WCF SOAP
Tutorial do ćwiczenia

Paulina Sadowska, Rafał Araszkiewicz

13 października 2016

1. Szablon projektu

Szablon projektu znaleźć można pod dwoma alternatywnymi adresami:

- https://www.dropbox.com/s/v5c3958bml39ksw/WcfServiceEmptyDemo.zip?dl=0
- https://github.com/PaulinaSadowska/WcfServiceDemo

Projekt zawiera pusty interfejs *IService* oraz implementującą go klasę *Service*. Początkowy kod przedstawiono na poniższym listingu.

```
namespace WcfServiceEmptyDemo
{
    public interface IService
    {
    }
}
```

```
namespace WcfServiceEmptyDemo
{
    public class Service : IService
    {
    }
}
```

2. Przesyłanie typów prostych

Pierwszym zadaniem będzie dodanie do usługi metody pobierającej imię, nazwisko oraz wiek użytkownika i zwracającej tekst z zdaniem opisującym ta osobę w formie **"{imie} {nazwisko} ma {wiek} lat"**.

1. oznacz interfejs IService adnotacją **[ServiceContract]**
2. do interfejsu dodaj metodę **PrintUserData** przyjmującą imię, nazwisko i wiek użytkownika, a zwracającą **string**. Metodę tą oznacz adnotacją **[OperationContract]**.
3. do klasy Service dodaj odpowiednią implementację metody

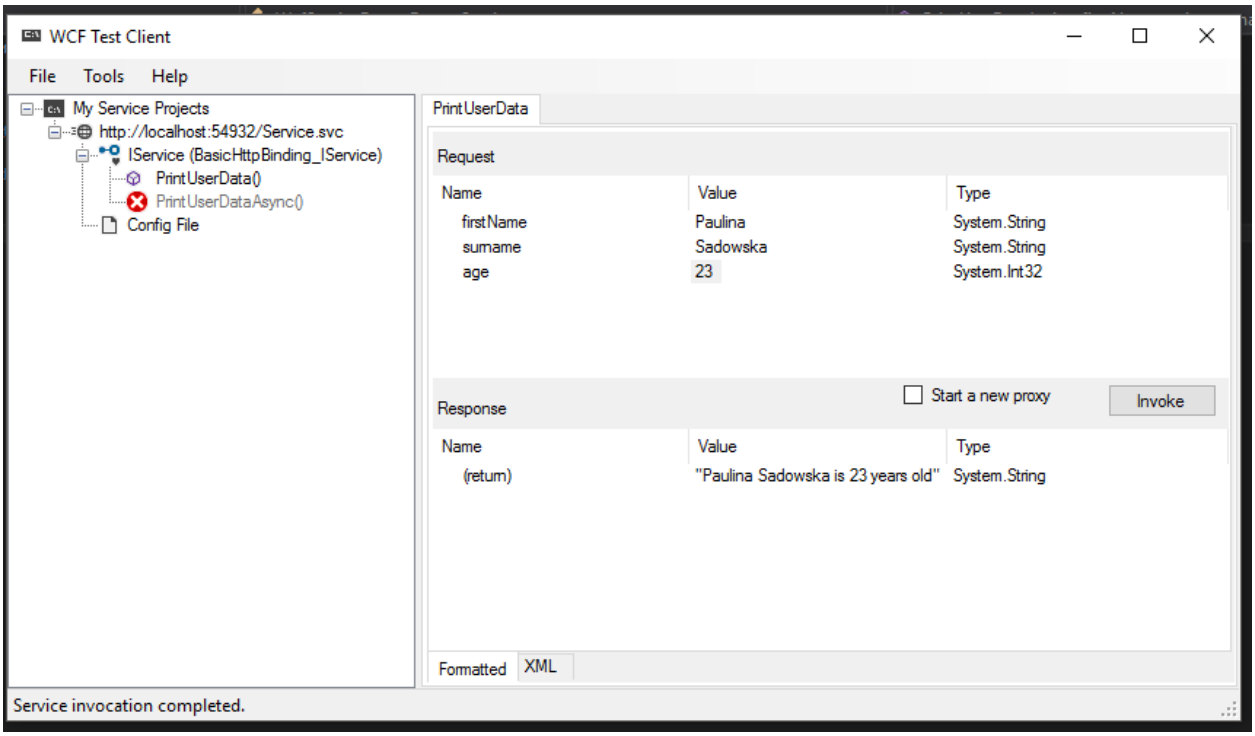
Po wykonaniu powyższych kroków powinieneś otrzymać kod analogiczny do poniższego:

```
using System.ServiceModel;

namespace WcfServiceEmptyDemo
{
    [ServiceContract]
    public interface IService
    {
        [OperationContract]
        string PrintUserData(string firstName, string surname, int age);
    }
}
```

```
using System.ServiceModel;

namespace WcfServiceEmptyDemo
{
    public class Service : IService
    {
        public string PrintUserData(string firstName, string surname, int age)
        {
            return $"{firstName} {surname} is {age} years old";
        }
    }
}
```

