**Programowanie w .NET.**

**Ćwiczenie praktyczne №6 (2 godz.)**

**Cel pracy:** Utworzenie i wykorzystanie rozproszonych systemów dystrybuowania danymi na platformie .NET, z wykorzystaniem WebService oraz WCF. Użycie elementów języka C# dostępnych od wersji .NET Framework 3.5.

1. **Zapoznać się z następnymi podstawami:**

* Ogólne założenia systemów rozproszonych
* Udostępnianie usług i danych za pomocą WebService
* Technologie systemów rozproszonych DCOM, .NET Remoting, WCF
* Komunikacja międzyplatformowa
* Protokoły transmisji danych

1. **WebService (+ metody rozszerzeń)**

Usługi sieciowe, znane również pod angielską nazwą WebServices to uniwersalny sposób wymiany danych oparty najczęściej na SOAP. Każda usługa powinna być opisana w języku WSDL. Dzięki wykorzystaniu SOAP czy też XML usługi sieciowe są często stosowane w rozproszonych aplikacjach napisanych w różnych językach.

* 1. Utwórz nową solucjęVisual Studio z projektem ASP.NET Web Service Aplication. Aby ten szablon projektu był dostępny wybierz w górnym menu .NET 3.5 Framework. Nazwij projekt UslugaSieciowa. Zauważ, że szablon Visual Studio zawiera już przykładową metodę.
  2. Dodaj nową metodę zgodnie z listingiem 1. Metoda ta to bardzo specyficzna, nie praktyczna funkcja skrótu, która zwraca 10 pierwszych znaków jeśli treść jest dłuższa niż 10 znaków, w przeciwnym wypadku uzupełnia do 10 zadanym znakiem. Uruchom solucję, w przeglądarce będzie możliwość sprawdzenia jak działa usługa. Na główniej stronie kliknij FunkcjaSkrotu i sprawdź jej działanie.

[WebMethod]

public string FunkcjaSkrotu(string text)

{

if (text.Length >= 10)

return text.Substring(0, 10);

else

return text.PadRight(10, 'X');

}

Listing 1

* 1. Dodaj nową klasę o nazwie RozszerzeniaString. Treść klasy uzupełnij zgodnie z listingiem 2. Jest to rozszerzenie do klasy string. Rozszerzenia nie modyfikują klas, które rozszerzają. Umożliwiają dodanie pewnych funkcji do już istniejących (i zamkniętych na zmiany) klas. Metoda rozszerzenia przyjmuje w parametrze specyficzny parametr "this" oraz określenie typu. Przekompiluj projekt. I zmień metodę w klasie web serwisu zgodnie z listingiem 3.

public static class RozszerzeniaString

{

public static string Do10Znakow(this string text)

{

if (text.Length >= 10)

return text.Substring(0, 10);

else

return text.PadRight(10, 'X');

}

Listing 2

[WebMethod]

public string FunkcjaSkrotu(string text)

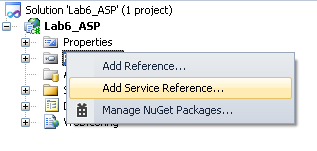
{

return text.Do10Znakow();

