SPRAWOZDANIE

Programowanie równoległe

Message brokers - użycie Work Queue i RPC

16.12.2015

Karol Suwalski 125NCI B

https://github.com/SuwalskiKarol/oor.git

MSMQ + WCF

Jak to mówią:

Nie lubie mainstreamu, zamiast opisywać w ostatnich latach coraz bardziej popularnego RabbitMQ postanowiłem skupić się na sprawdzonym MSMQ.

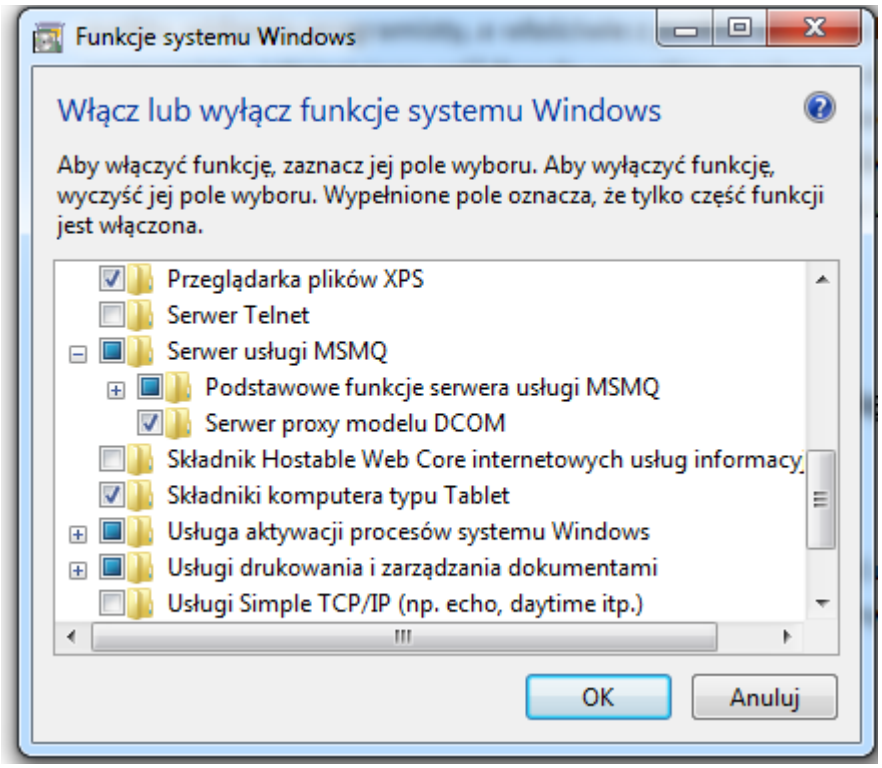
MSMQ jest implantacją kolejek na platformie .NET. Nie jest to bardzo dobrze udokumentowana technologia dlatego pomyślałem ,że spróbuję coś w tym zdziałać.

Kolejka komunikatów jest asynchronicznym protokołem komunikacyjnym, co oznacza, że odbiorca i nadawca wiadomości nie muszą łączyć się z kolejką w tym samym czasie. Komunikaty przesłane kolejce są przechowywane aż do czasu odebrania przez inny proces.

A czym jest WCF? Został on stworzony w celu ujednolicenia różnych sposobu komunikacji z punktu widzenia programisty, a właściwie z punktu widzenia logiki tworzonej przez programistę. Jaki jest tego cel? Przede wszystkim można go określić, jako elastyczność tworzonej architektury, gdyż dzięki WCF w łatwy i szybki sposób możemy przejść na przykład z komunikacji HTTP do komunikacji przy pomocy kolejek MSMQ.Pozwala na zmianę ograniczeń dotyczących przesyłanych wiadomości. Ma wiele więcej funkcji, których po prostu nie znam ☺.

Prosta aplikacja przedstawiająca kolejkę komunikatów.

