

Platforma .NET – Wykład 13 Tworzenie usług sieciowych SOAP i WCF

Osoba prowadząca wykład, laboratorium i projekt:
dr hab. inż. Marek Sawerwain, prof. UZ

Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

e-mail : M.Sawerwain@issi.uz.zgora.pl
tel. (praca) : 68 328 2321,
pok. 328a A-2,
ul. Prof. Z.Szafrana 2,
65-246 Zielona Góra

Ostatnia kompilacja pliku: Monday 5th June, 2023, t: 23:36

V1.00 – 1/ 81

Notatki

Spis treści

- Wprowadzenie
 - Plan wykładu
- Usługi rozproszone, czyli usługi sieciowe
 - Idea aplikacji rozproszonych
 - Istniejące technologie
 - Usługi sieciowe jako aplikacje rozproszone
- Protokół SOAP
 - Przeznaczenie SOAP
 - Elementy protokołu
 - Przykład zapytania
 - Web Services Description Language (WSDL)
- SOAP .NET oraz WCF
 - Usługi sieciowe w protokole SOAP
 - Wstęp do technologii WCF – część I
 - Architektura WCF
 - Trzy główne pojęcia WCF
- WCF – spojrzenie ogólne
 - Podstawowe pojęcia
- Praktyka WCF
 - Definicja kontraktu usługi
 - Udostępnienie (hostowanie) usługi
 - Klient korzystający z usługi
 - Przepływ transakcyjny
- Zakończenie

V1.00 – 2/ 81

Notatki

Plan wykładu – spotkania tydzień po tygodniu

- (1) Informacje o wykładzie, pojęcie platformy, podstawowe informacje o platformie .NET
- (2) Składowe platformy .NET: CLR, CTS, języki programowania, biblioteki klas, pojęcie podzespołu (ang. assembly)
- (3) Programowanie w C# – środowisko VS, MonoDevelop, syntaktyka C#, wyjątki, współpraca z DLL
- (4) Programowanie w C# – model obiektowy, typy uogólnione, lambda wyrażenia
- (5) Programowanie w C# – aplikacje „okienkowe”, programowanie wielowątkowe
- (6) Programowanie w F# – podstawy, przetwarzanie danych tekstowych,
- (*) "Klasówka I", czyli egzamin część pierwsza
- (7) Dostęp do baz danych

V1.00 – 3/ 81

Notatki

Plan wykładu – tydzień po tygodniu

- (8) Język zapytań LINQ, Entity Framework
- (9) Obsługa standardu XML
- (10) Technologia ASP.NET 1/2
- (11) Technologia ASP.NET 2/2
- (12) Model widok i kontroler – Model View Controller
- (13) Tworzenie usług sieciowych SOAP i WCF (komunikacja sieciowa)
- (14) Wykład monograficzny .NET 1
- (15) Wykład monograficzny .NET 2
- (*) "Klasówka II", czyli egzamin część druga

V1.00 – 4/ 81

Notatki

Plan wykładu – 1/2

1. Usługi rozproszone = Usługi sieciowe
 - 1.1 idea aplikacji rozproszonych,
 - 1.2 usługi sieciowe jako aplikacje rozproszone,
 - 1.3 istniejące technologie,
 - 1.4 SOA – Service-Oriented Architecture (architektura zorientowana na usługi).
2. Usługi sieciowe poprzez protokół SOAP
 - 2.1 definicja protokołu SOAP,
 - 2.2 .NET i SOAP,
 - 2.3 własne dane,
 - 2.4 testowanie usług.
3. Wstęp do technologii WCF
 - 3.1 główna idea WCF,
 - 3.2 przedstawienie architektury,
 - 3.3 główne elementy WCF.

V1.00 – 6/ 81

Notatki

Plan wykładu – 2/2

1. Miejsce WCF
 - 1.1 przegląd rozwiązań,
 - 1.2 miejsce WCF w stosie .NET,
 - 1.3 założenia projektowe WCF.
2. Podstawowe pojęcia
 - 2.1 równanie na WCF,
 - 2.2 najważniejsze pojęcia,
 - 2.3 model programowania,
 - 2.4 kanały komunikacji.
3. Praktyka WCF
 - 3.1 tworzenie usługi,
 - 3.2 „hostowanie” samodzielne, i jako usługa Windows,
 - 3.3

V1.00 – 6/ 81

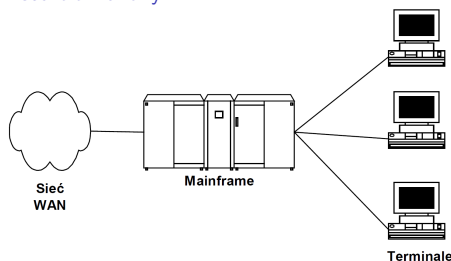
Notatki

Główne etapy rozwoju systemów komputerowych

Można wyróżnić trzy główne etapy:

1. System scentralizowany
2. System sieciowy
3. System rozproszony

System scentralizowany



V1.00 – 7/ 81

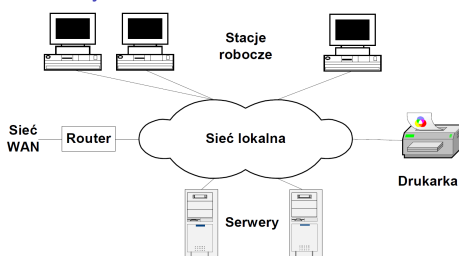
Notatki

Główne etapy rozwoju systemów komputerowych

Można wyróżnić trzy główne etapy:

1. System scentralizowany
2. System sieciowy
3. System rozproszony

System sieciowy



V1.00 – 8/ 81

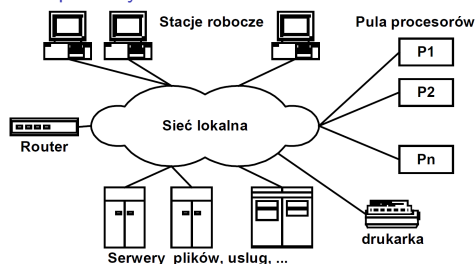
Notatki

Główne etapy rozwoju systemów komputerowych

Można wyróżnić trzy główne etapy:

1. System scentralizowany
2. System sieciowy
3. System rozproszony

System rozproszony



V1.00 – 9/ 81

Notatki

Systemy, aplikacje rozproszone

Opisowa, intuicyjna definicja systemu rozproszonego:

System rozproszony

Zestaw urządzeń fizycznych (komputerów) bądź wirtualnych, które z punktu widzenia innej aplikacji bądź użytkownika stanowią jedną logiczną całość i są nierozróżnialne dla użytkownika takiego systemu.

Istotne cechy systemu rozproszonego:

1. ukrycie różnic pomiędzy poszczególnymi maszynami,
2. ukrycie różnic w sposobie komunikowania się poszczególnych elementów systemu,
3. ukrycie wewnętrznej organizacji systemu,
4. spójny i ujednoczony interfejs interakcji,
5. łatwość skalowania systemu oraz jego rozszerzania.

