Platforma .NET – Wykład 14
WCF – komunikacja sieciowa

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

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: Thursday 8th April, 2021, t: 13:08

Notatki

Spis treści

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 cześć pierwsza
(7) Dostęp do baz danych

Notatki

Plan wykładu – tydzień po tygodniu

(8) Język zapytań LINQ
(8b) 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
(14) WCF – komunikacja sieciowa
(15) Wybrane elementy programowania równoległego w C#
(*) "Klasówka II", czyli egzamin cześć druga

Notatki

Plan wykładu

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

Notatki

Przetwarzanie rozproszone

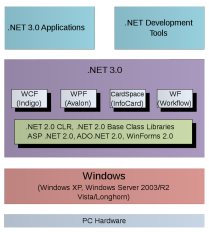
Najważniejsze etapy rozwoju przetwarzania rozproszonego:

- 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

Notatki

Stos .NET oraz WCF

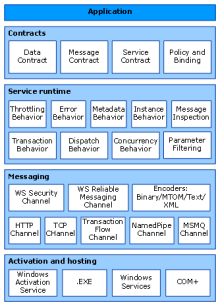
Miejsce WCF w stosie systemu .NET



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

Notatki

Główne pojęcia WCF



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

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.

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 dostępy do usługi.

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 – adress, 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/SampleWCFDotNetLecture,
- net.tcp://localhost:8733/SampleWCFDotNetLecture,
- net.msmq://localhost/private$/SampleWCFDotNetLectureMsmq,
- net.pipe://localhost/SampleWCFDotNetLecture.

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	<basicHttpBinding>	Jest konfiguracją używaną przy połączeniach ze standardowymi serwisami Web, co zapewnia swego rodzaju kompatybilność wstecz, a przede wszystkim utrzymuje możliwość kooperacji z innymi platformami. Domyślnie kodowanie dla wiadomości ustawione jest na XML.
BasicHttpContextBinding	<basicHttpContextBinding>	Rozszerzenie w stosunku do BasicHttpBinding pozwalające dodatkowo na uzyskanie kontekstu wykonania operacji.
WSHttpBinding	<wsHttpBinding>	Konfiguracja rozszerzona w stosunku do podstawowych wiązań o elementy bezpieczeństwa dla połączeń nieopartych na komunikacji typu duplex.
WSHttpContextBinding	<wsHttpContextBinding>	Rozszerzenie WSHttpBinding wykorzystujące nagłówki SOAP do przekazywania kontekstu operacji.
WSDualHttpBinding	<wsDualHttpBinding>	Konfiguracja umożliwiająca bezpieczne połączenia dla serwisów typu duplex (do innych serwisów WCF).
WSFederationHttpBinding	<wsFederationHttpBinding>	Konfiguracja oferująca obsługę WSFederation, które daje możliwość uwierzytelniania i autoryzacji użytkowników.
WebHttpBinding	<webHttpBinding>	Konfiguracja dla serwisów udostępniających funkcjonalność przy użyciu żądania HTTP (plain old XML-POX) zamiast poprzez wymianę wiadomości SOAP. Metody w takim serwisie muszą być oznaczone dodatkowo atrybutami WebGet lub WebInvoke (wykorzystywanymi 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	<netTcpBinding>	Konfiguracja zapewniająca komunikację pomiędzy serwisami na różnych maszynach za pomocą protokołu TCP.
NetTcpContextBinding	<netTcpContextBinding>	Rozszerzenie w stosunku do NetTcpBinding pozwalające na wykorzystanie nagłówków SOAP do przekazywania kontekstu wykonania.
NetNamedPipeBinding	<netNamedPipeBinding>	Konfiguracja zoptymalizowana pod kątem komunikacji w obrębie tej samej maszyny. Najszybsza forma wymiany danych.
NetPeerTcpBinding	<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
NetMsmqBinding	<netMsmqBinding>	Konfiguracja umożliwia komunikację opartą na kolejkach wiadomości pomiędzy aplikacjami utworzonymi za pomocą WCF z możliwością komunikacji między maszynami.
MsmqIntegrationBinding	<msmqIntegrationBinding>	Konfiguracja analogiczna do NetMsmqBinding, jednakże daje możliwość zintegrowania z istniejącymi już aplikacjami opartymi na technologii COM, natywnym C++ lub aplikacjach .NET wykorzystujących kolejki 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);
}
```

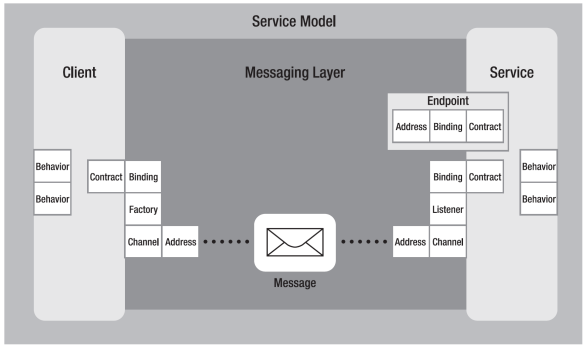

Notatki

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 przekazywania 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 (realiable session – niezawodne/wiarygodne),
- tryby komunikacji: simplex, duplex, send and wait,
- tryby bezpieczeństwa.

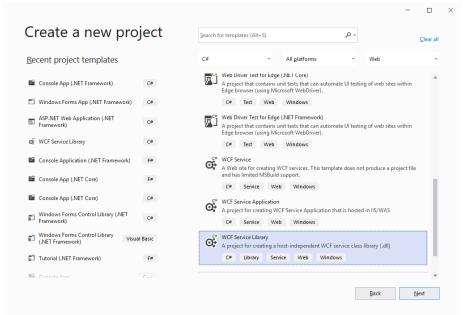
Kanały, i protokoły komunikacji zapewniają także pewien zakres interoperacyjności z innymi platformami:

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

Notatki

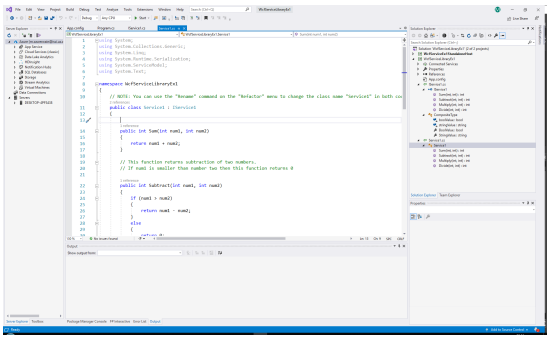
Tworzenie biblioteki dla usługi

Utworzenie 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; }
        }
    }
}
```