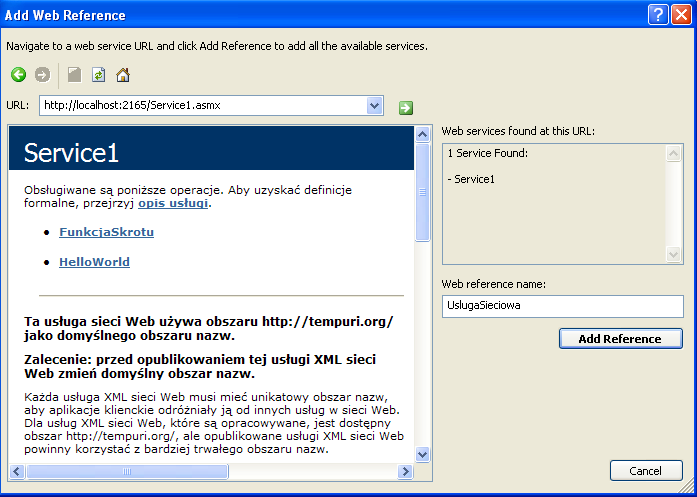
}

Listing 3

* 1. Dodaj własne metody od WebService, używając innych typów danych niż przedstawionych wcześniej. Podstawowe typy danych są uniwersalne dla wielu platform. Dla prezentacji pozostałych typów danych lub utworzenia własnych klas i struktur można wykorzystać XML, np. metoda może zwracać obiekt XmlDoxument. Ponadto do wielu rozwiązań można zastosować mechanizm serializacji.
  2. Uruchom solucję z WebService, następnie utwórz nową solucję VisualStudio z dowolną aplikacją .NET (WinForms, WPF, konsolowa, ASP.NET). W nowym projekcie dodaj referencję do serwisu, patrz rysunek 1. Następnie kliknij Advanced, następnie Add Web Reference. W nowym oknie wprowadź adres utworzonego wcześniej WebService. Adres ten możesz skopiować z przeglądarki. Naciśnij zieloną strzałkę w prawo aby pobrać informacje o tym serwisie (dostarczone za pomocą WSDL), zmień nazwę referencji na UsługaSieciowa, jak na rysunku 2. Naciśnij AddReference. W aplikacji z tej solucji dodaj pole tekstowe do wprowadzania tekstu, przycisk do wykonania akcji, oraz etykietę do wyświetlenia wyniku. Uzupełnij kod zdarzenia na naciśnięcie przycisku wzorując się na listingu 4.



Rysunek 1 - dodawanie referencji do projektu



Rysunek 2 - wskazanie adresu serwisu

UslugaSieciowa.Service1 serwis = new UslugaSieciowa.Service1();

Label1.Text = serwis.FunkcjaSkrotu(TextBox1.Text);

Listing 4

* 1. Dodaj obsługę pozostałych metod WebService.

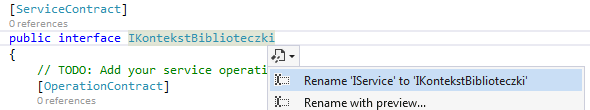
1. **WCF (+ typy anonimowe, metody anonimowe, Linq, wyrażenia Lambda)**

Odpowiedzią Microsoftu na wiele różnych standardów komunikacji sieciowej (w tym rozproszonej) jest jedna uniwersalna technologia, która jest w stanie zastąpić (i współdziałać) większość dotychczasowych technologii sieciowych, a ponadto dostarcza nowe rozwiązania. Technologia ta ułatwia programiście wiele aspektów związanych z komunikacją, aby ten mógł skupić swoją uwagę na dziedzinie problemowej.

Podstawowy system oparty na technologii WCF składa się z trzech elementów: pakietu usług WCF, mechanizmu hostującego usługę, klienta korzystającego z usługi. Duża uniwersalność niesie za sobą mozolną i często trudną konfigurację, dlatego w poniższym przykładzie zostanie wykorzystana przykładowa konfiguracja, bez ingerencji w parametry techniczne. Podstawowa konfiguracja WCF zakłada wykorzystanie adresu, wiązania oraz kontraktu, co jest nazywane często ABC (skrót od angielskich słów). Adres określa po prostu adres serwisu oraz definiuje formę komunikacji np. HTTP lub TCP. Wiązanie określa sposób wymiany danych, w tym różne parametry, np. zabezpieczenia, transakcyjność, kodowanie. Kontrakt stanowi definicję danych, które są wymieniane oraz udostępniane przez usługę. Całość ABC tworzy punkt końcowy (Endpoint).

Sugerowana dekompozycja na warstwy to wydzielenie biblioteki zawierającej jedynie interfejsy i hostowanie jej poprzez WFC. W osobnej bibliotece powinny znaleźć się implementacje tych interfejsów. W poniższym przykładnie zostanie to uproszczone do jednej biblioteki.