V1.00 – 10/ 81

Notatki

Architektura systemu rozproszonego

System rozproszony do system budowany w sposób warstwowy (architektura warstwowa):

- ▶ lokalne systemy operacyjne,
- ▶ obecność warstwy pośredniczącej,
- ▶ ujednoczony interfejs dostępu i komunikacji.

Zastosowana architektura pozwala na osiągnięcie następujących założeń:

1. łatwość uzyskania relacji/połączenia użytkownik – zasoby,
2. ukrycie rozproszenia zasobów,
3. otwartość systemu,
4. skalowalność i podatność na modyfikacje.

V1.00 – 11/ 81

Notatki

Technologie rozproszone

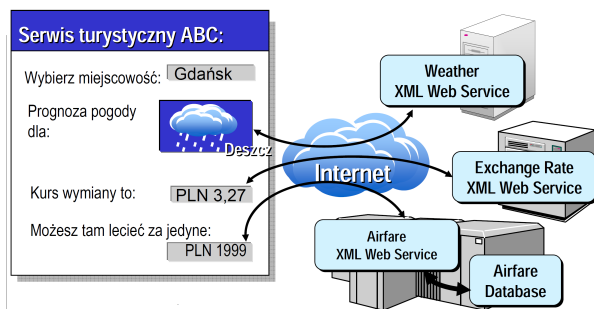
Istniejące technologie tworzenia aplikacji rozproszonych:

- ▶ SOAP (Simple Object Access Protocol), RPC (Remote Procedure Call)
- ▶ REST (REpresentational State Transfer)
- ▶ DCOM (Distributed Component Object Model)
- ▶ CORBA (Common Object Request Broker Architecture)
- ▶ ICE (Internet Communications Engine)
- ▶ Web Services
- ▶ DDS (Data distribution service)
- ▶ Java RMI (Java Remote Method Invocation)
- ▶ WCF (Windows Communication Framework)

V1.00 – 12/ 81

Notatki

Usługi sieciowe jako aplikacje rozproszone



V1.00 – 17/ 81

Notatki

Usługi sieciowe poprzez protokół SOAP

SOAP to protokół oparty o XML, przeznaczony do wymiany informacji ponad protokołem HTTP, krótko mówiąc SOAP to protokół dostępu do usług sieciowych.

Główne cechy i własności protokołu SOAP:

1. SOAP to skrót od Simple Object Access Protocol,
2. SOAP jest protokołem komunikacyjnym pomiędzy aplikacjami,
3. SOAP to format przesyłania komunikatów w sieci Internet,
4. SOAP jest niezależny od platformy i języka,
5. SOAP jest oparty o XML,
6. cechą SOAP jest prostota i łatwość rozszerzania,
7. łatwo także przestać wiadomości SOAP poprzez „firewall-e”
8. SOAP posiada rekomendacje W3C.

V1.00 – 18/ 81

Notatki

Wiadomość SOAP to zwykły dokument XML zawierający następujące elementy:

1. element Envelope – znacznik identyfikujący dokument XML zawierający wiadomość SOAP,
2. element Header – nagłówek zawierający dodatkowe informacje,
3. element Body – główny element zawierający zapytanie i odpowiedź,
4. element Fault – informacje o błędzie i statusie informacji.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
  ...
  </soap:Header>
  <soap:Body>
  ...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

V1.00 – 19/ 81

Notatki

Envelope, Header, Body, Fault

Główne pojęcia SOAP:

1. Envelope – koperta – główny element pliku XML z komunikatem SOAP,
2. Header – nagłówek – opcjonalny element, mogący zawierać elementy specyficzne dla danej aplikacji, obecny jest atrybut mustUnderstand, wymagający poprawnej analizy nagłówka przez serwer czy klienta (istnieją jeszcze dwa inne zdefiniowane atrybuty: actor oraz encodingStyle),
3. Body – ciało dokumentu – właściwa treść lub odpowiedź na zapytanie,
4. Fault – błąd – informacje o błędzie i statusie komunikatu SOAP.

V1.00 – 20/ 81

Notatki

Zapytanie SOAP

Zapytanie SOAP o cenę akcji IBM:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

V1.00 – 21 / 81

Notatki

Otrzymana odpowiedź SOAP

Zapytanie SOAP o cenę akcji IBM:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>
</soap:Envelope>
```

V1.00 – 22 / 81

Notatki

Web Services Description Language (WSDL)

Język opisujący usługę sieciową, główne elementy tego języka to min:

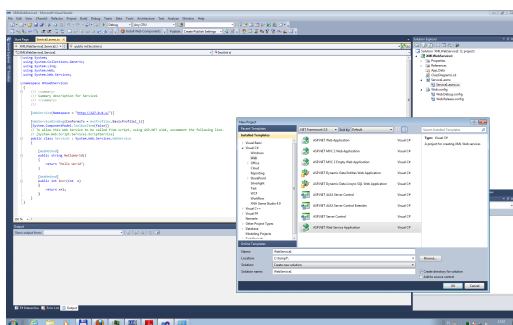
1. types – tag podstawowy <wsdl:types> obejmujący wszystkie definicje typów w danej usłudze sieciowej,
2. message – Wiadomość to dane XML które będą przekazywane pomiędzy usługą sieciową a klientem. Znacznik <wsdl:message> reprezentuje tą wiadomość,
3. portType – Znacznik <wsdl:portType> reprezentuje sekcję zawierającą listę operacji (metod sieciowych) eksponowanych przez usługę,
4. binding – Binding, czyli przyłączy to protokół oraz format użyty przez port, jest reprezentowany przez znacznik <wsdl:binding>,
5. port – Port – punkt końcowy komunikacji z usługą sieciową. Jest reprezentowany przez znacznik <wsdl:port>,
6. service – Usługa to kolekcja jednego lub więcej portów, reprezentowana przez znacznik <wsdl:service>.

V1.00 – 23 / 81

Notatki

Usługi sieciowe w .NET

Tworzenie usługi sieciowej w protokole SOAP do wersji 3.5 .NET (wersje wyższe oferują usługi WCF):



V1.00 – 24 / 81

Notatki

Usługa z „pudełką” – 1/4

Kod źródłowy prawie pustej usługi sieciowej:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Services;  
  
namespace XMLWebService1  
{  
  
    [WebService(Namespace = "http://127.0.0.1/")]  
  
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]  
    [System.ComponentModel.ToolboxItem(false)]  
  
    public class Service1 : System.Web.Services.WebService  
    {  
        [WebMethod]  
        public string HelloWorld()  
        {  
            return "Hello World";  
        }  
    }  
}
```

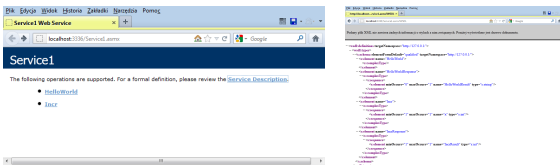
V1.00 – 25 / 81

Notatki

Usługa z „pudełką” – 2/4

Kod źródłowy prawie pustej usługi sieciowej:

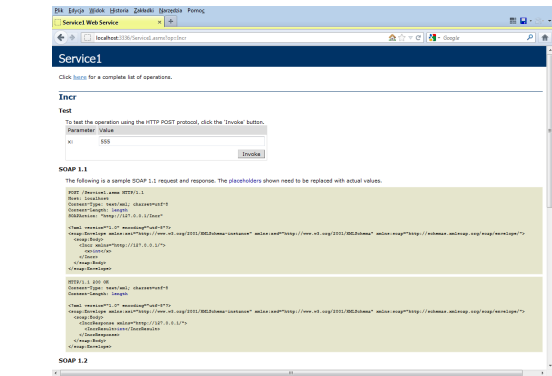
```
[WebMethod]  
public int Incr(int x) { return x+1; }  
  
[WebMethod(MessageName="HelloWorldAgain")]  
public string HelloWorld(string name) {  
    return "Hello " + name;  
}  
}  
}
```



V1.00 – 26 / 81

Notatki

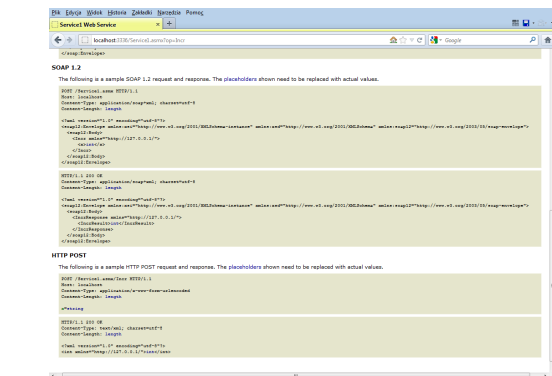
Usługa z „pudełką” – 3/4



V1.00 – 27 / 81

Notatki

Usługa z „pudełką” – 4/4



V1.00 – 28 / 81

Notatki

Inne użyteczne atrybuty dla metod sieciowych

Dodatkowe atrybuty dla metod sieciowych:

1. buforowanie odpowiedzi: [WebMethod(BufferResponse = true)], ustalenie na true powoduje iż dane do klienta będą przesyłane po zakończeniu odpowiedzi, w momencie gdy będzie ona kompletna, zwiększa wydajność dla małych zbiorów danych,
2. tworzenie bufora pomocnicznego zapamiętującego pytania i usyskane odpowiedzi dla podanych argumentów: [WebMethod(CacheDuration = 30)], w argumentcie atrybutu czas przechowywania odpowiedzi w pamięci pomocniczej,
3. utworzenie sesji: [WebMethod(EnableSession=true)], identyfikator sesji jest zapamiętywane jako „ciasteczko”,
4. wprowadzenie transakcji: [WebMethod(TransactionOption = TransactionOption.Required)], możliwe wartości Disabled, NotSupported, Supported, Required (T), RequiresNew (T), wystąpienie wyjątku odwołuje transakcję.
5. opis metody: [WebMethod(Description = "opis")].

Notatki

Przesłanie własnych danych z metody sieciowej

.NET wspiera wiele różnych typów danych np.: DataSet, można także wykorzystywać klasy:

```
public class Employee
{
    private int intID;
    private string strFName;
    private string strLName;
    private string strHPhone;
    private string strNotes;

    public int EmployeeID {
        get { return intID; }
        set { intID = value; }
    }
    public string FirstName {
        get { return strFName; }
        set { strFName = value; }
    }
    public string LastName {
        get { return strLName; }
        set { strLName = value; }
    }
    public string HomePhone {
        get { return strHPhone; }
        set { strHPhone = value; }
    }
    public string Notes {
        get { return strNotes; }
        set { strNotes = value; }
    }
}
```

Notatki

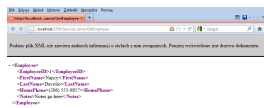
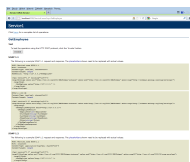
Odpowiednia metoda sieciowa

Wygenerowanie danych o pracowniku sprowadza się tylko do utworzenia odpowiedniego obiektu:

```
[WebMethod]
public Employee GetEmployee() {
    Employee emp = new Employee();

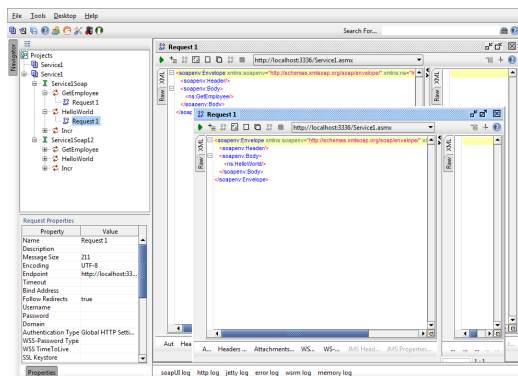
    emp.EmployeeID = 1;
    emp.FirstName = "Nancy";
    emp.LastName = "Davolio";
    emp.HomePhone = "(206) 555-9857";
    emp.Notes = "Notes go here";

    return emp;
}
```



Notatki

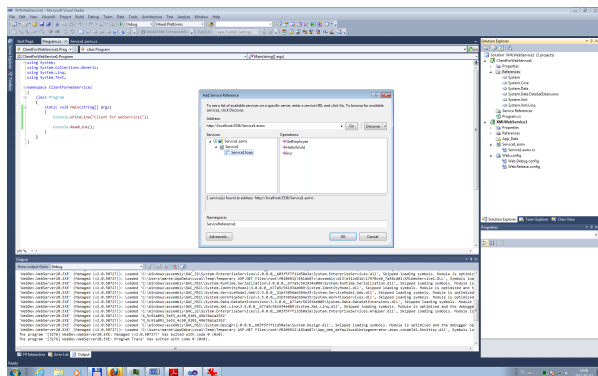
Inne narzędzia do testowania usług SOAP



Notatki

Użycie usługi SOAP w aplikacji .NET

Dodanie referencji do usługi sieciowej:



81

Notatki

Użycie usługi SOAP w aplikacji .NET

Użycie usługi jest następujące:

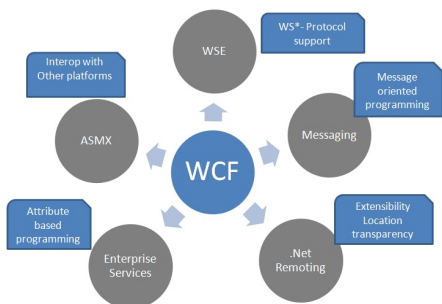
```
CFWS1.SR1.Service1SoapClient  
    srv = new CFWS1.SR1.Service1SoapClient();  
  
var v1 = srv.HelloWorld();  
Console.WriteLine("v1={0}", v1);  
  
var v2 = srv.Incr( 5 );  
Console.WriteLine("v2={0}", v2
```

V1.00 – 34 / 81

Notatki

Windows Communication Foundation

Windows Communication Foundation (WCF) to platforma dla programistów oraz system uruchomieniowy przeznaczony do budowy, konfiguracji oraz rozwoju usług sieciowych. Obecnie jest to jedna z najnowszych technologii zorientowanych na usługi, jej nadrzędną cechą jest interoperacyjność. Dostarcza także ujednoczony model programowania od wersji 3.0 .NET. WCF łączy w sobie następujące technologie: Web Service, Remoting, MSMQ and COM+, co oznacza iż WCF to podstawowa platforma do komunikacji w aplikacjach .NET.