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;
            }
        }
```


Notatki

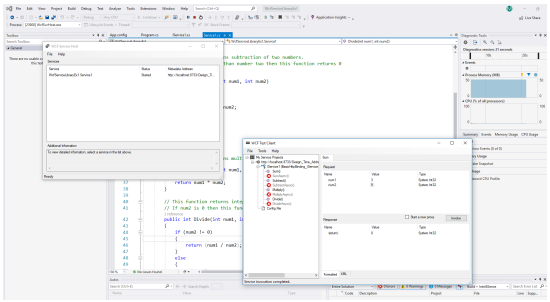
Implementacja interfejsu, dokończenie:

```
        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;
            }
        }
    }
}
```


Notatki

Testowanie usługi

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



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>
        </host>
        <!-- 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>
        </endpoint>
```


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>
```


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(Service1))) {
   Console.WriteLine("Personal Information Service host starting");
   host.Open();
   Console.WriteLine("Press [ENTER] to stop service...");
   Console.ReadLine();
}
```


Notatki

Bardziej rozbudowana konfiguracja:

```
Uri baseAddr = new Uri("http://localhost:9000/Service1/");
using (ServiceHost host = new ServiceHost(typeof(Service1), 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();
}
```

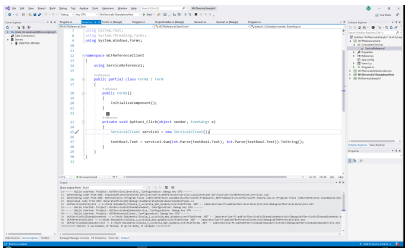

Notatki

Klient z usługi

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

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

Podejście pierwsze poprzez referencję:



Notatki

Samodzielnie zestawienie kanału transportu dla usługi:

```
Uri baseAddr = new Uri(
            "http://localhost:8088/PersonInformationSrvc.svc/");

ChannelFactory<IPersonInformationSrvc> factory =
    new ChannelFactory<IPersonInformationSrvc>(new WSHttpBinding(),
                                        new EndpointAddress(baseAddr));

IPersonInformationSrvc proxy = factory.CreateChannel();
var response = proxy.GetPersonInformation(
    new PersonInformationSrvc.Messages.PersonInformationSrvcRequest()
    {
        PId = 1
    });
```

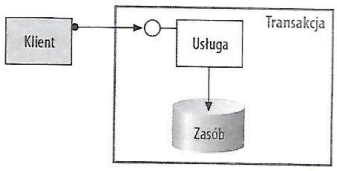

Notatki

Asynchroniczne wywołanie klienta:

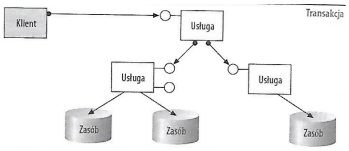
```
using (PersonalInformationServiceClient client =
       new PersonInformationSrvcClient(
         "BasicHttpBinding_IPersonInformationSrvc")) {
    client.GetPersonInformationCompleted +=
        new EventHandler<GetPersonInformationCompletedEventArgs>(
                    client_GetPersonInformationCompleted);
    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);
    }

}
```


Notatki

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

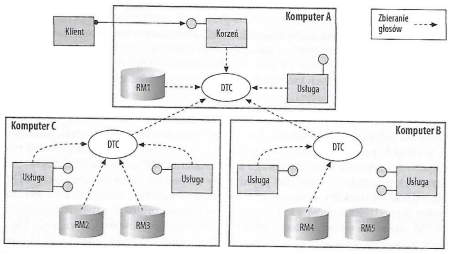


Rozproszona transakcja:



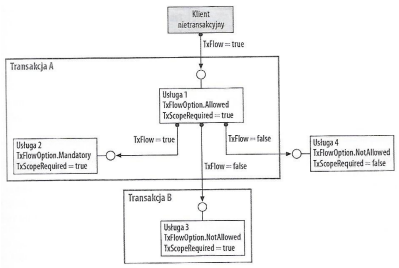
Notatki

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

Fragmenty konfiguracji przyłącza do obsługi transakcji:

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


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);
```


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);
```


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");
```


Notatki

W następnym tygodniu między innymi

Wybrane pojęcia i zagadnienia omawiane na następnym wykładzie:

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. Nowe rozwiązanie gRPC w miejsce WCF.
2. Omówienie obsługi transakcji WCF.
3. Aspekty bezpieczeństwa aplikacji WCF.

Dziękuje za uwagę!!!

Notatki

Notatki