Programowanie współbieżne i rozproszone

WYKŁAD 7

Jan Kazimirski

Programowanie serwisów WEB SOAP

Literatura

- **Programming Web Services with SOAP**, D. Tidwell, J. Snell, P. Kulchenko, O'Reilly, 2001
- **Understanding Web Services - XML, WSDL, SOAP and UDDI**, E. Newcomer, Addison-Wesley, 2002

Co to jest Web Service?

- Web service – dostępny przez internet interfejs do funkcji aplikacji
- Interfejs używa standardowych technologii (HTTP, XML, SMTP itp.)
- Web service tworzy warstwę abstrakcji między kodem aplikacji a jej klientem
- Web service pozwala na łatwy dostęp do aplikacji niezależnie od platformy i szczegółów klienta.

Co to jest Web Service c.d.

- Web service definiuje protokół komunikacji między aplikacją i klientem.
- Zwykle usługa web service polega na zdalnym wywołaniu określonej funkcji z danymi parametrami i odebranie rezultatu
- Aplikacja web service nie musi mieć określonej struktury.

Serwisy webowe i JIT

- Podejście J-I-T - „Just in time”
- W przypadku serwisów podejście zakłada, że klient nie jest na sztywno związany z usługą.
- Architektura serwisów zakłada istnienie „dostawcy serwisów” (**service provider**) który publikuje dostępne serwisy.
- Klient (**service consumer**) ma dostęp do listy serwisów (**service registry**) i może wybrać odpowiedni serwis wtedy gdy jest potrzebny (late binding)

Architektura WS

- Architektura serwisu webowego ma strukturę warstwową.
  - Discovery
  - Description
  - Packaging
  - Transport
  - Network
- Dodatkowo wyróżnia się też warstwę aplikacji zawierająca kod danego serwisu webowego.

Architektura WS c.d.

- Dodatkowe aspekty mogą być realizowane przez protokoły poza głównymi warstwami, np..
  - **XML-Dsig** – podpis cyfrowy dokumentu XML
  - **XML-Enc** – szyfrowanie dokumentu XML
  - **XKMS** – zarządzanie PKI
  - ...

Discovery

- Warstwa Discovery pozwala klientowi uzyskać informację o dostawcach serwisu webowego.
- Stosowane mechanizmy:
  - **UDDI** - Universal Description, Discovery and Integration - standard sponsorowany przez konsorcjum OASIS.
  - **WS-Inspection** – alternatywny standard (IBM, Microsoft)

Description

- Aby skorzystać z serwisu klient musi uzyskać informacje o protokołach używanych przez serwis
- Do opisu szczegółów korzystania z serwisu stosuje się różne mechanizmy np.,
  - **WSDL** - Web Service Description Language
  - **RDF** – Resource Description Framework
  - **DAML** - DARPA Agent Markup Language

Packaging

- Dane przekazywane w sieci wymagają „opakowania” tak, aby mogły być odczytane niezależnie od rodzaju klienta
- Przykłady protokołów:
  - **HTML** – niezbyt wygodny bo mocno związany z prezentacją danych
  - **XML** – bardzo wygodny, często używany jako format bazowy
  - **SOAP** – często stosowany format oparty o XML

Transport

- Warstwa określa protokół transportowy stosowany do przekazywania danych w sieci.
- Można stosować dowolne protokoły HTTP, SMTP, Jabber, itp..
- Często stosuje się HTTP ze względu na ustawienia routerów i zabezpieczeń (firewall).
- Protokół Jabber (XMPP) pozwala na wygodną komunikację asynchroniczną

Network

- Warstwa sieci odpowiada warstwie sieci w typowym stosie TCP/IP tzn. odpowiada za
  - Adresowanie
  - Trasowanie
  - Podstawową komunikację

