

X Konferencja PLOUG
Kościelisko
Październik 2004

Usługi WebServices w Oracle

Bartłomiej Jabłoński

*Uniwersytet Łódzki
e-mail: bartek@math.uni.lodz.pl*

Abstrakt

WebServices – usługi sieciowe – to zbiór szybko ewoluujących standardów. Z biznesowego punktu widzenia te technologie oferują przede wszystkim nowe możliwości w zakresie integracji aplikacji pomiędzy różnymi partnerami biznesowymi. Od samego początku, poprzez organizacje W3C i OASIS, firma Oracle uczestniczy w pracach nad rozwojem standardów. W swoich produktach dostarcza narzędzi do konstruowania aplikacji aktywnie wyszukujących usługi w sieci oraz aplikacje te usługi realizujące.

Artykuł przedstawia główne koncepcje związane z usługami sieciowymi, problemy i korzyści związane z rozproszeniem aplikacji. Przegląd narzędzi Oracle 10g związanych z usługami sieciowymi wzbogacony jest przykładami ilustrującymi podstawowe techniki implementowania, udostępniania i zabezpieczania usług.

1. Czym są usługi sieciowe?

1.1. Definicja

Nie ma jednej definicji usług sieciowych, w literaturze spotkać można ich wiele i większość z nich ma jedną, zasadniczą wadę: dążąc do maksymalnej perfekcji stały się nieczytelne i trudne do wyobrażenia dla zarówno dla programisty, jak i osoby, w której gestii leży decyzja o zastosowaniu tej technologii w praktyce.

Na początek warto sobie wyjaśnić, że „usługi sieciowe” nie muszą mieć nic wspólnego z siecią. W szczególnym przypadku usługi te mogą być wywoływane wewnątrz tego samego procesu (lub różnych procesów z wykorzystaniem mechanizmu pamięci współdzielonej) w systemie operacyjnym. Jednak, jako że najczęściej podawane przykłady oparte są o HTTP, usługi te kojarzone są z siecią Internet, a w szczególności z WWW.

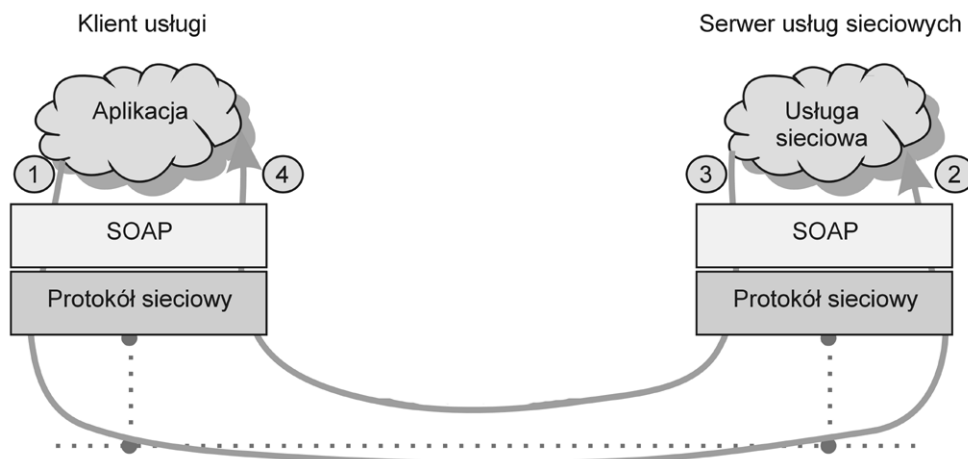
Posługując się definicją IBM [Kreg01] można powiedzieć, że:

„Usługa sieciowa jest interfejsem opisującym kolekcję operacji, do których, na podstawie standaryzowanych komunikatów w XML-u, istnieje dostęp przez sieć. Usługi sieciowe wykonują określone zadanie lub zestaw zadań. Usługa sieciowa opisana jest standardowym, formalnym dokumentem XML, zwanym opisem usługi, która zawiera wszystkie potrzebne do interakcji z usługą, włącznie z opisem formatu komunikatów (dzięki którym są wykonywane operacje), protokołów transportowych i lokalizacji.

Interfejs z założenia ukrywa szczegóły implementacji usługi, dzięki czemu można z niej korzystać w ten sam sposób niezależnie od platformy sprzętowej i systemowej, na której usługa działa oraz od języka programowania, w jakim usługa została napisana. Takie podejście pozwala i zachęca do tego, aby aplikacje oparte na usługach sieciowych były luźno powiązane, zbudowane z komponentów i niezależne we współpracy z innymi usługami w celu przeprowadzenia złożonej operacji lub transakcji biznesowej.”

Rysunek 1 podaje przykładową konfigurację. Aplikacja realizuje proces biznesowy, którego jeden z etapów obsługiwany jest przez inny system. Np. realizacja zamówienia klienta związana jest z wystawieniem faktury i jej odpowiednim zaksięgowaniem w systemie FK. Aby proces ten przebiegał bezawaryjnie należy usunąć z niego najbardziej zawodny element. Niestety okazuje się, że jest nim najczęściej człowiek. Automatyzacja procesu biznesowego (czytaj: integracja systemów) to zadanie niezwykle trudne, gdy każdy z tych systemów pracuje w innym środowisku, na innym serwerze, napisany był przy pomocy różnych narzędzi, przez różne firmy i na dodatek nie ma możliwości wglądu w kod źródłowy.