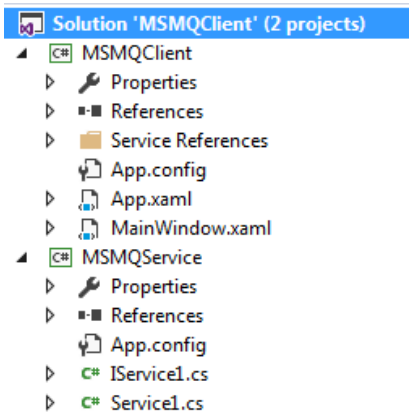
1. Przed rozpoczęciem pisania aplikacji musimy włączyć usługę MSMQ. Jest to usługa wbudowana w system operacyjny Windows, ale domyślnie wyłączona. Wystarczy ją włączyć, niczego nie trzeba pobierać.



2. W celu prezentacji działania MSMQ potrzebujemy dwa projekty;

   - Projekt WCF Service Library, który będzie hostował naszą usługę MSMQ na serwerze.
   - Projekt,który stworzy nam klienta aplikacji, który może być uruchomiony na innym bądź tym samym komputerze. Postawiłem na WPF w celu podszkolenia tej technologii. Windows formy już mnie nudzą ☺, poza tym i tak coraz rzadziej się je stosuje na rzecz chociażby właśnie WPF.

Żeby nie tworzyć dwóch oddzielnych programów postawiłem oby dwa projekty w jednej solucji.



3. Do interfejsu usługi(ISrevice1.cs) musimy dodać dwie metody odpowiedzialne za przyjmowanie wiadomości tekstowych oraz datę wysłania tych wiadomości.

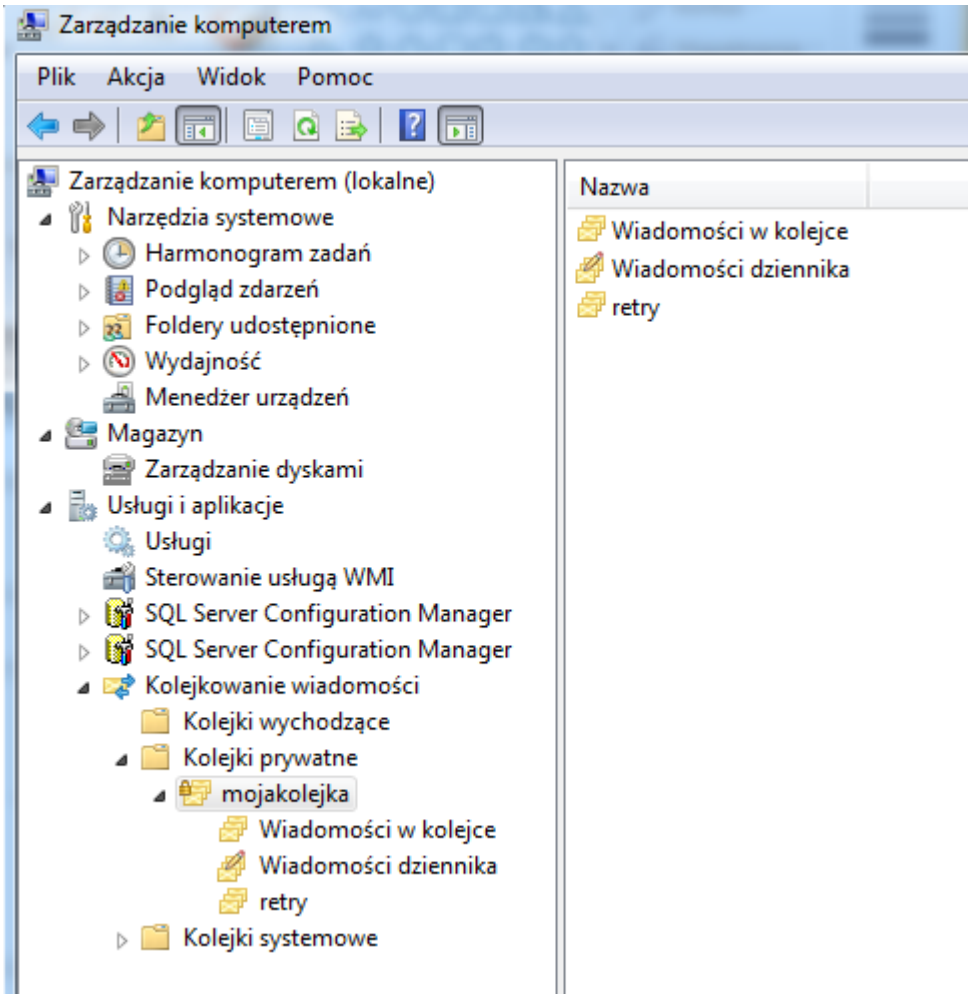
```
[ServiceContract]
publicinterfaceIService1
    {
        [OperationContract(IsOneWay = true)]
void SendLockMessage(string msg, DateTime sendDate);

        [OperationContract(IsOneWay = true)]
void SendToOutputWindow(string msg, DateTime sendDate);
       }
```