Model P2P WS

- Typowa sytuacja zakłada ostry rozdział dostawców i konsumentów serwisów. Często nie jest to prawda
- Model P2P zakłada, że aplikacja może być zarówno dostawcą, jak i klientem.
- Przykład: komunikatory internetowe
  - Serwer pełni rolę rejestru serwisów
  - Komunikator klienta może pełnić rolę zarówno dostawcy, jak i konsumenta serwisu

SOAP

- Protokół definiujący sposób „opakowania” wiadomości przesyłanych między aplikacjami
- Definiuje opartą na XML-u „kopertę” oraz zestaw reguł określający sposób tłumaczenia przesyłanych danych na format XML
- Ze względu na elastyczność i prostotę cieszy się dużą popularnością.

XML

- Elastyczny standard przekazywania informacji w formie zrozumiałej i dla człowieka i dla komputera
- W odróżnieniu od HTML-a, XML koncentruje się na treści. XML pozwala na separację treści i sposobu jej prezentacji
- XML umożliwia automatyczną walidację dokumentu
- Ze względu na czysto tekstową formę XML jest niezależny od platformy sprzętowej

XML c.d.

- XML jest meta-językiem (definiującym inne języki)
- Jest podzbiorem języka SGML (używanego m.in. do definicji języka HTML)
- SGML jest złożony i trudny do automatycznego parsowania
- SGML używany jest zwykle przez duże instytucje, założeniem XML-a jest prostota i możliwość użycia przez niespecjalistów.

XML - Podstawy

- XML jest (podobnie jak HTML) językiem znaczników (tag) i atrybutów.
- Przykład użycia znacznika:

  <h1>Tytuł</h1>
- Przykład znacznika z atrybutem

  <image source=”img.png” />

XML a HTML

- W XML wielkość znaków jest istotna
- Wartości atrybutów muszą być ujęte w cudzysłów lub apostrof
- Elementy muszą być zamykane
- Elementy muszą być prawidłowo zagnieżdżone

XML DTD

- DTD (Document Type Definition) – określa w jaki sposób XML jest używany w danym dokumencie
- DTD nie jest wymagany ale jego zdefiniowanie zwiększa bezpieczeństwo i wygodę poprzez m.in.
  - Możliwość walidacji dokumentu
  - Możliwość stosowania makr
  - Dostarczenie wartości domyślnych atrybutów

XML – RPC i EDI

- **RPC** – Remote Procedure Call – Zdalne wywoływanie funkcji.
  - Jeden program wywołuje funkcję w innym programie przekazując do niego argumenty i odbierając rezultar
- **EDI** – Electronic Document Interchange – Elektroniczna wymiana dokumentów
  - Aplikacje wymieniają się danymi – np. dane biznesowe, transakcje handlowe itd.
- W obu przypadkach można wykorzystać XML.

XML – RPC i EDI c.d.

- XML zapewnia wygodny format przekazywania danych zarówno dla RPC, jak i EDI.
- Poza decyzją o stosowaniu XML potrzebne są jeszcze dodatkowe informacje:
  - Jakie dane są przekazywane?
  - Jak są reprezentowane za pomocą XML?
  - Jak należy je przetwarzać?
- **SOAP** dostarcza tych informacji.

SOAP – szkielet komunikatu

- Komunikat SOAP składa się z kilku elementów:
  - **Koperta** (Envelope) – zewnętrzny kontener zawierający całą wiadomość SOAP
  - **Nagłówek** (Header) – Dodatkowe informacje dotyczące przetwarzania wiadomości SOAP
  - **Ciało** (Body) – Treść wiadomości
- Budowę komunikatu SOAP określa definicja:
  http://www.w3.org/2001/06/soap-envelope

Budowa komunikatu SOAP c.d.

- Element **Envelope** musi zawierać dokładnie jeden element **Body**.
- Element **Body** może zawierać dowolną liczbę elementów.
- Element **Envelope** może też zawierać element **Header**. Jeżeli tak, to musi on być jeden i umieszczony przed elementem **Body**.
- Element **Header** składa się z bloków.