3. Sprawdzanie działania serwisu

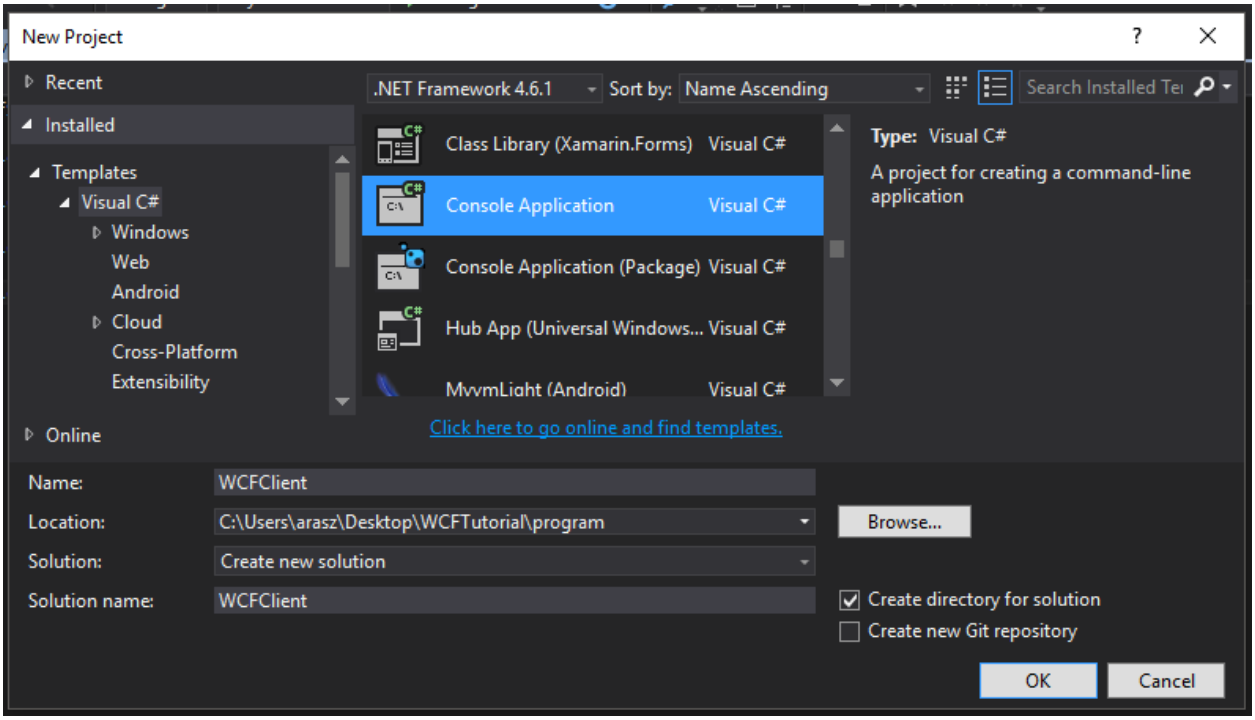
Działanie serwisów w WCF bardzo łatwo sprawdza sie w Visual Studio za pomocą **WCF Test Client**. Otwiera sie on zaraz po odpaleniu serwisu jeżeli otwarty jest plik **Service.svc.cs**. Po wybraniu metody **PrintUserData()** z prawej strony okna w tabeli pojawiają się argumenty jakie przyjmuje metoda. Po uzupełnieniu wartości i wcisnięciu przycisku **Invoke** otrzymujemy odpowiedź serwisu.