Usługi sieciowe webservices to zupełnie nowe podejście do tworzenia aplikacji. Aplikacja, do której istnieje standardowe „dojście” jest na pewno bardziej atrakcyjnym produktem, niż system hermetyczny, którego najmniejsza rozbudowa związana jest z zaangażowaniem producenta. W podejściu SOA (Service-Oriented Architecture) zakłada się, że aplikacja podzielona jest na szereg komponentów – każdy z nich realizuje jakąś funkcję biznesową – które porozumiewają się ze sobą realizując określony proces biznesowy i przekazując użytkownikowi (lub innemu komponentowi) określony rezultat. Każdy z komponentów może być potraktowany jako czarna skrzynka, której lokalizacja określona może być w fazie projektowania lub dopiero podczas eksploatacji systemu. Istotną cechą zastosowanego tu rozproszenia jest to, że lokalizacja usług może być realizowana przez niezależne podmioty biznesowe.



Rys. 1. Komunikacja w usługach sieciowych poprzez SOAP

Umieszczony na rysunku protokół SOAP jest uniwersalnym standardem, niezależnym od platformy czy języka, protokołem, przy pomocy którego różne aplikacje mogą wymieniać między sobą komunikaty lub wywoływać zdalnie procedury.

1.2. Dlaczego tak a nie inaczej?

Czy można inaczej? Oczywiście, że tak! Usługi sieciowe nie są rozwiązaniem idealnym dla każdego problemu. W dzisiejszych czasach przy tworzeniu systemów informatycznych decydencki muszą kierować się oceną kilku ważnych aspektów:

- ceną tworzenia i utrzymywania oprogramowania (często związane są z tym koszty samej infrastruktury: komputery, sieć, szkolenia, centra zapasowe),
- niezawodnością realizowanych usług,
- wydajnością.

Istotnym czynnikiem wpływającym na koszty systemu jest wykorzystywanie rozwiązań już istniejących i sprawdzonych. Jeśli wszystkie komponenty będą umiały się ze sobą dogadać, to pozostanie tylko połączyć je w całość. Wspomniane założenie proste w sformułowaniu niestety przez długi czas nie sprawdzało się w praktyce. Każdy z większych producentów oprogramowania lansował swoje koncepcje doprowadzając do sytuacji, w której standardem stał się nadmiar nie pasujących do siebie standardów.

Usługi sieciowe nie stanowią rewolucji. Są niewielkim, ale nieuniknionym krokiem na drodze rozwoju Internetu. Usługi sieciowe zawierają potencjał dający się wykorzystać w najbliższej przyszłości do znacznego uproszczenia integracji aplikacji, zdefiniowania jasnego interfejsu wymiany danych pomiędzy zaufanymi partnerami biznesowymi, [...] a w nieco dalszej przyszłości na tworzenie serwisów komunikujących się między sobą automatycznie. [Spic02]

Protokół SOAP w chwili obecnej stanowi ustabilizowany już standard komunikacji między aplikacjami. Gartners Group przewiduje, że liczba aplikacji stosujących się do niego wzrośnie gwałtownie w roku 2005. Należy spodziewać się zatem, że przyszłość należy do systemów heterogenicznych, niewykluczone, że rozproszonych w całym Internecie.

1.3. Usługi sieciowe a Oracle

Przy tworzeniu standardów usług sieciowych najczęściej do powiedzenia mają, oczywiście, firmy zajmujące się rozwojem języków programowania (np. Sun, Microsoft, Borland) oraz serwerów internetowych (np. Apache, Macromedia, IBM). Firma Oracle Corp. bardzo wcześnie zaczęła uczestniczyć w pracach nad tworzeniem specyfikacji. Składają się chyba na to dwa powody:

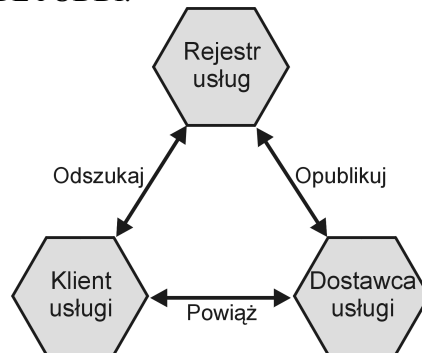
1. podstawą działania większości aplikacji jest baza danych – łatwo sobie wyobrazić komponent-usługę, która będzie pobierała i udostępniała z bazy określone dane

2. jednym z filarów, na którym opiera się potęga Oracle to Application Server, którego podstawą działania jest HTTP oraz J2EE – protokoły, bez których usługi sieciowe nieomal nie mogą się obejść.

Niezwykle trafnie rolę usług sieciowych w środowisku Oracle określił Larry Ellison na jednej z konferencji w 2002 roku: „*Usługi sieciowe są być może modne, ale nie są panaceum na wszystkie dolegliwości. [...] to bardzo ważna nowa technologia a my ją popieramy. Ideą Oracle jest to, aby uczynić z usług sieciowych interfejs dla naszych aplikacji, i żeby Siebel¹ także to zrobił ze swoimi aplikacjami, a będzie łatwiej połączyć zarówno Oracle do SAP-a jak i Siebel-a do SAP-a. To najdziwniejsza rzecz jaką słyszałem w całym życiu*” [CoWe02].