Atrybut`[OperationContract(IsOneWay=true)]` określa ,że wywołanie metody nigdy nie zwróci klientowi żadnego komunikatu.Metody, więc niczego nie zwracają.

4. W Service1 Tworzymy statyczny obiekt tylko do odczytu, który będzie służył nam za informatora określającego czy inny wątek go właśnie używa. Dopóki pierwszy wątek nie zwróci tego obiektu pozostałe wątki muszą na niego czekać. Wątek używający ten obiekt zwróci go dopiero po wykonaniu bloku kodu zwartego w słowie kluczowym “lock”. Jest to nam potrzebne w przypadku, gdy aplikacja serwerowa zostanie wyłączona a komunikaty wciąż będą wysyłane. Gdy serwer zostałby włączony wszystkie komunikaty zostałyby wykonane prawie równolegle. Bez kolejki. Wyskoczyłby nam błąd, że proces nie może uzyskać dostępu do pliku, bo jest używany przez inny proces.

```
publicvoid SendLockMessage(string msg, DateTime sendDate)
        {
lock (SyncObject)
            {
                W.WriteLine("Otrzymana: {0} {1}", DateTime.Now.ToLongTimeString(),
DateTime.Now.ToLongDateString());
                W.WriteLine("Wysłana {0} {1}", sendDate.ToLongTimeString(),
                    sendDate.ToLongDateString());
                W.WriteLine(" :");
                W.WriteLine(" :{0}", msg);
W.WriteLine("------------------------------");
                W.Flush();
            }
       }
```

5. Przyszedł czas na stworzenie kolejki. Tworzymy ją w oknie zarządzania komputerem.



6. Cała magia związana z konfiguracją usługi MSMQ odbywa się w pliku app.config. Nie pokaże całego pliku, bo połowy tej magii sam nie rozumiem, ale przedstawię, co ważniejsze konfiguracje

   Zmieniamy nasz bazowy adres. Nie ma tu słowa “localhost”. Mamy natomiast nazwę komputera.

```
<baseAddresses>
<addbaseAddress="http://Czarli-Komputer:8080/MSMQService/"/>
</baseAddresses>
```

Ustawiamy endpointy

```
<endpointaddress="net.msmq://Czarli-Komputer/private/mojakolejka"
binding="netMsmqBinding"bindingConfiguration="MyBinding"
contract="MSMQService.IService1">
<identity>
<dnsvalue="localhost" />
</identity>
</endpoint>
```