Przykład SOAP - RPC

- Wymiana komunikatów w przypadku RPC zwykle odbywa się parami (choć nie jest to konieczne)
  - Klient SOAP wysyła do serwera SOAP zapytanie
  - Serwer SOAP wysyła do klienta SOAP odpowiedź
- Przykładowo funkcja dostępna na serwerze SOAP może mieć postać:

  public Float getQuote(String symbol);

RPC SOAP - Zapytanie

```
<s:Envelope
 xmlns:s="http://www.w3.org/2001/06/soap-envelope">
    <s:Header>
       <m:transaction xmlns:m="soap-transaction"
                      s:mustUnderstand="true">
          <transactionID>1234</transactionID>
       </m:transaction>
    </s:Header>
    <s:Body>
       <n:getQuote xmlns:n="urn:QuoteService">
          <symbol xsi:type="xsd:string">
             IBM
          </symbol>
       </n:getQuote>
    </s:Body>
</s:Envelope>
```

RPC SOAP - Odpowiedź

```
<s:Envelope
 xmlns:s="http://www.w3.org/2001/06/soap-envelope">
    <s:Body>
       <n:getQuoteRespone
              xmlns:n="urn:QuoteService">
          <value xsi:type="xsd:float">
             98.06
          </value>
       </n:getQuoteResponse>
    </s:Body>
</s:Envelope>
```

Atrybut mustUnderstand

- W pewnych sytuacjach niektóre bloki nagłówka mogą być niezrozumiałe dla odbiorcy, ale nie wpływa to na odbiór komunikatu.
- Ustawienie atrybutu **mustUnderstand** na wartość true powoduje odrzucenie przez odbiorcę całego komunikatu jeżeli dany blok nagłówka nie jest dla niego zrozumiały.

Atrybut encodingStyle

- Określa sposób kodowania danych (wymagany przy RPC)
- Pozwala przekazywać dane między platformami różniącymi się ich reprezentacją.
- Definicje typów danych używany w RPC
  http://www.w3.org/2001/06/soap-encoding
- EDI zwykle stosuje własne definicje danych wiadomości.

Błąd SOAP

- Błędy przetwarzania wiadomości SOAP raportowane są poprzez specjalny typ komunikatu zawierającego elementy:
  - **faultcode** – Kod identyfikujący błąd (QName)
  - **faultstring** – Rodzaj błędu
  - **faultactor** – miejsce wystąpienia
  - **details** – szczegółowe informacje o błędzie

Błąd SOAP - przykład

```
<s:Envelope xmlns:s="...">
   <s:Body>
      <s:Fault>
       <faultcode>Client.Authentication</faultcode>
       <faultstring>
          Invalid credentials
       </faultstring>
       <faultactor>http://acme.com</faultactor>
       <details>
          <!-- application specific details -->
       </details>
      </s:Fault>
   </s:Body>
</s:Envelope>
```

Standardowe kody błędów SOAP

- Standard definiuje 4 kody błędów:
  - **VersionMismatch** – Błędna wersja SOAP
  - **MustUnderstand** – niezrozumiały nagłówek z atrybutem mustUnderstand
  - **Server** – Błąd po stronie serwera
  - **Client** – Błąd wiadomości
- Standardowe kody mogą być rozszerzane, np. Client.Authentication może oznaczać błąd autentykacji klienta.