V1.00 – 35 / 81

Notatki

WCF – główne funkcje

Współdziałanie i integracja różnych interfejsów API to dwa ważne aspekty WCF. WCF zapewnia również bogate API uzupełniające technologię zdalnych wywołań (Remoting). List podstawowych funkcji WCF jest następujący:

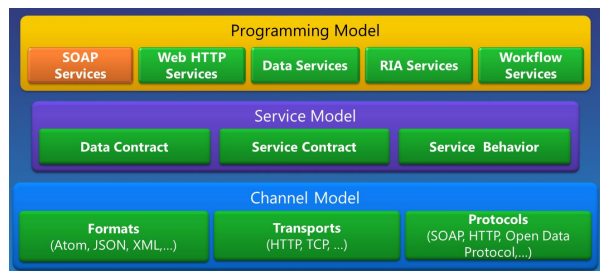
- ▶ Wsparcie dla silnego typowania ale bez-typowe wiadomości także są dopuszczalne. Takie podejście pozwala aplikacją .NET do łatwego dzielenia typów pomiędzy aplikacjami. Takie podejście jest utrudnione w przypadku komunikacji w oparciu o XML.
- ▶ Wsparcie dla różnych przyłączy (np. raw HTTP, TCP, MSMQ, and named pipes), co pozwala na wybranie odpowiedniego kanału komunikacji właściwego do rozwiązywanego zadania.
- ▶ Wsparcie dla specyfikacji usług sieciowych w standardzie (WS-*).
- ▶ Wsparcie dla modelu bezpieczeństwa opartego o natywne rozwiązania Windows/.NET a także możliwe jest stosowanie innych rozwiązań bezpieczeństwa opartych o standardy usług sieciowych.
- ▶ Możliwe jest też stosowanie zarządzanie stanem w trybie sesji, dopuszczalne są komunikaty bezstanowe oraz jednokierunkowe.

V1.00 – 36 / 81

Notatki

Architektura – WCF

Ogólny schemat architektury WCF:

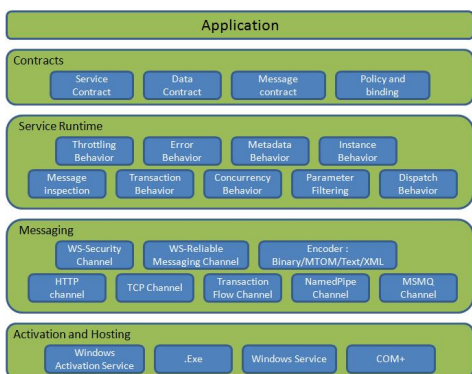


V1.00 – 37 / 81

Notatki

Architektura – WCF

WCF bardziej dokładnie:

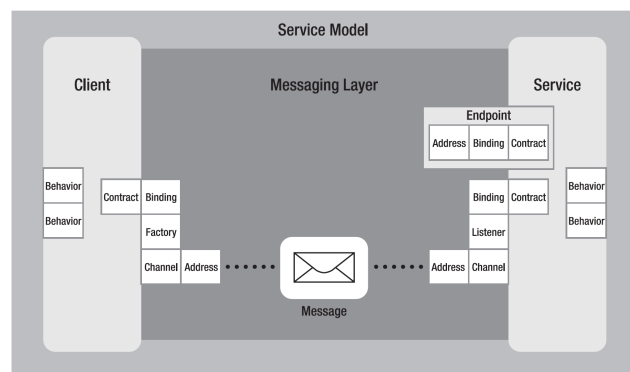


V1.00 – 38 / 81

Notatki

Architektura – WCF

Model programowania WCF:



Adresy (Addresses), przyłącza (Bindings), kontrakty (Contracts)

Adresowanie to podstawowa usługa umożliwiająca poprawne funkcjonowanie usługi sieciowej. Adres jest niezbędny, aby poprawnie wysłać komunikat. Adresy w przypadku WCF to URL'e określające także jaki protokół został zastosowany, adres maszyny gdzie uruchomiono usługę oraz ścieżkę do usługi. Podać można także numer portu, zależny od zastosowanej usługi jednak jest on opcjonalny.

Ścieżka adresowa

scheme://<machinename>[:port]/path1/path2

Główne elementy ścieżki adresowej:

- ▶ Transport scheme – określa protokół/schemat transportu,
- ▶ Machine name – specyfikuje dziedzinę w której znajduje się dana maszyna.
- ▶ Port – opcjonalne pole określające numer portu, wartość 80 to domyślna wielkość dla protokołu HTTP.
- ▶ Path – ścieżka dostępu do usługi, można stosować ścieżki jako katalogi np.: /Stock/GetQuote jest ścieżką w adresie http://localhost:8080/Stock/GetQuote.

V1.00 – 40 / 81

Notatki

Notatki

Adresy (Addresses), przyłącza (Bindings), kontrakty (Contracts)

Przyłącze określa sposób komunikacji z usługą, jest to ważny element programowania w WCF, i można wyróżnić następujące rodzaje typy przyłącza:

- ▶ transport/protokół/schemat (HTTP, MSMQ, Named Pipes, TCP)
- ▶ rodzaj kanału (one-way, duplex, request-reply)
- ▶ kodowanie (XML, binary, Message Transmission Optimization Mechanism [MTOM])
- ▶ wsparcie dla protokołów WS-* (WS-Security, WS-Federation, WS-Reliability, WSTransactions)

WCF dostarcza domyślny zbiór przyłączy, które spełniają większość podstawowych wymagań. Model programowania WCF pozwala na stosowanie także własnych przyłączy dziedziczących z CustomBinding.

V1.00 – 41 / 81

Notatki

Adresy (Addresses), przyłącza (Bindings), kontrakty (Contracts)

Podczas tworzenia usług poprawne określenie zachowania się procesów na tzw. brzegach komunikacji, gdzie warto rozdzielić interfejs od implementacji. Ten rozdział to pojęcie kontraktu w którym wyróżnia się interfejs w którym opisuje się co jest eksponowane w usłudze, detale implementacji są implementowane oddzielnie bez naruszania wcześniej określonego interfejsu. Dzięki kontraktom pojęcie interoperacyjności jest możliwe do osiągnięcia.