* 1. Utwórz nową solucję z projektem WCF Service Library. Nazwij ją BiblioteczkaWCF. Szablon tego projektu domyślnie utworzy przykładowy interfejs usługi oraz jego implementację. Dokonaj zmian tej implementacji zgodnie z listingiem 5, 6 i 7. Aby zmienić nazwę klasy, metody czy pola we wszystkich powiązanych wystąpieniach skorzystaj z podpowiedzi z rysunku 3. W interfejsie IKontekstBiblioteczki zawarte są deklaracje możliwych funkcji. Aby żądana funkcja była dostępna poprzez usługę WCF należy oznaczyć ją adnotacją [OperationContract]. Klasa Ksiazka, która opisuje model danych książki również wymaga dodania adnotacji, są to odpowiednio [DataContract] dla wskazania, że cala klasa należy do kontraktu, oraz [DataMember] aby określić dostępne dane w tej klasie. Implementacja interfejsu zawarta w klasie KontekstBiblioteczki jest odpowiednikiem stosowanego wcześniej kontekstu (np. w ćwiczeniu ASP.NET MVC). Uruchom solucję, automatycznie uruchomi się aplikacja WCF Test Client - wykorzystaj tę aplikację do przetestowania podstawowych funkcji serwisu. Jako pierwszą wywołaj metodę pobierającą wszystkie książki, spróbuj też dodać coś do kolekcji.



Rysunek 3 - zmiana nazwy wraz ze zmianą nazw wystąpień

[ServiceContract]

public interface IKontekstBiblioteczki // wcześniej IService

{

// TODO: Add your service operations here

[OperationContract]

void Dodaj(string tytul, string autor, int iloscStron);

[OperationContract]

void Edytuj(int id, string tytul, string autor, int iloscStron);

[OperationContract]

void Usun(int id);

[OperationContract]

Ksiazka Pobierz(int id);

[OperationContract]

List<Ksiazka> PobierzWszystkie();

}

Listing 5

[DataContract]

public class Ksiazka

{

private int id;

[DataMember]

public int ID

{

get { return id; }

set { id = value; }

}

private string tytul;

[DataMember]

public string Tytul

{

get { return tytul; }

set { tytul = value; }

}

private string autor;

[DataMember]

public string Autor

{

get { return autor; }

set { autor = value; }

}

private int iloscStron;

[DataMember]

public int IloscStron

{

get { return iloscStron; }

set { iloscStron = value; }

}

public Ksiazka(int id, string tytul, string autor, int iloscStron)

{

this.id = id;

this.tytul = tytul;

this.autor = autor;

this.iloscStron = iloscStron;

}

public override string ToString()

{

return string.Format("{0} - \"{1}\"", Autor, Tytul);

}

}

Listing 6

public class KontekstBiblioteczki : IKontekstBiblioteczki // wcześniej Service

{

private List<Ksiazka> kolekcjaKsiazek;

public KontekstBiblioteczki()

{

kolekcjaKsiazek = new List<Ksiazka>();

ZaladujDane();

}

private void ZaladujDane()

{

this.Dodaj("Pan Tadeusz", "Adam Mickiewicz", 800);

this.Dodaj("Agile. Programowanie zwinne", "Robert C. Martin", 700);

this.Dodaj("Czysty kod", "Robert C. Martin", 400);

}

public void Dodaj(string tytul, string autor, int iloscStron)

{

int id = this.kolekcjaKsiazek.Count + 1;

this.kolekcjaKsiazek.Add(new Ksiazka(id, tytul, autor, iloscStron));

}

public void Edytuj(int id, string tytul, string autor, int iloscStron)

{

Ksiazka ks = Pobierz(id);

ks.Tytul = tytul;

ks.Autor = autor;

ks.IloscStron = iloscStron;

}

public void Usun(int id)

{

Ksiazka ks = this.Pobierz(id);

if (ks != null)

this.kolekcjaKsiazek.Remove(ks);

}

public Ksiazka Pobierz(int id)

{

var ksiazka = (from k in this.kolekcjaKsiazek where k.ID == id select k).FirstOrDefault();

return ksiazka as Ksiazka;

}

public List<Ksiazka> PobierzWszystkie()

{

return this.kolekcjaKsiazek;