Ścieżka przetwarzania komunikatu SOAP

- W założeniu komunikat SOAP przekazywany jest od nadawcy do odbiorcy.
- Standard dopuszcza możliwość istnienia „pośredników” (actor) przetwarzających wiadomość pomiędzy nadawcą i odbiorcą
- Definicja przekazywania komunikatu do pośredników nie jest częścią SOAP (istnieją rozszerzenia np. WS-Routing Microsoftu)
- SOAP dostarcza mechanizm identyfikacji bloków nagłówka adresowanych do pośredników

Atrybut actor

- Do identyfikacji pośredników służy atrybut actor bloku nagłówka. Np.

```
<s:Envelope xmlns:s="...">
   <s:Header>
      <x:signature actor="uri:SignatureVerifier">
         ...
      </x:signature>
   </s:Header>
   <s:Body>
      <abc:purchaseOrder>...</abc:purchaseOrder>
   </s:Body>
</s:Envelope>
```

SOAP i Web Serwisy

- Web serwisy są jednym z głównych zastosowań protokołu SOAP
- Wykorzystują one trzy typy komunikatów:
  - Wywołanie metody
  - Odpowiedź
  - Komunikat o błędzie
- Komunikaty o błędach wykorzystują standardową wiadomość o błędzie SOAP.

WS – przykładowa metoda

- Przykład metody:

  String checkStatus(String orderCode,
  String customerID);

- Przykładowe wywołanie:

  result = checkStatus("abc123", "Bob's Store")

SOAP – Wywołanie metody i rezultat

```
  <s:Envelope xmlns:s="...">
    <s:Body>
      <checkStatus xmlns="..."
                   s:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
        <orderCode xsi:type="string">abc123</orderCode>
        <customerID xsi:type="string">
          Bob's Store
        </customerID>
      </checkStatus>
    </s:Body>
  </s:Envelope>
```

```
<s:Envelope xmlns:s="...">
  <s:Body>
    <checkStatusResponse
     s:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
      <return xsi:type="xsd:string">new</return>
    </checkStatusResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

Kodowanie danych SOAP

- Specyfikacja SOAP zawiera definicję sposobu kodowania danych. Nie jest ona obowiązkowa, można użyć własnej.
- Wszystkie typy danych użyte w komunikacie muszą być zdefiniowane (lub wywodzić się z typów zdefiniowanych) w definicji kodowania danych.
- Typ elementu w komunikacie może być albo określony bezpośrednio, albo poprzez odwołanie do odpowiedniego dokumentu z definicją typu.

Typy złożone

- Standard definiuje sposób reprezentacji typów złożonych: struktur i tablic
- Łańcuchy znakowe traktowane są jako typy proste, a nie tablice.
- Tablice danych binarnych powinny być kodowane base64
- Standard pozwala przekazywać również fragmenty tablic oraz tablice rzadkie.
- Specjalny atrybut określa, że dana wartość jest pusta

SOAP – przykłady kodowania tablic

```
<some_array xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="se:string[3]">
   <se:string>Joe</se:string>
   <se:string>John</se:string>
   <se:string>Marsha</se:string>
</some_array>