2. Architektura usług webservices i stosowane protokoły

Wszystkich uczestników biorących udział w przekazywaniu informacji za pomocą usług sieciowych można podzielić na trzy grupy – mówimy o tak zwanych rolach: Dostawca usługi, Klient usługi i Rejestr usług. Do porozumiewania się między sobą używają oni co najmniej trzech protokołów komunikacyjnych: SOAP, WSDL i UDDI.



Rys. 2. Architektura usług sieciowych WebServices.

Głównymi cechami usług sieciowych są rozproszenie i automatyzm. Pierwsza z tych cech może stanowić naturalne przedłużenie coraz modniejszego relegowania niektórych usług biznesowych specjalizowanym firmom (*ang. off-source*). Druga – polega na tym, że etap publikowania i wyszukiwania nie musi być realizowany w momencie projektowania/implementacji systemu – może on zachodzić w trakcie wykonywania danego procesu biznesowego, z mniejszym lub większym udziałem użytkownika.

2.1. Modelowy schemat

Typowym przykładem jest następująca sytuacja: dostawca usługi, np. firma specjalizująca się w określonym biznesie, chce świadczyć usługi na zewnątrz.

1. Tworzenie usługi

Tworzy usługę – specjalny program – i udostępnia go w sieci.

2. Publikacja

Jedyny problem firmy polega na tym, że nikt o tej usłudze nie wie. Można oczywiście rozesłać stosowny komunikat wszystkim swoim dotychczasowym klientom, można skorzystać z usług innych firm (przy okazji powiększyć gigabajty spamu), a można też wpisać parametry usługi na żółte kartki książki telefonicznej – najlepiej żeby ta ostatnia była w formie elektronicznej bazy danych i dawała się łatwo przeszukiwać. Pozostaje tylko czekać, aż ktoś zainteresowany skorzysta z takiego rejestru usług, znajdzie stosowny anons i będzie umiał go obsłużyć.

3. Szukanie

¹ Producent systemów CRM

Przyjrzyjmy się teraz tej sytuacji z pozycji potencjalnego klienta. Chce on znaleźć partnera w interesach, który zechce zrealizować naszą potrzebę. Nic prostszego, niż skorzystać z „książki telefonicznej”. Po wyszukaniu listy firm realizujących szukaną usługę, wybiera on najlepszą ofertę i czyta specyfikację techniczną usługi (adres serwera, port nasłuchujący, lista parametrów, itp.).

4. Realizacja

Mając wszystkie parametry techniczne potrzebne do zestawienia połączenia, wiedząc, jakie parametry i w jaki sposób mają być przekazane, może on już realizować usługę.

2.2. Tak proste, że aż podejrzane

Integratorzy aplikacji, mogą w tym momencie uśmiechnąć się z politowaniem. Z doświadczenia wynika, że pierwszych kilka tygodni (jeśli nie miesięcy) zajmuje udokumentowanie interfejsów wymiany danych, następnie króciutka implementacja, po czym następuje długa faza testowania i nanoszenia poprawek. W przypadku integracji systemów różnych firm dochodzi do tego jeszcze aspekt związany z zachowaniem stosownych wymogów bezpieczeństwa. Czyżby jakaś nowa, rewelacyjna metoda integrowania?

Nie należy oczekiwać tu żadnych cudów. Metoda jest dobra, ale należy o niej pamiętać już podczas projektowania systemów. Tam, gdzie do tej pory w programach wyodrębniane były procedury/obiekty/biblioteki teraz mamy do czynienia z usługami. Korzyść jaką odnosi się z podejścia usługowego w tworzeniu systemu polega na tym, że każda z usług może być wykonana przez niezależny zespół informatyków lub niezależną wyspecjalizowaną firmę, oraz że usługa ta może być zaimplementowana w dowolnym środowisku, dowolnej lokalizacji.

„Usługowość” w ramach jednego systemu nie daje nic, poza kłopotami z wdrażaniem nowej technologii. Zalety SOA ujawniają się dopiero wtedy, gdy zmuszeni jesteśmy łączyć ze sobą dwa (lub więcej) systemy.

Mogłoby się wydawać, że oddawanie kontroli nad częścią systemu obcej firmie jest niebezpieczne. Lecz jak traktować kupowanie specjalizowanych bibliotek, które choć dolinkowywane do własnego kodu są jednak nieznaną czarną dziurą (podobnie jak czarną dziurą są kompilatory, systemy operacyjne, BIOSy komputerów, itp.). Praktyka polegająca na wykorzystywaniu gotowych rozwiązań sprawdziła się, gdyż prowadzi do znaczącego obniżenia kosztów produkcji i utrzymania oprogramowania.