Rysunek 1: WCF Test Client

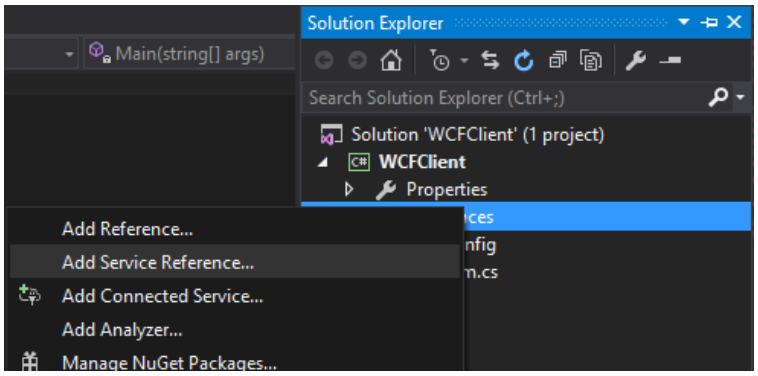
4. Prosta aplikacja kliencka

Do utworzenia prostej aplikacji klienckiej utworzyć należy pusty projekt aplikacji konsolowej w Visual Studio.



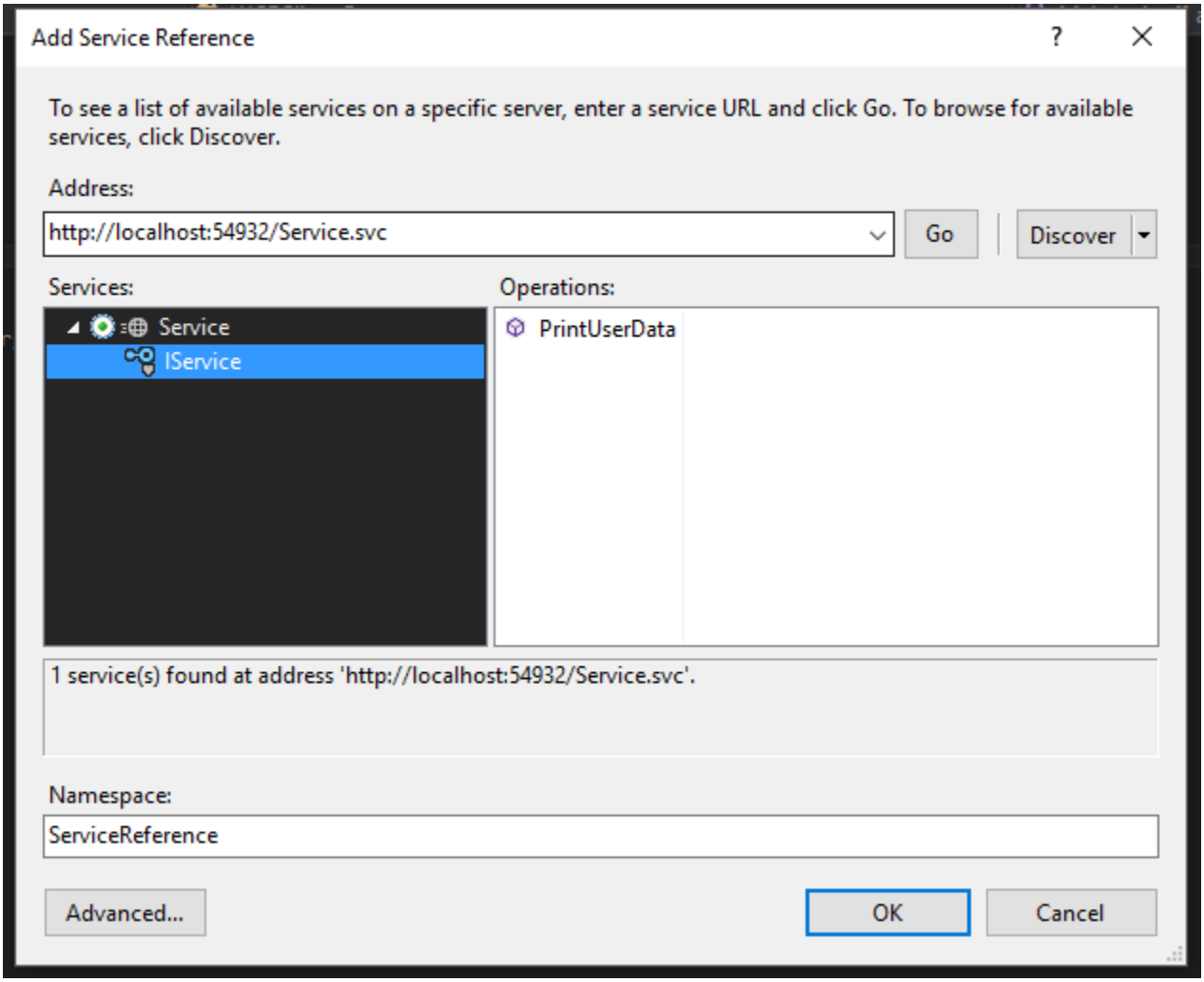
Rysunek 2: Tworzenie aplikacji klienckiej