<data xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[2][]">
   <names href="#names-1"/>
   <names href="#names-2"/>
</data>
<names id="names-1" xsi:type="SOAP-ENC:Array"
       SOAP-ENC:arrayType="xsd:string[2]">
    <name>joe</name>
    <name>john</name>
</names>
<names id="names-2" xsi:type="SOAP-ENC:Array"
       SOAP-ENC:arrayType="xsd:string[2]">
    <name>mike</name>
    <name>bill</name>
</names>
```

Przekazywanie komunikatów SOAP

- Komunikaty SOAP mogą być przekazywane za pomocą różnych protokołów transportowych, HTTP, FTP, SMTP, TCP itd.
- Ze względu na powszechność najbardziej rozpowszechnione jest używanie HTTP
- Specyfika protokołu HTTP (zapytanie-odpowiedź) odpowiada specyfice używania SOAP RPC.

Tworzenie Web Serwisu

- Web Serwis składa się z 3 elementów
  - Odbiorca wiadomości (listener) – odbiera komunikat SOAP.
  - Pośrednik (proxy) – tłumaczy komunikat SOAP na natywne wywołanie metody.
  - Natywny kod realizujący zadaną funkcję
- Dla wielu języków programowania dostępne są gotowe biblioteki ukrywające szczegóły protokołu i ułatwiające tworzenie Web Serwisów.

Definiowanie Web Serwisu SOAP

- Serwisy SOAP można definiować za pomocą specjalnych opisów w języku WSDL
- **WSDL** (Web Services Description Language) – oparty o XML język opisu Web Serwisu
- Zalety WSDL
  - Zapewnia przejrzysty opis interfejsu serwisu
  - Pozwala na tworzenie bardziej przejrzystych klientów (automatyzacja działań klienta)
  - Modyfikacje interfejsu serwisu nie mają wpływu na kod klienta.

Tworzenie plików WSDL

- Ręczne tworzenie opisów serwisu może być złożone i podatne na błędy
- Dostępne są narzędzia pozwalające tworzyć definicję WSDL na bazie istniejącego serwisu
- Tworząc Web Serwisy nie mamy obowiązku tworzyć ich definicji WSDL, ale ich użycie znacznie ułatwia dostęp do serwisu

Elementy opisu Web Serwisu

- Plik WSDL opisuje następujące elementy Web Serwisu:
  - Typy danych
  - Komunikaty
  - Interfejsy
  - Serwisy

WSDL – Typy danych

- Web Serwisy powinny być niezależne od platformy, tzn. nie powinny zakładać zależnej od platformy reprezentacji danych
- W pliku WSDL należy umieścić definicje typów używanych w komunikatach
  - Za pomocą odwołania do odpowiedniej definicji (element **wsdl:import**)
  - Za pomocą umieszczenia odpowiednich definicji bezpośrednio w opisie (element **wsdl:types**).

WSDL – Typy danych c.d.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="HelloWorldDescription"
   targetNamespace="urn:HelloWorld"
   xmlns:tns="urn:HelloWorld"
   xmlns:types="urn:MyDataTypes"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

   <wsdl:import namespace="urn:MyDataTypes"
                location="telephonenumber.xsd" />

</wsdl:definitions>
```

WSDL - Interfejs

- Definicja interfejsu WS obejmuje:
  - Komunikaty wejściowe (zbiór argumentów potrzebnych do wykonania operacji)
  - Komunikaty wyjściowe (wartości będące rezultatem operacji)
  - Definicje błędów, które mogą wystąpić w czasie wykonywania operacji

WSDL – Interfejs c.d.

```
<definitions ...>
  <wsdl:message name="sayHello_IN">
    <part name="name" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="sayHello_Out">
    <part name="greeting" type="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="HelloWorldInterface">
    <wsdl:operation name="sayHello">
      <wsdl:input message="tns:sayHello_IN" />
      <wsdl:output message="tns:sayHello_OUT" />
    </wsdl:operation>
  </wsdl:portType>
  ...
</definitions>
```

WSDL – Interfejs c.d.

- Element **portType** definiuje operację
- Operacja wymaga komunikatu wejściowego i generuje komunikat wyjściowy
- Komunikat wejściowy składa się pojedynczego parametru 'name' typu string
- Komunikat wyjściowy jest parametrem typu string.

WSDL - Implementacja

- Opis WSDL serwisu poza interfejsem może też definiować implementację serwisu
- Elementy opisu implementacji serwisu:
  - Binding – powiązanie serwisu ze specyficznymi protokołami (np.. HTTP)
  - Service – położenie serwisu w sieci.

WSDL - Binding

```
<wsdl:binding name="HelloWorldBinding"
              type="tns:HelloWorldInterface">

 <soap:binding style="rpc"
   transport="http://schemas.xmlsoap.org/soap/http"/>

 <wsdl:operation name="sayHello">
   <soap:operation soapAction="urn:Hello" />

    <wsdl:input>
      <soap:body use="encoded"
                 namespace="..."
                 encodingStyle="..." />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="encoded"
                 namespace="..."
                 encodingStyle="..." />
    </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
```

WSDL - Binding

- Na podstawie informacji o powiązaniu serwisu z protokołami transportu, klient może automatycznie utworzyć wymagany komunikat
- W szczególnym przypadku wiązania do HTTP-GET parametry mogą być przekazane jako część adresu serwisu (element http:urlEncoded)

WSDL Service

- Definiuje miejsce (adres w sieci) gdzie serwis jest dostępny
- W jednej definicji WSDL można zdefiniować wiele różnych adresów serwisu (np. dla różnych wersji lub implementacji serwisu).