Patrząc na historię rozwoju przetwarzania rozproszonego, widać że tendencja ta umacnia się coraz bardziej i będzie stanowiła podstawę przy tworzeniu dużych systemów. Coraz częściej w literaturze pojawia się pojęcie wirtualnego przedsięwzięcia (*ang. virtual enterprise*) [HeMa03, ADG02]

Tabela 1. Historia przetwarzania rozproszonego [GSB+03]

1987	Firma <i>Sun Microsystems</i> tworzy ONC (Open Network Computing) bazujący na RPC jako podstawowy protokół dla sieciowego systemu plikowego. Firma <i>Apollo Computer</i> tworzy NCS (Network Computing System) również bazujący na RPC na potrzeby systemu operacyjnego Domain.
1989	Organizacja <i>Open Software Foundation</i> (obecnie <i>The Open Group</i>) publikuje RFC dla systemu RPC. Powstaje <i>Object Management Group</i> (OMG) – grupa zajmująca się opracowywaniem specyfikacji definiujących przetwarzanie rozproszone niezależne od platformy i języka. Specyfikacja nosi nazwę <i>Common Object Request Broker Architecture</i> (CORBA).
1990	Na rynek wchodzi firma <i>Microsoft</i> , która swoje prace opiera na zmodyfikowanej wersji DCE/RPC.
1991	Ukazuje się specyfikacja DCE 1.0 Ukazuje się CORBA 1.0

1996	Firma Microsoft przedstawia architekturę DCOM (bazująca na starszym OLE, OLE2, i ActiveX). Ukazuje się standard CORBA 2.0. Częścią jej jest protokół IIOP.
1997	Firma Sun wypuszcza pakiet JDK 1.1 zawierający mechanizm Remote Method Invocation (RMI) Firma Microsoft ogłasza powstanie protokołu COM+. Jest to kolejna wersja protokołu COM zbliżająca go do protokołu CORBA.
1999	Firma Sun publikuje specyfikację J2EE (integracja RMI z IIOP). Ukazuje się specyfikacja SOAP - <i>Simple Object Access Protocol</i> .

Używane w usługach standardy

Przetwarzanie rozproszone w abstrakcji od platformy i języka nie byłoby możliwe, gdyby nie wcześniejsze ustalenia jak się porozumiewać, jak przekazywać sobie różnorodne parametry i wyniki. Niewątpliwym standardem, wykorzystywanym tutaj jawi się **XML** (wraz z MIME) jako bardzo elastyczne medium wymiany dowolnej informacji pomiędzy platformami, językami czy programami/procesami. Dokumenty XML przesyłane są standardowymi protokołami transportowymi, takimi jak.: HTTP, HTTPS, SMTP, FTP, RMI/IIOP czy MQSeries.

Standard XML jest jednak zbyt elastyczny, wymaga on stosownego ograniczenia, tak aby strony biorące udział w komunikacji mogły np. jednoznacznie odróżnić nazwę parametru od jej wartości. Takim standardem jest **SOAP** - Simple Object Access Protocol, język oparty na XML-u. Jest to protokół, który specyfikuje jaka usługa (procedura) ma zostać wywołana, jakie mają być parametry jej wywołania i jak ma być zwrócony rezultat jej działania. Dane mogą być uzupełnione dodatkową informacją służącą np. do optymalizacji, skalowania, zachowania bezpieczeństwa, itp.

Ponieważ konfiguracja korzystania z usługi ma docelowo odbywać się automatycznie, potrzebny jest język opisu usług – **WSDL** – Web Services Description Language, również oparty na XML-u. Dokument WSDL opisuje, na jakich serwerach znajdują się jakie usługi, zawiera opisy parametrów (typy danych, opcjonalność) oraz może zawierać cały szereg informacji opisowych o charakterze komentarzowym.

Całość dopełnia standard **UDDI** – Universal Description, Discovery and Integration, który służy jako rejestr usług wraz z metodami jego aktualizacji i przeszukiwania.

Należy zaznaczyć tutaj, że nie są to jedyne obecnie używane standardy. Niektóre z wymienionych tutaj mają swoje odpowiedniki lansowane przez różne firmy w postaci propozycji lub gotowych standardów. Istnieje również cała gama innych protokołów towarzyszących, realizujących funkcje pomocnicze, np.: WDDX, XML-RPC, WSEL, WSFL, ebXML, XLANG, ADS, DISCO, WSIF, XKMS, SAML, WS-Security, XACML i wiele innych.

3. Przykład

Bieżący rozdział zawiera wiele mało czytelnych plików XML. Ich gruntowne studiowanie nie jest konieczne do zrozumienia samej idei usług sieciowych. Większość z nich została wygenerowana automatycznie przez narzędzi, które przy pomocy myszki potrafią modelować i generować stosowne procedury i skrypty instalacyjne. Znajomość szczegółów protokołów SOAP czy WSDL staje się potrzebna dopiero podczas korzystania z zaawansowanych opcji lub usuwania usterek.

Pokazane tu zastosowanie zostało przetestowane w środowisku Apache Axis, lecz bez większych trudności może być przeniesione do OC4J lub innego serwera SOAP.