Do projektu dodaj referencję do utworzonego wczesniej serwisu przez wciśnięcie prawym przyciskiem **References** i wybranie opcji **Add Service Reference**



Rysunek 3: Dodanie referencji do serwisu

W otwartym oknie podaj adres serwisu. Aby go uzyskać uruchomić należy serwis w przeglądarce i skopiować link do pliku serwisu. Jeżeli serwis i klient należą do jednego *Solution* wykorzystać można opcję **Discover**. Po wpisaniu adresu wciśnij przycisk **Go** aby wyszukać serwis pod podanym adresem.

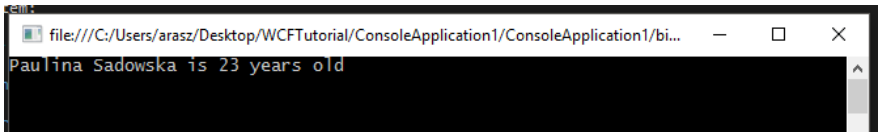


Rysunek 4: Dodanie referencji do serwisu - podanie adresu

Po wciśnieciu przycisku *OK* referencja do serwisu zostaje dodana do projektu klienta.
Nastepnie w metodzie **Main**:

1. utwórz obiekt klienta (klasa **ServiceClient**))
2. wywołaj metodę **PrintUserData** znujdującą się w klasie **ServiceClient** i wyświetl wynik w konsoli
3. zamknij klienta wywołując na nim metodę **close)**

Po odpaleniu programu wyświetlić powinna sie odpowiedź z serwisu podobna do przedstawionej na poniższym zdjeciu:



Rysunek 5: Aplikacja kliencka - wyswietlenie odpowiedzi z serwera

5. Przesyłanie typów złożonych

W następnym kroku do serwisu dodać należy metodę o identycznym działaniu jak **PrintUserData**, ale pobierającą jako argument obiekt **UserData** zawierający informację o imieniu, nazwisku i wieku użytkownika.

