public double Summarize(double n1)</code> <code>{</code> <code> sum += n1;</code> <code> return sum;</code> <code>}</code>
<p>11. Modyfikacja hosta serwisu</p>	<ul style="list-style-type: none"> • W projekcie hosta w pliku konfiguracyjnym zdefiniuj dodatkowy punkt końcowy dla serwisu o podanym wiązaniu. Otwórz plik App.config i wpisz w węźle <configuration>, za węzłem <startup>, kod (kolejno): <ul style="list-style-type: none"> ○ Usługę (service) – węzeł <service> – a w nim: ○ Punkt końcowy (endpoint) usługi – węzeł <endpoint>,

	<pre> <system.serviceModel> <services> <service name="WcfServiceContract1.MyCalculator"> <endpoint name="myEndpoint3" address="/endpoint3" binding="wsHttpBinding" contract="WcfServiceContract1.ICalculator"> </endpoint> </service> </services> </system.serviceModel> </pre> <p>W tym pliku można konfigurować wszelkie elementy serwisu.</p> <ul style="list-style-type: none"> Otwórz plik Program.cs hosta i dopisz kod realizujący następujące funkcje: <ul style="list-style-type: none"> Dodanie kolejnego endpointa (dla transportu BasicHttp). Wyświetlenie informacji o kontrakcie. W bloku try {...} <u>przed uruchomieniem serwisu</u> (przed funkcją <code>Open()</code>) utwórz obiekt wiązania <code>BasicHttpBinding</code> (dla transportu Basic Http) i dodaj dodatkowy punkt końcowy/endpoint (jednocześnie pobierając go), oraz pobierz punkt końcowy serwisu zdefiniowany w pliku <code>app.config</code>. Za endpointem 1 dopisz kod: <pre> BasicHttpBinding binding2 = new BasicHttpBinding(); ServiceEndpoint endpoint2 = myHost.AddServiceEndpoint(typeof(IKalkulator), binding2, "endpoint2"); ServiceEndpoint endpoint3 = myHost.Description.Endpoints.Find(new Uri("http://localhost:xxx/MyService/endpoint3")); </pre> Następnie dopisz kod wyświetlający informacje o endpointach (jak poniżej dla endpointa 1), powielając go dla każdego endpointa: <pre> Console.WriteLine("\n---> Endpointy:"); Console.WriteLine("\nService endpoint {0}:", endpoint1.Name); Console.WriteLine("Binding: {0}", endpoint1.Binding.ToString()); Console.WriteLine("ListenUri: {0}", endpoint1.ListenUri.ToString()); </pre>
12. Testowanie działania serwisu	<ul style="list-style-type: none"> Przebuduj (opcja Rebuild) kontrakt usługi i host usługi. Uruchom z konsoli serwis i sprawdź działanie. Zapoznaj się z wyświetlanymi danymi w konsoli hosta.
13. Modyfikacja klienta dla działania z nowym hostem	<ul style="list-style-type: none"> Upewnij się, że działa host i zaktualizuj referencje serwisu w aplikacji klienta: <ul style="list-style-type: none"> zaznacz w oknie solucji prawym klawiszem myszy węzeł Service References → ServiceReference1 i wybierz opcję Update Service Reference. Sprawdź jakie zmiany pojawiły się w pliku App.config klienta.

	<p><i>Kod poniżej wykonuje się ponieważ proxy klienta można tworzyć bez specyfikacji endpointu tylko gdy jest jeden endpoint danego typu. W innym przypadku trzeba zawsze specyfikować endpoint dla proxy.</i></p> <ul style="list-style-type: none"> W pliku Program.cs zakomentuj instrukcję tworzenia proxy klienta bez specyfikacji endpointu w konstruktorze <pre>//CalculatorClient myClient = new CalculatorClient();</pre> W zamian dopisz instrukcje utworzenia proxy klientów dla operacji za pomocą różnych punktów końcowych. Przy tworzeniu klientów specyfikuj nazwę punktu odpowiednio do tych wygenerowanych w pliku App.config. <pre>CalculatorClient client1 = new CalculatorClient("WSHttpBinding_ICalculator"); CalculatorClient client2 = new CalculatorClient("BasicHttpBinding_ICalculator"); CalculatorClient client3 = new CalculatorClient ("myEndpoint3");</pre> <p>Nazwy punktów końcowych (klient 3) oraz nazwy wiązań (klient 1 i 2) muszą być zgodne z nazwami widocznymi w pliku App.config.</p> Stosownie do powyższego zakomentuj instrukcję wywołania operacji Dodaj(...), Odejmij(...), itd. poprzedniego proxy klienta tj. myClient i w zamian dopisz wywołania dla nowych klientów: <pre>result = clientx.operacja(...);</pre> <p>gdzie x- nr klienta, <i>operacja(...)</i> – to wywołanie Add(...), Sub(...), Multiply(...)...</p> Dopisz wywołanie operacji Summarize dwukrotnie dla każdego proxy klienta (czyli przez wszystkie punkty końcowe). <pre>result = clientx.Summarize(jakaś_wartość);</pre> <p>gdzie x- nr klienta, <i>jakaś_wartość</i> – wartość liczbowa.</p> Dopisz wyświetlanie odpowiednich komentarzy i wyników działań w konsoli klienta.
14. Testowanie działania usługi	<ul style="list-style-type: none"> Uruchom w konsoli klienta (pamiętaj wcześniej uruchomić serwis). Zwróć uwagę na poszczególne wartości wyników sumowania (czyli operacji Summarize(...)). Zatrzymaj serwis i klienta. W pliku Service1.cs kontraktu usługi (implementacji kontraktu) dopisz tuż przed definicją klasy instrukcję specyfikującą zachowanie serwisu jako pojedynczej instancji dla wszystkich klientów: <pre>[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]</pre> Przebuduj (opcja Rebuild) kontrakt i host usługi.

	<ul style="list-style-type: none">• Uruchom z konsoli serwis.• Uaktualnij referencję serwisową klienta i przebuduj klienta.• Uruchom w konsoli klienta (dwukrotnie) i ponownie zwróć uwagę na poszczególne wartości wyników sumowania – zmianę w porównaniu do poprzedniej wersji usługi.
--	--

3 Zadanie – część II

A. Przećwicz technikę tworzenia serwisów i klientów WCF.

1. Definiowanie i implementacja kontraktów.
2. Tworzenie hosta usługi.
3. Definiowanie punktów końcowych (endpoint) o określonych adresach.
4. Definiowanie sposobu komunikacji (BasicHttp, WSHttp, NetTCP).
5. Tworzenie klienta, wiązanie i wywoływanie operacji usług.

B. Przygotować się do napisania na zajęciach aplikacji o podobnych funkcjonalnościach według wskazówek prowadzącego.