3.1. Tworzenie i korzystanie z usługi WebServices

Tworzymy prosty przykład, którego zadaniem będzie pokazać maksymalną pensję pracownika określonego departamentu (korzystamy tutaj ze standardowej aplikacji *demo*: human_resources).

```

import oracle.jdbc.driver.*;
import oracle.jdbc.*;
import java.sql.*;

public class MaxSalary {
    public static double doCheck(int Dept) {
        double rt = 0.0;
        String dpt = String.valueOf(Dept);
        try {
            DriverManager.registerDriver(new
                oracle.jdbc.driver.OracleDriver());
            Connection conn =
                DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:ora", "hr", "hr");
            Statement stmt = conn.createStatement();
            ResultSet rset = stmt.executeQuery(
                "select max(salary) from Employees "
                + " where department_id = "+dpt);
            rset.next();
            rt = rset.getDouble(1);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return rt;
    }
}

```

Nagrywamy plik jako *MaxSalary.jws* i możemy wywołać go jako usługę sieciową, podając w przeglądarce odpowiedni URL do tego pliku: <http://localhost:8080/axis/MaxSalary?method=doCheck&Dept=100>. Pokazana wyżej klasa jest w rzeczywistości zwykłym serwletem, który pełni tutaj rolę klienta SOAP. Uwalnia nas to (w tym konkretnym przypadku, w którym chcemy wyświetlić rezultat w oknie przeglądarki) od tworzenia wywołania SOAP-owego.

Rezultat działania usługi przedstawia poniższy listing:

```

<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
    <doCheckResponse
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <doCheckReturn href="#id0" />
    </doCheckResponse>
    <multiRef id="id0" soapenc:root="0"
        soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
        xsi:type="xsd:double"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        12000.0
    </multiRef>
</soapenv:Body>
</soapenv:Envelope>

```

Dalej już nietrudno dopisać stosowny arkusz XSL, który sprawdzi, czy nie wystąpił błąd i ewentualnie wyświetli w atrakcyjnej formie rezultat (na powyższym listingu został on wyluszczo-ny).

Oczywiście nic nie stoi na przeszkodzie, abyśmy sami nie utworzyli wywołania SOAP. Jak? Jeżeli nie znamy specyfikacji usługi, możemy pobrać opis usługi w formacie WSDL (<http://localhost:9090/axis/MaxSalary?wsdl>), którego fragment widać poniżej:

```
<wsdl:message name="doCheckRequest">
  <wsdl:part name="Dept" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="doCheckResponse">
  <wsdl:part name="doCheckReturn" type="xsd:double"/>
</wsdl:message>
<wsdl:portType name="MaxSalary">
  <wsdl:operation name="doCheck" parameterOrder="Dept">
    <wsdl:input message="impl:doCheckRequest" name="doCheckRequest"/>
    <wsdl:output message="impl:doCheckResponse" name="doCheckResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:service name="MaxSalaryService">
  <wsdl:port binding="impl:MaxSalarySoapBinding" name="MaxSalary">
    <wsdlsoap:address
      location="http://localhost:9090/axis/MaxSalary.jws"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Analizując powyższy plik można łatwo znaleźć w nim adres, nazwę usługi i listę parametrów oraz utworzyć wywołanie SOAP zgodne z powyższą specyfikacją:

```
POST /axis/MaxSalary.jws HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 424
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
  <doCheck>
    <arg0 xsi:type="xsd:int">100</arg0>
  </doCheck>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.2. Publikacja i wyszukiwanie usługi

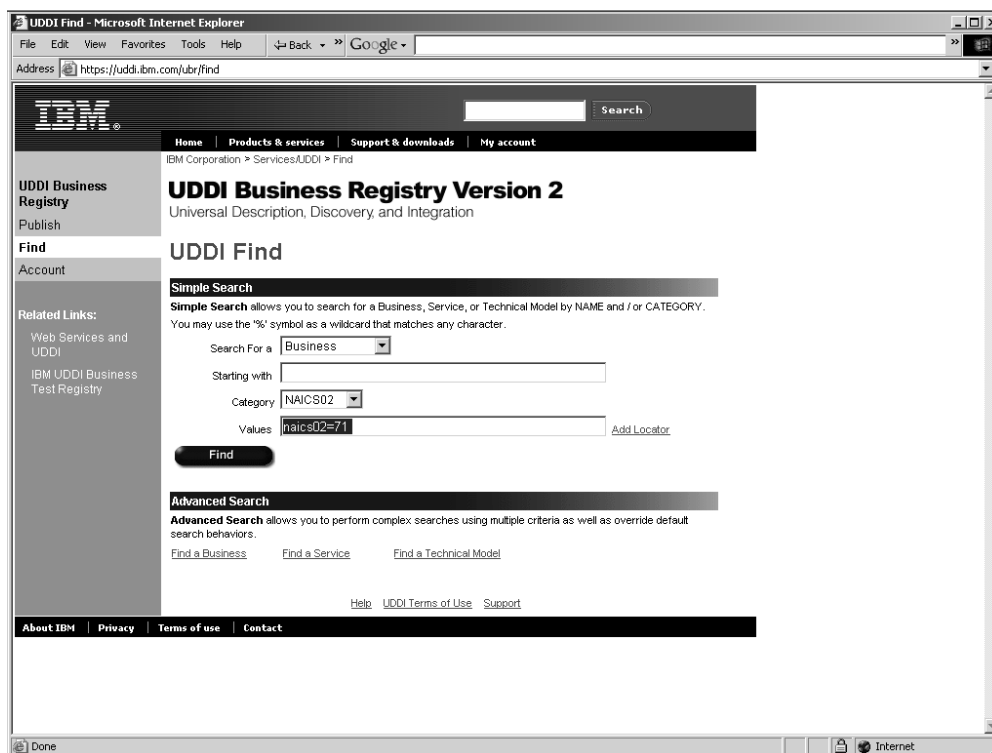
Przypuśćmy, że chcemy teraz informować wszystkie zainteresowane podmioty na całym świecie o tym ile zarabia się w naszej firmie. Jedną z metod jest umieszczenie specyfikacji WSDL na naszej stronie WWW. Tylko skąd potencjalny klient ma wiedzieć, gdzie jej szukać, jeżeli zupełnie nie ma pojęcia o naszym istnieniu?