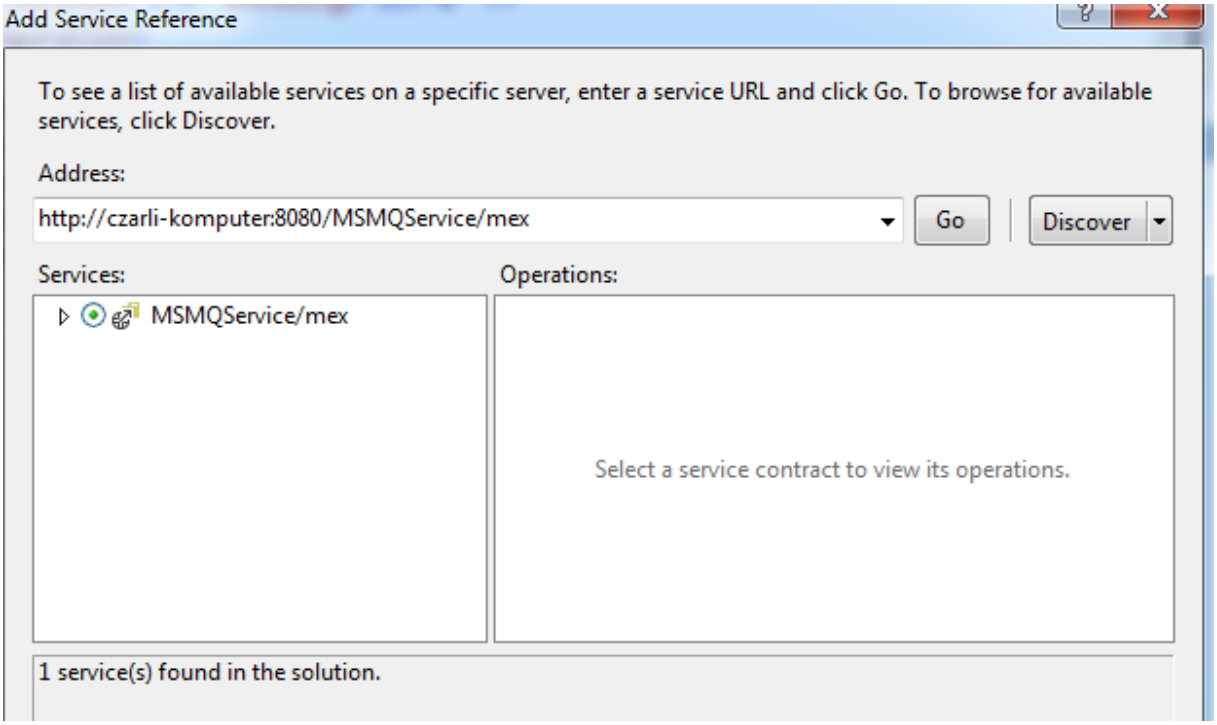
Endpointy określają swój adres, w którym mogą być odnalezione jak i określają protokół, w jakim klient ma się z nim komunikowa, Adres endpointu wskazuje na wcześniej utworzoną kolejkę prywatną jak i na komputer na którym ta kolejka się znajduję… chyba☺

Sprawdzamy czy nazwa usługi jest zgodna z pełną ścieżka do obiektu klasy.

```
<servicename="MSMQService.Service1">
```

7. Jeżeli wszystko jest zrobione dobrze to możemy sprawdzić działanie testowego klienta WCF

8. Stwórzmy klienta. Do projektu klienta musimy dodać referencję usługi MSMQ.



   Dzięki temu do projektu powinny zostać dodane odpowiednie referencję do dll-ek jak i folder “Service References”.

9. Tworzymy w XAMLu interfejs klienta. Buttona oraz Textbox.

10. W kodzie pobocznym tworzymy instancje klienta i odbierają tekstową informację od użytkownika zapisaną w textbox-ie,po czym wysyłamy tą informację przy pomocy dwóch metod SendLockMessage i SendToOutputWindows.

```
private   MSMQServiceReference.Service1Client client;

public MainWindow()
        {
            InitializeComponent();
            client = newService1Client();
        }

privatevoid buttonSendMsg_Click(object sender, RoutedEventArgs e)
        {
            client.SendLockMessage(textBox1.Text, DateTime.Now);
            client.SendToOutputWindow(textBox1.Text, DateTime.Now);

        }
```

   Wszystko gotowe. Teraz możemy wysyłać wiadomości do kolejki. Jak to wszystko działa opowiem na laboratorium ☺.