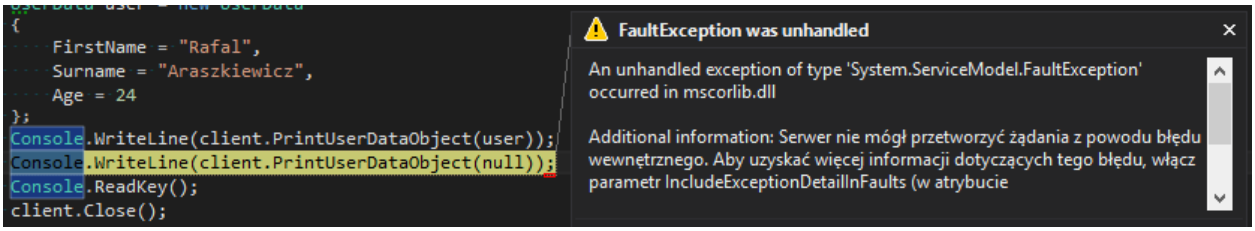
1. utwórz klasę **UserData** i oznacz ją adnotacją **[DataContract]**
2. dodaj do niej atrybuty: imie, nazwisko i wiek wraz z getterami is setterami
3. oznacz atrybuty adnotacją **[DataMember]**

```
using System.Runtime.Serialization;

namespace WcfServiceEmptyDemo
{
    [DataContract]
    public class UserData
    {
        [DataMember]
        public int Age { get; set; }

        [DataMember]
        public string FirstName { get; set; }

        [DataMember]
        public string Surname { get; set; }
    }
}
```

4. do interfejsu **IService** dodaj metodę **PrintUserDataObject** przyjnującą jako argument obiekt typu **UserData**
5. zaimplementuj metodę **PrintUserDataObject** w klasie **Service**
6. sprawdź działanie serwisu za pomocą **WCF Test Client**
7. dopisz do aplikacji klienta wywołanie metody **PrintUserDataObject** i wyswietl w konsoli odpowiedź serwisu

6. Obsługa błędów

Jeżeli obiekt klasy **UserData** przekazany do metody **PrintUserDataObject** będzie miał wartość null w klasie serwisu wystąpi wyjątek. Wywołanie takiej sytuacji w programie klienta spowoduje otrzymanie wyjątku o tresci pokazanej na poniższym zdjeciu:



Rysunek 6: wystąpienie NullPointerException widziane przez klienta

Jak widać klient nie otrzymuje w tez sposób żadnej informacji o tym co było powodem błędu.

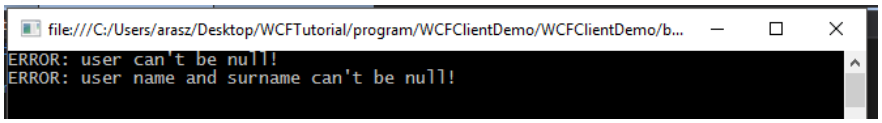
Ostatnim zadaniem będzie więc zmodyfikowanie metody **PrintUserDataObject** tak by klient otrzymał informacje o błędzie, jeżeli obiekt klasy **UserData** wynosi null lub nie ustawiono imienia lub nazwiska użytkownika.

1. w metodzie **PrintUserDataObject** sprawdź czy podane dane są poprawne (czy user nie wynosi null i czy podano imię i nazwisko użytkownika). Jeżeli nie, wyrzuć wyjątek typu **FaultException()** z odpowiednim komunikatem błędu jako argument

2. dodaj do aplikacji klienckiej wywołania metody **PrintUserDataObject** z danymi powodującymi wyjątki i zobacz jak zmienił się otrzymany komunikat

3. przechwyć wyjątek i wyświetl w konsoli komunikat błędu

Po wykonaniu tych kroków w konsoli powinny wyswietlić się komunikaty podobne do pokazanych na obrazku poniżej:



Rysunek 7: wystąpienie NullPointerException widziane przez klienta

Aby ułatwić obsługę wyjątków które mogą wystapić na serwerze oprócz komunikatu błędu do **FaultException** podać można również obiekt klasy **FaultCode**. W aplikacji klienta można wtedy oprócz jedynie wyświetlenia komunikatu odpowiednio obsłużyć konkretny kod błędu.