W tym celu należy umieścić specyfikację usługi w *rejestrze usług*. Rejestry usług UDDI można podzielić na dwie kategorie: prywatne i operatora. Prywatne, jak sama nazwa wskazuje, pozwalają na przechowywanie danych o usługach, głównie z przeznaczeniem do wykorzystania przez nas samych lub naszych partnerów biznesowych. Rejestry operatora, są to publiczne bazy danych prowadzone przez firmy trzecie. Można w miarę łatwo ustalić reguły wzajemnego synchronizowania danych między bazami UDDI. Najbardziej popularnymi z rejestrów publicznych są dwa zlokalizowane

pod adresami: <http://uddi.ibm.com> oraz <http://uddi.microsoft.com>. Rejestry te wzajemnie się synchronizują. (Niestety należy z ubolewaniem stwierdzić, że często są one wykorzystywane do testowania i ok. 48% wpisów jest fikcyjna lub niepełna).

Rejestr UDDI to jednak coś więcej niż tylko rejestr firm i usług. Standard ten definiuje również zbiór struktur danych i specyfikację API umożliwiające programową rejestrację i wyszukiwanie firm, usług, wiązań i typów usług.

Dostęp do formularza rejestracji i wyszukiwania możliwy jest nie tylko z poziomu API, ale również poprzez strony WWW.



Rys. 3. Wyszukiwanie usług w rejestrze usług UDDI.

4. Inne możliwości usług sieciowych

Podany w poprzednim rozdziale przykład nie wyczerpuje wszystkich możliwości SOAP-a. Ze względu na limit objętościowy artykułu poniżej wspomniano tylko o najważniejszych cechach. Czytelnik łatwo znajdzie konkretne zastosowania i kody przykładowych aplikacji [np. GSB+03].

- Koperta SOAP (czyli treść komunikatu) może zawierać szereg specyficznych rozszerzeń, które wzbogacają podstawowe możliwości usług sieciowych. Nagłówki tworzą bardzo elegancki, a jednocześnie prosty mechanizm rozszerzania komunikatów SOAP w sposób zdecentralizowany. Typowe ich zastosowania dotyczą takich zadań jak: uwierzytelnianie, autoryzacja, obsługa transakcji, przetwarzanie wpłat, śledzenie, itp.
- Poprzez zastosowanie kodowania base64 lub formatu MIME można przysyłać dane dowolnego typu, w tym również binarne.
- Komunikat SOAP może być również przetwarzany przez szereg serwerów (pośredników), każdy z nich może mieć dostęp do różnych części komunikatu SOAP. Pośredniki są doskonałym mechanizmem zabezpieczania danych przy przekraczaniu granic domen zaufania oraz wszędzie tam, gdzie zachodzi potrzeba skalowalności systemu.

- Istnieje kilka modeli dystrybucji komunikatów SOAP: jeden do jednego, jeden do wielu lub wiele do wielu. Połączenia mogą być synchroniczne lub asynchroniczne. Komunikaty mogą być kolejgowane (np. poprzez mechanizm AQ). Do przesyłania komunikatów można używać nie tylko protokołu HTTP, ale również innych: FTP, SMTP, itp.
- Szczególny nacisk w usługach sieciowych poświęcono na aspekty związane z bezpieczeństwem. Poprzez infrastrukturę klucza publicznego PKI możliwe jest zabezpieczenie zarówno transmisji komunikatów jak i samej treści komunikatu, np w celu ukrycia fragmentów zawartości przed niektórymi pośrednikami.
- Protokół SOAP został wybrany jako najlepszy i najbardziej popularny protokół transportowy do przesyłania komunikatów dla transakcji BTP². Tradycyjne transakcje, blokujące zasoby przez długi czas, nie są najlepszym rozwiązaniem w sytuacji, gdy pojedyncze operacje mogą trwać godzinami.