WSDL Service c.d.

```
<wsdl:service name="HelloWorldService">
  <wsdl:port name="HelloWorldPort"
             binding="tns:HelloWorldBinding">
    <soap:address location="http://localhost:8080" />
  </wsdl:port>
</wsdl:service>
```

„Odkrywanie” Web Serwisów

- Opis WSDL pozwala klientowi określić jak wywołać określony serwis
- Jak klient może znaleźć interesujący go serwis?
- Rejestry serwisów
  - UDDI - Universal Description, Discovery, and Integration
  - WS-Inspection (IBM, Microsoft)

UDDI

- Rejestr Web Serwisów
- Pozwala firmom umieszczać w rejestrze informacje o rodzaju działalności i dostępnych Web Serwisach
- Klient może przeglądać rejestr wyszukując interesujące go serwisy (usługi) na podstawie typu usługi i rodzaju działalności.

Korzystanie z UDDI

- Typowy scenariusz korzystania z WS za pośrednictwem UDDI:
  - Lokalizacja serwisu w rejestrze serwisów UDDI
  - Pobranie definicji serwisu WSDL
  - Wywołanie odpowiedniej funkcji serwisu

Web Serwisy i PHP

- W PHP dostępnych jest kilka możliwości tworzenia Web Serwisów, m.in..
  - Standardowe rozszerzenie SOAP PHP
  - Rozszerzenia będące częścią frameworków (np. Zend SOAP)
  - Zewnętrzne biblioteki np. NuSOAP

PHP SOAP

- Dwie podstawowe klasy:
  - **SoapClient** – implementuje klienta SOAP
  - **Soapserver** – implementuje serwer SOAP
- Obsługują standardy SOAP 1.1 i SOAP 1.2
- Mogą używać opisu WSDL

PHP SOAP - serwer

```php
<?php

function response() {
   return "Hello!";
}

$server = new SoapServer(null,
        array('uri' => "urn:example_SOAP_response"));
$server->addFunction("response");
$server->handle();

?>
```

SoapServer

- Klasa SoapServer implementuje serwer SOAP
- Może być uruchomiony w trybie WSDL (pierwszy argument – URI do definicji) lub bez WSDL (pierwszy argument null, URI przekazane w opcjach).
- Dodatkowe opcje konstruktora kontrolują zachowanie serwera.

SoapServer – c.d.

- **addFunction** – dodaje jedną lub wiele funkcji do serwisu (wiąże serwis z funkcjami)
- **setClass** – dodaje klasę do serwisu (udostępniane są wszystkie publiczne metody klasy)
- **handle** – obsługuje otrzymany komunikat SOAP

PHP SOAP - klient

```
<?php

    $client = new SoapClient(null, array(
                'location' =>
                 "http://localhost/~jankazim/ex07.php",
                 'uri'      => "urn:example_SOAP_response",
                 'trace'    => 1 ));

    $return = $client->__soapCall("response",array());
     echo "\nReturning value of __soapCall() call: ".
     $return."\n\n";

?>
```

SoapClient

- Klasa SoapClient implementuje klienta SOAP
- Klient może wykorzystywać plik opisu WSDL (pierwszy argument – URI do opisu), lub działać w trybie bez WSDL (informacje o serwisie w opcjach)
- Dodatkowe opcje kontrolują zachowanie klienta, sposób przekazywania danych, śledzenie błędów itp.

SoapClient c.d.