Schematy wymiany wiadomości:

- ▶ Request-Reply,
- ▶ One-Way,
- ▶ Duplex.

Kontrakty są obowiązujące dla następujących pojęć: Service and Operations, Data, Message.

V1.00 – 42 / 81

Notatki

WCF – Windows Communication Framework

V1.00 – 43 / 81

Notatki

Przetwarzanie rozproszone

Najważniejsze etapy rozwoju przetwarzania rozproszonego:

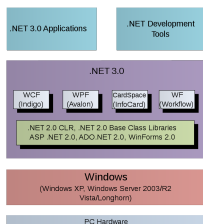
- ▶ 1964 – Dartmouth Time Sharing System, 1969 – First link of ARPANET installed
- ▶ 1974 – First TCP specification, 1978 – TCP/IP specification
- ▶ 1980 – Ethernet
- ▶ 1983 – Berkely sockets released with BSD
- ▶ 1990 – CORBA 1.0
- ▶ 1991 – OLE
- ▶ Early 90's - DCE/RPC
- ▶ 1993 – COM
- ▶ 1994 – CORBA 2.0
- ▶ 1996 – ActiveX, DCOM
- ▶ 1997 – Java RMI (Sun jdk 1.1, Java 2.0)
- ▶ 2002 – .Net Remoting
- ▶ 2006 – WCF 3.0, 2007 – WCF 3.5, 2017 WCF 4.5
- ▶ 2019 - WCF Core 3.1.0 – for client apps
- ▶ 2020 - gRPC for .NET (HTTP/2)

V1.00 – 44 / 81

Notatki

Stos .NET oraz WCF

Miejsce WCF w stosie systemu .NET

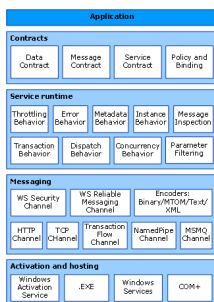


Rysunek pochodzi z: https://en.wikipedia.org/wiki/Windows_Communication_Foundation.

V1.00 – 45 / 81

Notatki

Główne pojęcia WCF



Rysunek pochodzi z: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/architecture>.

V1.00 – 46 / 81

Notatki

Najważniejsze przestrzenie nazw związane z WCF:

- ▶ System.ServiceModel – podstawowe definicje wiązań, sposoby udostępniania, zabezpieczenia,
- ▶ System.ServiceModel.Configuration – API do zmian w konfiguracji,
- ▶ System.ServiceModel.Description – definicja typów stosowanych w aplikacjach WCF,
- ▶ System.ServiceModel.MsmqIntegration – typu związane z systemem MSMQ,
- ▶ System.ServiceModel.Security – typy odnoszące się do kwestii bezpieczeństwa tworzonego serwisu.

Aplikacje WCF wykorzystują również serializację zatem korzysta się również z System.Runtime.Serialization.

V1.00 – 47 / 81

Notatki

Założenia projektowe WCF:

- ▶ jasno określone mechanizmy/protokoły komunikacji,
- ▶ usługi są autonomicznymi bytami (a ich proces zarządzania, uruchamiania również cechuje się niezależnością),
- ▶ podstawowym pojęciem jest kontrakt (nie są akcentowane typy), to kontrakt opisuje zachowanie się, natomiast klient uzyskuje referencję do kontraktu, nie do usługi jako takiej,
- ▶ kompatybilność jest oparta zasadzie rozdzielności zachowania od sposobu dostępu do usługi.

V1.00 – 48 / 81

Notatki

Najważniejsze pojęcia WCF

Za M.Grabek, WCF od podstaw. Komunikacja sieciowa nowej generacji, Helion 2012, wzór na WCF:

- E = A + B + C = WCF.

Rola poszczególnych elementów:

- E – endpoint, punkt końcowy,
- A – adres, adres,
- B – binding, wiązanie,
- C – contract, kontrakt.

Pojęcia E,A,B,C to główne idee na jakich oparto cały WCF. Przy czym, elementy środowe A, B, C stanowią podstawę tworzenia aplikacji, choć dla wielu prostszych zastosowań implementacja może zostać ograniczona tylko do kontraktu: C.

Notatki

Adres

Adres określa miejsce umieszczenia usługi. Ogólny schemat:

- transport://nazwaHosta[:port]/ścieżka/do/serwisu

Dostępne są cztery podstawowe rodzaje transportu:

- http – wykorzystanie standardowego protokołu, np. klasy BasicHttpBinding, WsHttpBinding,
- net.tcp – wykorzystanie protokołu TCP, klasa NetTcpBinding,
- net.msmq – wykorzystanie systemu kolejek MSMQ, klasa NetMsmqBinding,
- net.pipe – nazwane potoki, wykorzystywane do komunikacji w ramach jednej maszyny.

Przykładowe adresy:

- http://localhost:8733/SampleWCFFDotNetLecture,
- net.tcp://localhost:8733/SampleWCFFDotNetLecture,
- net.msmq://localhost/private\$/SampleWCFFDotNetLectureMsmq,
- net.pipe://localhost/SampleWCFFDotNetLecture.

Notatki

Wiązanie

Wiązanie odnosi się ściśle do transportu jaki jest używany do komunikacji.

Tabela 1.1. Wiązania oparte na protokole HTTP

Element konfiguracyjny	Opis
BasicHttpBinding	Jest konfiguracją używaną przy połączeniach ze standardowymi serwisami Web, co zapewnia swego rodzaju kompatybilność wstecz, a przede wszystkim stymuluje możliwość kooperacji z innymi platformami. Domyślnie kodowanie dla wiadomości ustawione jest na XML.
BasicHttpContextBinding	Rozszerzenie w stosunku do BasicHttpBinding pozwalające dodatkowo na uzyskanie kontekstu wykonania operacji.
WSHttpBinding	Konfiguracja rozszerzona w stosunku do podstawowych wiązań o elementy bezpieczeństwa dla połączeń niesoportych na komunikacji typu duplex.
WSHttpContextBinding	Rozszerzenie WSHttpBinding wykorzystujące nagłówki SOAP do przesyłania kontekstu operacji.
WSDualHttpBinding	Konfiguracja umożliwiająca bezpieczne połączenia dla serwisów typu duplex (do innych serwisów WCF).
WSFederationHttpBinding	Konfiguracja oferująca obsługę WS Federation, które daje możliwość uwierzytelnienia i autentykacji użytkowników.
WebHttpBinding	Konfiguracja dla serwisów udostępniających funkcjonalność przy użyciu żądania HTTP (platin oś XML-JSON) zamiast poprzez wymaganie wiadomości SOAP. Metody w takim serwisie mogą być oznaczone dodatkowo atrybutami WebResource lub WebResourceAttribute (wykorzystywane przez AJAX).

Notatki

Wiązanie odnosi się ściśle do transportu jaki jest używany do komunikacji.