5. Usługi sieciowe a Oracle

5.1. Serwlet XSQL

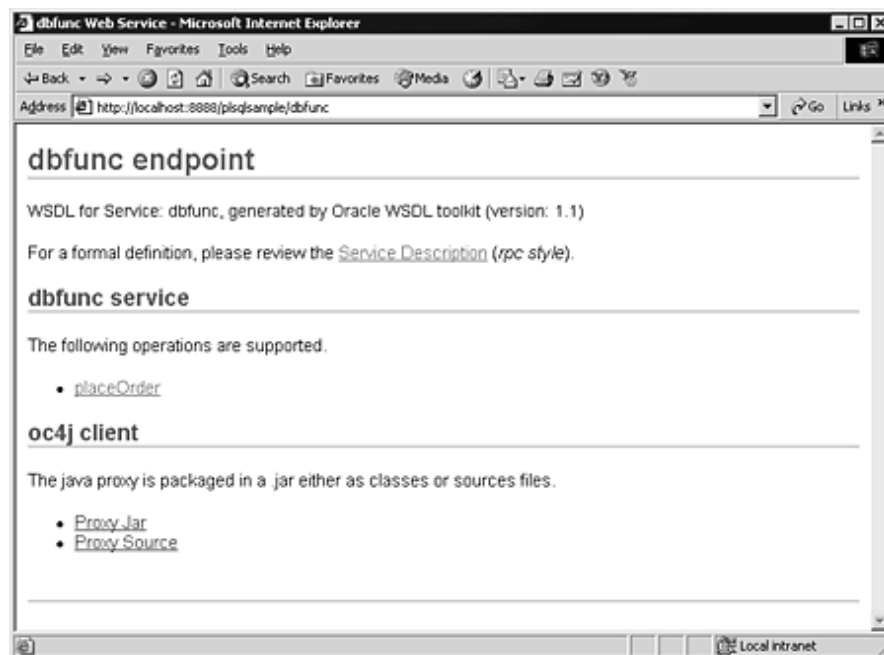
Serwlet XSQL sam w sobie nie jest serwerem SOAP, ale łatwo go doń przystosować. Wystarczy przypomnieć, że zarówno żądanie, jak i odpowiedź SOAP są dokumentami XML, które mogą po przekształceniu XSL mogą stać się parametrami tego serwletu i posłużyć do wykonania jakiejś akcji w bazie danych. Na przykład:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="MaxSalarySoapResponse.xsl"?>
<page connection="xmldemo"
  allow-client-style="no"
  xmlns:xsql="urn:oracle-xsql"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <xsql:set-page-param
    name="code"
    xpath="/SOAP-ENV:Envelope/SOAP-ENV:Body/SOAP-ENV:doCheck/arg0"/>
  <xsql:include-param name="Dept"/>
  <xsql:query bind-params="Dept" tag-case="lower">
    SELECT max(Salary) FROM Employees WHERE Department_id = ?
  </xsql:query>
</page>
```

5.2. Application Server i OC4J

Zarówno Application Server 10g jak i OC4J posiadają wbudowaną obsługę komunikatów SOAP. Podobnie jak to miało miejsce w rozdziale 3 ich obsługa może również odbywać się poprzez „zwykłe” wywołania WWW. Podany poniżej przykład realizuje wywołanie funkcji PL/SQL poprzez wysłanie żądania SOAP (wcześniej należało skonfigurować serwer – wystarczył do tego kilkulinijkowy dokument XML i trzy polecenia [Pri04]):

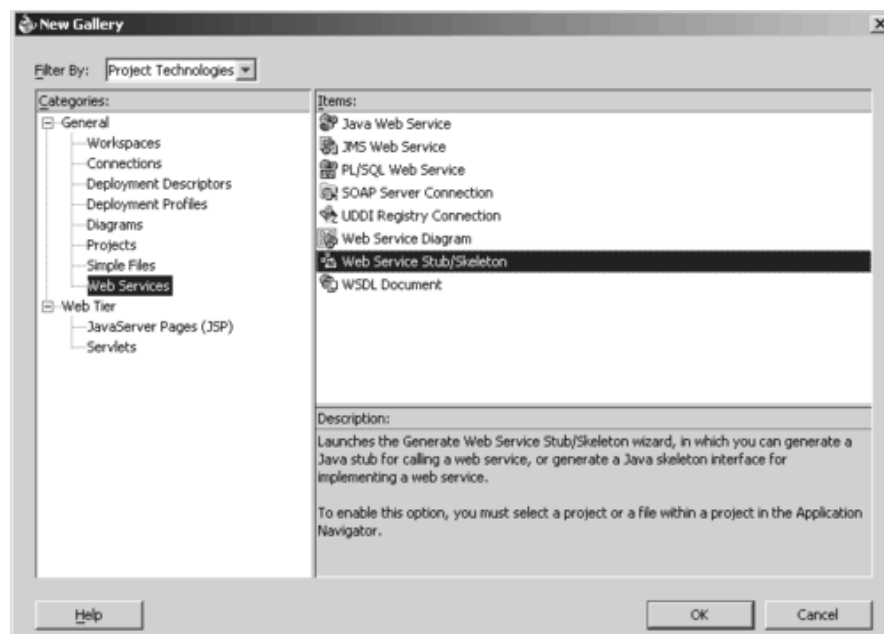
² Patrz http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction



Rys. 4. Realizacja usługi poprzez funkcję PL/SQL.

5.3. JDeveloper 10g

Przy pomocy narzędzia JDeveloper 10g można utworzyć usługę sieciową, opublikować w rejestrze UDDI oraz dokładnie ją przetestować. Zaletą tego narzędzia jest to, że usługi te można modelować w UML-u i integrować z resztą aplikacji (EJB, klasy Javy, procedury PL/SQL, usługi .NET). Dodatkową zaletą JDevelopera jest to, że posiada nawigator usług zgodny z rejestrem UDDI.

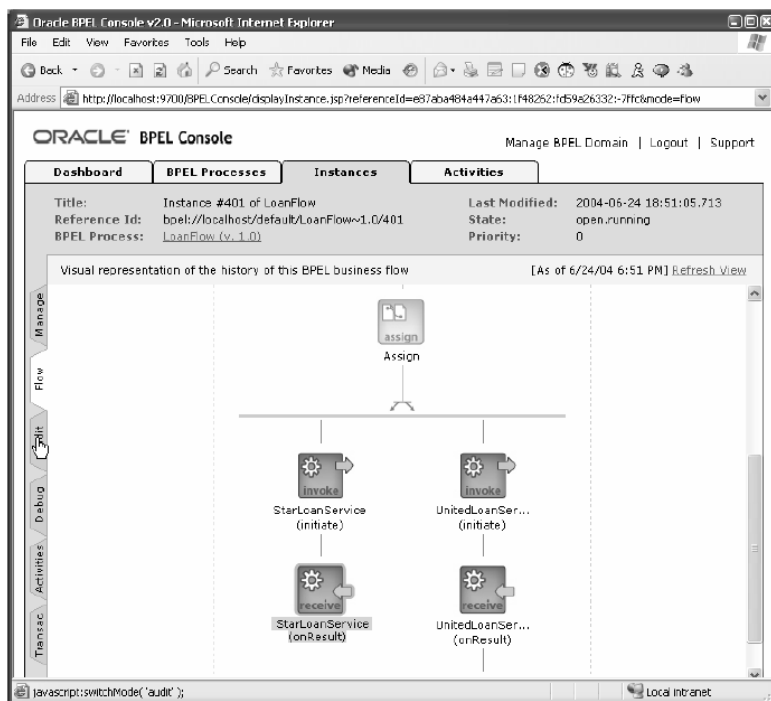


Rys. 5. Kreator usługi w JDeveloper 10g.

5.4. Oracle BPEL Process Manager

Wspomniane w tytule narzędzie to graficzny interfejs obsługi języka o tym samym akronimie – BPEL – Business Process Execution Language. Przy jego pomocy można zamodelować skompliko-

wane procesy biznesowe i zaimplementować je jako usługi sieciowe. Na szczególną uwagę zasługuje taka implementacja usług, która pozwala na synchronizację podścieżek w sytuacji, gdzie proces biznesowy rozdwaja się, aby po kilku kolejnych krokach z powrotem złączyć się w jedną ścieżkę.



Rys. 6. Modelowanie procesów w Oracle BPEL.

5.5. JPublisher

JPublisher to komplet narzędzi przeznaczonych do używania w środowisku Javy. Pozwala m.in. na generowanie klas Javy odpowiadających strukturze istniejących tabel oraz pakietów PL/SQL. Struktury te można następnie wykorzystywać w swoich aplikacjach, w szczególności do tworzenia usług sieciowych.

Przykładowo, następujący zestaw opcji:

```
-sqlstatement.class=MySQLStatements
-sqlstatement.getEmp="select ename from emp
                    where ename=:myname VARCHAR}"
-sqlstatement.return=both
```

wygeneruje kod klasy Javy, realizujący wspomniane zapytanie. Po umieszczeniu tej klasy w środowisku serwera (np. OC4J), będzie można zapewnić dostęp do danych z poziomu SOAP.

5.6. UDDI

Rejestr usług jest oddzielną aplikacją instalowaną w środowisku Oracle Application Server lub OC4J (do pobrania z OTN). Rejestr jest w pełni zgodny z wersjami UDDI v1 i v2 oraz posiada niektóre ważniejsze cechy nowej jeszcze wersji v3. Dostęp do rejestru odbywa się poprzez standardowe API lub strony WWW. Możliwe jest również zarządzanie rejestrem z poziomu Oracle Enterprise Managera.

Bibliografia

1. [ADG02] Anzbock R., Dustdar S., Gall H.: Software configuration, distribution, and deployment of Web-Services, ACM 2002

2. [CoWe02] Computer Weekly, 10 kwietnia 2002, „Ellison knocks Webservices”
3. [CrKa04] Crawford W., Kaplan J.: J2EE – Stosowanie wzorców projektowych, Helion 2004
4. [CSK01] Chang B., Scardina M., Kiritzov S.: Oracle 9i XML Handbook, Osborne/McGraw-Hill, 2001, ISBN 0-07-213495-X
5. [GSB+03] Graham S., Simeonov S., Boubez T., Davis D., Daniels G., i in.: Java Usługi WWW Vademecum profesjonalisty, Helion 2003.
6. [HeMa03] Heuvel W., Maamar Z.: Communication of the ACM, October 2003/Vol 46. No 10.
7. [Kreg01] Kreger H.: Web services conceptual architecture (WSCA 1.0), IBM Software Group, May 2001 (<http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>)
8. [Leh04] Lehmann M.: Your first BPEL project. Oracle Magazine 09-10/2004
9. [Lit03] Little M.: Transactions and Web services, Communication of the ACM, October 2003/Vol. 46, No. 10
10. [Mue02] Mueller J.P.: Poznaj SOAP, Mikom 2002
11. [Neil00] O’Neill M.: Usługi WWW – Bezpieczeństwo, Edition2000 ISBN 83-7366-071-2
12. [Ora1] Dokumentacja Oracle, Oracle 10g HTML DB User’s Guide (PN: 10992-01)
13. [Ora2] Dokumentacja Oracle, Oracle 10g JPublisher User’s Guide (PN: 10983-01)
14. [Pri04] Prise J.: Build a PL/SQL Web Service, OTN
15. [Ship03] Shiping R.: A model of Web services discovery with QoS, ACM 2003
16. [Spic02] Spicer J.: Getting Good Service, Oracle Magazine march 2002.