}

}

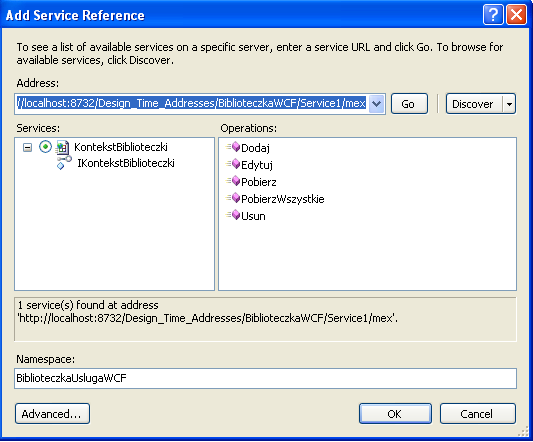
Listing 7

* 1. Usługa WCF jest już gotowa do wykorzystania. Hostowanie takich usług możliwe jest na trzy sposoby:
* Self hosting - polega na stworzeniu prostej aplikacji (np. konsolowej), która hostuje usługę poprzez wykorzystanie klasy ServiceHost;
* IIS - poprzez wykorzystanie serwera aplikacji IIS, takie rozwiązanie daje większe możliwości w stosunku do Self Hosting i zapewnia większe bezpieczeństwo;
* Usługa Windows - polega na dodaniu usługi WCF do działa w tle jako usługa windows (w taki sposób działa np. Microsoft SQL Server).

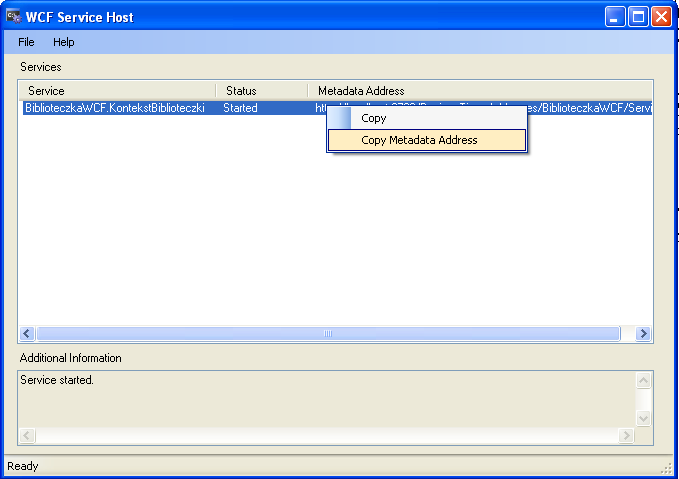
Każdy z tych sposbów wymaga odpowiedniego przygotowania pliku konfiguracyjnego, co wykracza poza ramy tego ćwiczenia. Opisu i podpowiedzi należy szukać w dokumentacji. W celu łatwiejszej realizacji następnych kroków ćwiczenia wyłącz uruchamianie aplikacji WcfTestClient. Otwórz właściwości projektu BiblioteczkaWCF, przejdź do zakładki Debud i wykasuj następującą treść z pola Command line arguments: /client:"WcfTestClient.exe". Zapisz zmiany.

* 1. Utworzenie aplikacji klienckiej i metody anonimowe

Utwórz nową aplikację .NET dowolnego typu (ASP.NET, konsolowa, WinForms, WPF), lub zaadoptuj odpowiednio aplikację z ćwiczenia dotyczącego ASP.NET MVC. Dodaj referencję sieciową, jak przedstawiono na rysunku 4. Aby zdobyć adres usługi odszukaj w zasobniku systemowym (tzw. tray’u, koło godziny) ikonę WCF Service Host i uruchom. Skopiuj adres usługi jak przedstawiono na rysunku 5. Skorzystanie z usługi polega na utworzeniu jej instancji, jak w przykładowym listingu 8 (nie zapomnij o using KlientWCF.BiblioteczkaUslugaWCF;). Wykonaj teraz implementację dla pozostałych funkcji udostępnianych przez usługę. Przed implementacją możesz zapoznać się z pozostałymi punktami ćwiczenia.