Tabela 1.2. Zestawienie wiązań opartych na protokole TCP

Element konfiguracyjny	Opis
NetTcpBinding	Konfiguracja zapewniająca komunikację pomiędzy serwisami na różnych maszynach za pomocą protokołu TCP.
NetTcpContextBinding	Rozszerzenie w stosunku do NetTcpBinding pozwalające na wykorzystanie nagłówków SOAP do przesyłania kontekstu wywołania.
NetNamedPipeBinding	Konfiguracja zoptymalizowana pod kątem komunikacji w obrębie tej samej maszyny. Najwyższa forma wymiany danych.
NetPeerTcpBinding	Konfiguracja zapewniająca bezpieczne połączenia typu P2P (ang. Peer-to-Peer).

Notatki

Wiązanie odnosi się ściśle do transportu jaki jest używany do komunikacji.

Tabela 1.3. Zestawienie wiązań opartych na MSMQ

Element konfiguracyjny	Opis
<netTcpBinding>	Konfiguracja umożliwia komunikację opartą na kolejkach wiadomości pomiędzy aplikacjami utworzonymi za pomocą WCF z możliwością komunikacji między maszynami.
<msmqIntegrationBinding>	Konfiguracja analogiczna do netTcpBinding, jednakże daje możliwość zintegrowania z istniejącymi już aplikacjami opartymi na technologii COM, napisanymi C++ lub aplikacjach .NET wykorzystujących kolejkę poprzez przestrzeń nazw System.Messaging.

Notatki

Kontrakt

Kontrakt, to pojęcie celowo zaczerpnięcie z języka powszechnego, i określa on detale jakie występują podczas komunikacji pomiędzy klientem a usługą. Kontrakt jest oparty o interfejs:

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    int Sum(int num1, int num2);

    [OperationContract]
    int Subtract(int num1, int num2);

    [OperationContract]
    int Multiply(int num1, int num2);

    [OperationContract]
    int Divide(int num1, int num2);
}
```

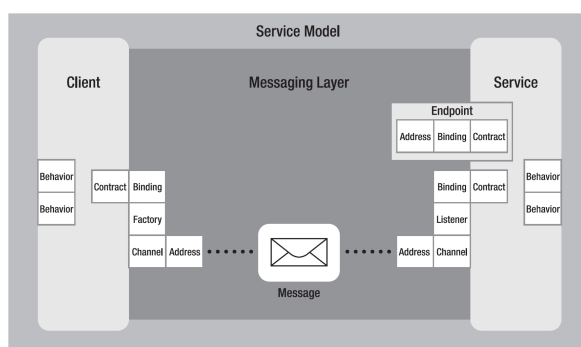
Notatki

Wybrane pięć atrybutów:

- ▶ ServiceContract – oznaczenie iż dany interfejs (dopuszcza się też klasę) zawiera deklaracje metod tworzących usługę,
- ▶OperationContract – metoda jest dostępna w usłudze,
- ▶DataContract – klasa oznaczona tym atrybutem będzie wykorzystywana w wymianie danych,
- ▶DataMember – oznaczenie iż dane pole klasy będzie widoczne dla klienta,
- ▶EnumMember – wartość typu wyliczeniowego.

Notatki

Model programowania WCF



Notatki

Kanał komunikacyjny

Kanał komunikacyjny jest odpowiedzialny za przekazywanie wiadomości pomiędzy usługą a klientem. Do jego najważniejszych zadań przynależy:

- ▶ obsługa protokołów transportu poprzez przyłącza typu: HTTP, WSHTTP, TCP, MSMQ, potoki nazwane,
- ▶ kodowanie oraz szyfrowanie,
- ▶ zarządzanie sesjami (reliable session – niezawodne/wiarygodne),
- ▶ tryby komunikacji: simplex, duplex, send and wait,
- ▶ tryby bezpieczeństwa.

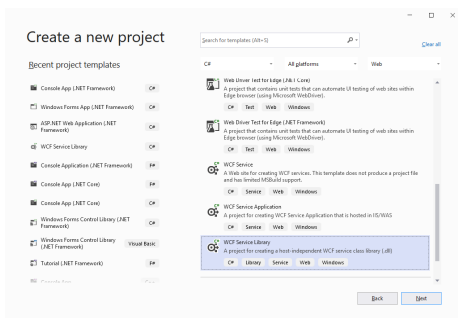
Kanały, i protokoły komunikacji zapewniają także pewien zakres interoperacyjności z innymi platformami:

- ▶ BasicHttpBinding – dla każdego klienta HTTP,
- ▶ WSHttpBinding – platformy that use ws extensions
- ▶ NetTcpBinding – aplikacje .NET,
- ▶ MSMQ – obsługa WCF przez aplikacje MSMQ nie implementujących bezpośrednio elementów WCF.

Notatki

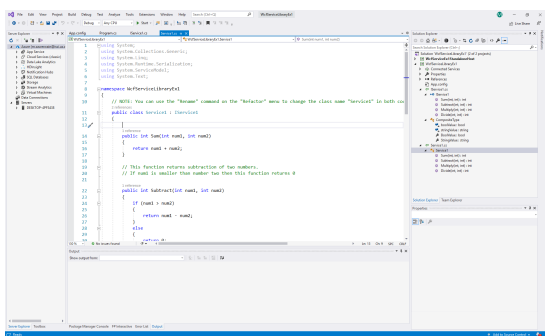
Tworzenie biblioteki dla usługi

Tworzenie projektu tj. biblioteki z usługą WCF:



Notatki

Projekt z serwisem:



Notatki

Interfejs usługi

Interfejs usługi, początek:

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

namespace WcfServiceLibraryEx1 {
    [ServiceContract]
    public interface IService1 {
        [OperationContract]
        int Sum(int num1, int num2);

        [OperationContract]
        int Subtract(int num1, int num2);

        [OperationContract]
        int Multiply(int num1, int num2);

        [OperationContract]
        int Divide(int num1, int num2);
    }
}
```

Notatki

Interfejs usługi, dokończenie:

```
[DataContract]  
public class CompositeType  
{  
    bool boolValue = true;  
    string stringValue = "Hello ";  
  
    [DataMember]  
    public bool BoolValue  
    {  
        get { return boolValue; }  
        set { boolValue = value; }  
    }  
  
    [DataMember]  
    public string StringValue  
    {  
        get { return stringValue; }  
        set { stringValue = value; }  
    }  
}
```