Rysunek 4 - dodanie referencji do serwisu WCF



Rysunek 5 - kopiowanie adresu usługi WCF

KontekstBiblioteczkiClient client = new KontekstBiblioteczkiClient();

var ksiazki = client.PobierzWszystkie();

Listing 8

* 1. Typy anonimowe - zwróć uwagę na typ danych na listingu 8. Typ danych var jest typem anonimowym, co oznacza, że programista w momencie pisania kodu programu nie musi zwracać uwagi jaki typ danych zostanie zwrócony. Kompilator sam dokona odpowiedniej analizy i dostosuje ten typ danych. Typy anonimowe muszą mieć wartość i muszą być zainicjalizowane w chwili deklaracji, zapisanie linijki var zmienna; poskutkuje błędem. Język C# charakteryzuje się silną kontrolą typów, a typ danych anonimowych pozwala na zmniejszenie znaczenia tej cechy. Ma to szczególne zastosowanie gdy jakaś biblioteka (np. dostarczona przez zewnętrznego producenta) może zmieniać swój interfejs bez zmian w polach i metodach. Zastosuj typy anonimowe w kodzie aplikacji klienckiej i usługi WCF.
  2. Metody anonimowe - na listingu 9 przedstawiony jest prosty przykład metody anonimowej. Zdarza się, że jakaś metoda w kodzie programu wywoływana jest tylko przez jedno zdarzenie, np. kliknięcie przycisku. Z różnych powodów można nie implementować osobnej metody dla takiego pojedynczego zdarzenia i skorzystać z metody anonimowej. Wykonywany kod należy wpisać w nawiasy klamrowe. Zamień kod obsługi podstawowych zdarzeń na metody anonimowe.

button1.Click += delegate(Object o, EventArgs e)

{

dataGridView1.DataSource = client.PobierzWszystkie();

};

Listing 9

* 1. Zapytania Linq i wyrażenia lambda. W tym ćwiczeniu jak też w wcześniejszych ćwiczeniach przykładowe kody były zapisane z użyciem zapytań Linq. Listing 10 przedstawia kod metody pobierającej książkę o konkretnym ID. Listing 11 przedstawia ten sam kod wzbogacony o wyrażenie Lambda. Zwróć uwagę na różnicę w składni, w przypadku Linq należy posłużyć się słowami kluczowymi znanymi z języka SQL. W przypadku wyrażeń Lambda należy określić parametr "k", następnie umownie wskazać strzałką "=>" co należy wykonać, w przypadku listingu 11 spełnienie parametru k.ID == id. Przy czym litera "k" jest jedynie przykładem, można zastosować inne nazwy, jak też parametrów może być więcej. Zaimplementuj dodatkowe metody w usłudze WCF pozwalające na zwrócenie kolekcji według specyficznych kryteriów (np. ilość stron, tytuł zaczynający się od słów, połączenie dwóch parametrów wyszukiwania itp). Zaimplementuj te metody w zapisie Linq i Lambda.

public Ksiazka Pobierz(int id)

{

var ksiazka = (from k in this.kolekcjaKsiazek where k.ID == id select k).FirstOrDefault();

return ksiazka as Ksiazka;

}

Listing 10

var Linq = (from k in this.kolekcjaKsiazek where k.ID == id select k).FirstOrDefault();

var Lambda = this.kolekcjaKsiazek.Where(k => k.ID == id).FirstOrDefault();

Listing 11

Literatura.

1. Jesse Liberty, C#. Programowanie, Helion, 2005
2. Wei-Meng Lee, C# 2008. Warsztat programisty, Helion, 2010
3. Troelsen Andrew, Język C# 2010 i platforma .NET 4, Helion, 2011
4. Paweł Chłosta. ASP.NET i kolekcje C#, MIKOM, 2004-10-01
5. Kevin Hoffman. Visual C# 2005, Helion, 2007
6. Jess Chadwick, Todd Snyder, Hrusikesh Panda, ASP.NET MVC 4. Programowanie, Helion 2013
7. Sławomir Orłowski, Maciej Grabek, C# Tworzenie aplikacji sieciowych, Helion 2012.