V1.00 – 61/ 81

Notatki

Implementacja interfejsu

Implementacja interfejsu, początek:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Runtime.Serialization;  
using System.ServiceModel;  
using System.Text;  
  
namespace WcfServiceLibraryEx1 {  
    public class Service1 : IService1 {  
  
        public int Sum(int num1, int num2) {  
            return num1 + num2;  
        }  
        public int Subtract(int num1, int num2)  
        {  
            if (num1 > num2) {  
                return num1 - num2;  
            }  
            else {  
                return 0;  
            }  
        }  
    }  
}
```

V1.00 – 62/ 81

Notatki

Implementacja interfejsu, dokończenie:

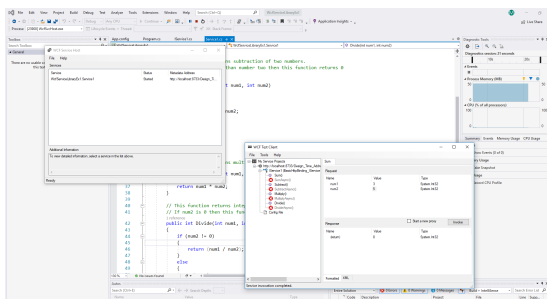
```
        public int Multiply(int num1, int num2)  
        {  
            return num1 * num2;  
        }  
  
        public int Divide(int num1, int num2)  
        {  
            if (num2 != 0)  
            {  
                return (num1 / num2);  
            }  
            else  
            {  
                return 1;  
            }  
        }  
    }  
}
```

V1.00 – 63/ 81

Notatki

Testowanie usługi

Podstawowa wersja usługi może być przetestowana za pomocą narzędzia WCF Test Client:



V1.00 – 64/ 81

Notatki

Opis konfiguracji

Istotnym elementem jest też plik konfiguracji serwisu:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding IService1" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:8733/Design_Time_Addresses/WcfServiceLibraryEx1/Service1/"
        binding="BasicHttpBinding" bindingConfiguration="BasicHttpBinding IService1"
        contract="ServiceReference1.IService1" name="BasicHttpBinding IService1" />
    </client>
    <services>
      <service name="WcfServiceLibraryEx1.Service1">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8733/Design_Time_Addresses/WcfServiceLibraryEx1/Service1/" />
          </baseAddresses>
          <faults>
            <!-- Service Endpoints -->
            <!-- Unless fully qualified, address is relative to base address supplied above -->
            <endpoint address="" binding="BasicHttpBinding" contract="WcfServiceLibraryEx1.IService1" />
            <!-- Upon deployment, the following identity element should be removed or replaced to reflect the
            identity under which the deployed service runs.  If removed, WCF will infer an appropriate identity
            automatically. -->
            <identity>
              <dns value="localhost"/>
            </identity>
          </faults>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
V1.00 – 65 / 81
```

Notatki

Istotnym elementem jest też plik konfiguracji serwisu, zakończenie pliku:

```
<!-- Metadata Endpoints -->
<!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->
<!-- This endpoint does not use a secure binding and should be secured or removed before deployment -->
<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
</service>
</services>
<behaviors>
  <serviceBehaviors>
    <behavior>
      <!-- To avoid disclosing metadata information,
      set the values below to false before deployment -->
      <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>
      <!-- To receive exception details in faults for debugging purposes,
      set the value below to true.  Set to false before deployment
      to avoid disclosing exception information -->
      <serviceDebug includeExceptionDetailInFaults="False" />
    </behavior>
  </serviceBehaviors>
</behaviors>
</system.serviceModel>
</configuration>
V1.00 – 66 / 81
```

Notatki

Udostępnienie usługi

W odróżnieniu od WEB Services, usługi WCF oferują bardzo elastyczne podejście do ich udostępniania:

- ▶ usługa samohostująca się (ang. self-hosting),
- ▶ usługa dostępna poprzez system usług Windows,
- ▶ wykorzystanie serwera IIS.

Elastyczność pozwala dostosowanie projektu usługi, do skali i zapotrzebowania na usługi, można utworzyć niewielką aplikację, albo wykorzystywać infrastrukturę systemu Windows do skalowania, i oferowania dużej wydajności w realizacji poszczególnych usług. Bądź wykorzystanie infrastruktury serwera IIS.

```
using (ServiceHost host =
new ServiceHost(typeof(IService1))) {
  Console.WriteLine("Personal Information Service host starting");
  host.Open();
  Console.WriteLine("Press [ENTER] to stop service...");
  Console.ReadLine();
}
V1.00 – 67 / 81
```

Notatki

Bardziej rozbudowana konfiguracja:

```
Uri baseAddr = new Uri("http://localhost:9000/Service1/");
using (ServiceHost host = new ServiceHost(typeof(IService1), baseAddr))
{
  //Add Endpoint
  host.AddServiceEndpoint(typeof(IService1),
  new BasicHttpBinding(), baseAddr);

  //Enable MEX
  ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
  //smb.HttpGetEnabled = true;
  host.Description.Behaviors.Add(smb);
  host.AddServiceEndpoint(new ServiceMetadataEndpoint(
  new EndpointAddress(baseAddr.AbsoluteUri + "mex"));

  Console.WriteLine("Personal Information Service host
  with inline configuration is starting");
  //run host
  host.Open();
  Console.WriteLine("Press [ENTER] to stop service...");
  Console.ReadLine();
}
V1.00 – 68 / 81
```

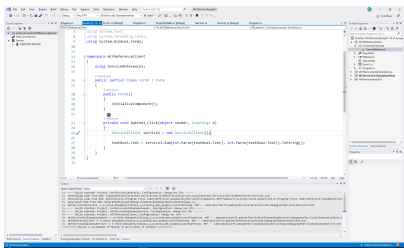
Notatki

Klient z usługi

Dostęp do usługi również można realizować na kilka sposobów np.:

- ▶ klient uzyskuje dostęp poprzez referencję,
- ▶ samodzielnie utworzony kanał transportu,
- ▶ asynchroniczne wywołanie klienta.

Podójście pierwsze poprzez referencję:



V1.00 – 69 / 81

Notatki

Samodzielnie zestawienie kanału transportu dla usługi:

```
Uri baseAddr = new Uri("http://localhost:8088/PersonInformationSvc.svc/");  
  
ChannelFactory<IPersonInformationSvc> factory =  
    new ChannelFactory<IPersonInformationSvc>(new WSHttpBinding(),  
        new EndpointAddress(baseAddr));  
  
IPersonInformationSvc proxy = factory.CreateChannel();  
var response = proxy.GetPersonInformation(  
    new PersonInformationSvc.Messages.PersonInformationSvcRequest()  
    {  
        PId = 1  
    });
```

V1.00 – 70 / 81

Notatki

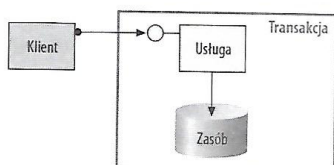
Asynchroniczne wywołanie klienta:

```
using (PersonalInformationServiceClient client =  
    new PersonalInformationSvcClient("BasicHttpBinding_IPersonInformationSvc") {  
        client.GetPersonInformationCompleted +=  
            new EventHandler<GetPersonInformationCompletedEventArgs>(client_GetPersonInformationCompleted);  
    })  
    client.GetPersonInformationAsync(new PersonInformationRequest()  
    {  
        PId = 1  
    });  
  
static void client_GetPersonInformationCompleted(object sender,  
    GetPersonInformationCompletedEventArgs e) {  
    foreach (var person in e.Result.Persons) {  
        Console.WriteLine("{0} : {1} {2}", person.PId,  
            person.FName, person.SName);  
    }  
}
```

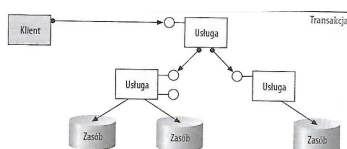
V1.00 – 71 / 81

Notatki

WCF oferuje także sterowanie przepływem transakcji do implementowania bezpiecznych usług. Transakcja dla jednej usługi:



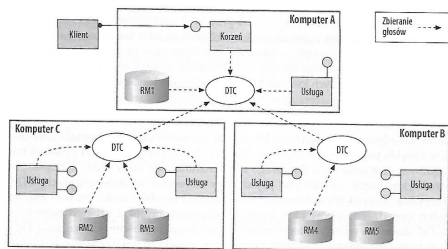
Rozproszona transakcja:



V1.00 – 72 / 81

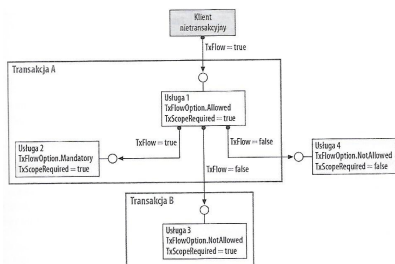
Notatki

Rozproszona transakcja zarządzana przez DTC:



Notatki

Przykładowy przepływ transakcyjny:



Notatki

Treść kontraktu z podstawową obsługą transakcji:

```
[ServiceContract(Namespace = "mega.service.namespace")]
public interface ICalculator
{
    [OperationContract]
    [TransactionFlow(TransactionFlowOption.Mandatory)]
    double Add(double n1, double n2);
    [OperationContract]
    [TransactionFlow(TransactionFlowOption.Allowed)]
    double Subtract(double n1, double n2);
    [OperationContract]
    [TransactionFlow(TransactionFlowOption.NotAllowed)]
    double Multiply(double n1, double n2);
    [OperationContract]
    double Divide(double n1, double n2);
}
```

Notatki

Konfiguracja przyłącza

Fragmety konfiguracji przyłącza do obsługi transakcji:

```
<bindings>
<netTcpBinding>
<binding name="transactionalOleTransactionsTcpBinding"
transactionFlow="true"
transactionProtocol="OleTransactions"/>
</netTcpBinding>
<wsHttpBinding>
<binding name="transactionalWsHttpBinding"
transactionFlow="true" />
</wsHttpBinding>
</bindings>
```

Notatki

Implementacja:

```
[ServiceBehavior(TransactionIsolationLevel = System.Transactions.IsolationLevel.Serializable)]  
public class CalculatorService : ICalculator  
{  
    [OperationBehavior(TransactionScopeRequired = true)]  
    public double Add(double n1, double n2)  
    {  
        RecordToLog(String.Format(CultureInfo.CurrentCulture, "Adding {0} to {1}", n1, n2));  
        return n1 + n2;  
    }  
  
    [OperationBehavior(TransactionScopeRequired = true)]  
    public double Subtract(double n1, double n2)  
    {  
        RecordToLog(String.Format(CultureInfo.CurrentCulture, "Subtracting {0} from {1}", n2, n1));  
        return n1 - n2;  
    }  
  
    [OperationBehavior(TransactionScopeRequired = true)]  
    public double Multiply(double n1, double n2)  
    {  
        RecordToLog(String.Format(CultureInfo.CurrentCulture, "Multiplying {0} by {1}", n1, n2));  
        return n1 * n2;  
    }  
  
    [OperationBehavior(TransactionScopeRequired = true)]  
    public double Divide(double n1, double n2)  
    {  
        RecordToLog(String.Format(CultureInfo.CurrentCulture, "Dividing {0} by {1}", n1, n2));  
        return n1 / n2;  
    }  
}
```

V1.00 – 77 / 81

Notatki

Używanie usługi Calculator oraz podstawowa obsługa transakcji:

```
using (TransactionScope tx =  
    new TransactionScope(TransactionScopeOption.RequiresNew))  
{  
    Console.WriteLine("Starting transaction");  
  
    // Call the Add service operation  
    // - generatedClient will flow the required active transaction  
    double value1 = 100.00D;  
    double value2 = 15.99D;  
    double result = client.Add(value1, value2);  
    Console.WriteLine(" Add({0},{1}) = {2}", value1, value2, result);  
  
    // Call the Subtract service operation  
    // - generatedClient will flow the allowed active transaction  
    value1 = 145.00D;  
    value2 = 76.54D;  
    result = client.Subtract(value1, value2);  
    Console.WriteLine(" Subtract({0},{1}) = {2}", value1, value2, result);  
}
```

V1.00 – 78 / 81

Notatki

Używanie usługi Calculator oraz podstawowa obsługa transakcji:

```
// Start a transaction scope that suppresses the current transaction  
using (TransactionScope txSuppress =  
    new TransactionScope(TransactionScopeOption.Suppress))  
{  
    // Call the Subtract service operation  
    // - the active transaction is suppressed from the generatedClient  
    // and no transaction will flow  
    value1 = 21.05D;  
    value2 = 42.16D;  
    result = client.Subtract(value1, value2);  
    Console.WriteLine(" Subtract({0},{1}) = {2}", value1, value2, result);  
  
    // Complete the suppressed scope  
    txSuppress.Complete();  
}  
  
// Call the Multiply service operation  
// - generatedClient will not flow the active transaction  
value1 = 9.00D;  
value2 = 81.25D;  
result = client.Multiply(value1, value2);  
Console.WriteLine(" Multiply({0},{1}) = {2}", value1, value2, result);
```

V1.00 – 79 / 81

Notatki

Używanie usługi Calculator oraz podstawowa obsługa transakcji:

```
// Call the Divide service operation.  
// - generatedClient will not flow the active transaction  
value1 = 22.00D;  
value2 = 7.00D;  
result = client.Divide(value1, value2);  
Console.WriteLine(" Divide({0},{1}) = {2}", value1, value2, result);  
  
// Complete the transaction scope  
Console.WriteLine(" Completing transaction");  
tx.Complete();  
}  
  
Console.WriteLine("Transaction committed");
```

V1.00 – 80 / 81

Notatki

W następnym tygodniu między innymi

1. techniki programowania równoległego w C#.
2. wątki, zadania raz jeszcze,
3. pakiet CUDAfy,
4. siatka obliczeniowa,
5. kernele obliczeniowe, przykłady.

Proponowane tematy prac pisemnych:

1. testy wydajności usług SOAP,
2. odkrywca i tester usług SOAP w IronPythonie,
3. porównanie narzędzi do serwisów SOAP w Javie i C#.
4. Nowe rozwiązanie gRPC w miejsce WCF,
5. Omówienie obsługi transakcji WCF,
6. Aspekty bezpieczeństwa aplikacji WCF.

Dziękuję za uwagę!!!

V1.00 – 81 / 81

Notatki

Notatki

Notatki

Notatki