- Funkcja __soapCall – pozwala wywołać określoną funkcję Web Serwisu.
- W przypadku trybu WSDL funkcja __soapCall nie jest potrzebna, funkcje Web Serwisu mapowane są jako metody obiektu klasy SoapClient

Zend SOAP

- Zend dostarcza klasy pozwalające na tworzenie Web Serwisów:
  - **Zend_Soap_Server** – serwer SOAP
  - **Zend_Soap_Client** – klient SOAP
- Dodatkowo Zend dostarcza jeszcze klasy:
  - **Zend_Soap_Wsdl** – pozwala na ręczne budowanie dokumentu WSDL
  - **Zend_Soap_AutoDiscover** – pozwala na automatyczne generowanie WSDL na podstawie kodu i dokumentacji klasy

Przykład WS - Magento

- Magento – popularna platforma e-commerce (http://www.magentocommerce.com/)
- Elastyczna, oparta o framework Zend
- Dostępne liczne moduły (darmowe i płatne)
- API SOAP umożliwia dostęp do większości podstawowych funkcji

Magento SOAP API

- Opis usług dostępny w postaci pliku WSDL
  http://magentohost/api/?wsdl lub
  http://magentohost/api/soap/?wsdl
- Korzystanie z API wymaga autentykacji (użytkownik API utworzony w panelu Magento)
- Dostęp do funkcji API można uzyskać za pomocą metody call.

Magento API – lista produktów

```
$client = new SoapClient('http://magentohost/api/soap/?wsdl');

// If somestuff requires api authentification,
// then get a session token
$session = $client->login('apiUser', 'apiKey');

$result = $client->call($session, 'catalog_product.list');
var_dump($result);

// If you don't need the session anymore
$client->endSession($session);
```

Magento API – dane produktu

```
$client = new SoapClient('http://magentohost/api/soap/?wsdl');

// If somestuff requires api authentification,
// then get a session token
$session = $client->login('apiUser', 'apiKey');

$result = $client->call($session, 'catalog_product.info', '4');
var_dump($result);

// If you don't need the session anymore
$client->endSession($session);
```

Magento API – Dodaj produkt

```
$client = new SoapClient('http://magentohost/api/soap/?wsdl');
$session = $client->login('apiUser', 'apiKey');

// get attribute set
$attributeSets = $client->call($session,'product_attribute_set.list');
$attributeSet = current($attributeSets);

$result = $client->call($session, 'catalog_product.create', array('simple',
$attributeSet['set_id'], 'product_sku', array(
    'categories' => array(2),
    'websites' => array(1),
    'name' => 'Product name',
    'description' => 'Product description',
    'short_description' => 'Product short description',
    ...
)));

var_dump ($result);
```

Przykład WS – serwis DPD

- Firma kurierska DPD udostępnia swoje usługi m.in. za pomocą Web Serwisu SOAP
  - Walidacja danych paczek i nadawanie numerów listów przewozowych
  - Generowanie etykiet listów przewozowych
  - Generowanie protokołów odbioru
  - Zamawianie kuriera

DPD – Walidacja danych paczek

- Dane wejściowe:
  - Lista przesyłek
  - Sposób obsługi błędów
  - Dane autoryzacyjne
- Dane zwracane
  - Sesja z listą przesyłek i paczek, identyfikator sesji

DPD – Walidacja danych paczek

- Sygnatura metody

```
public PackagesGenerationResponseV1
    generatePackagesNumbersV1(
                        OpenUMLFV1 openUMLV1,
                        PkgNumsGenerationPolicyV1 policyV1,
                        AuthDataV1 authDataV1)
```

- Dane wejściowe są obiektami reprezentującymi dane o przesyłkach, dane autoryzacyjne, sposób obsługi błędów.

Fragment zapytania SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <ns2:generatePackagesNumbersV1 xmlns:ns2="http://dpdservices.dpd.com.pl/">
         <openUMLV1>
            <packages>
               <parcels>
                  <content>Content</content>
                  <customerData1>CustomerData1</customerData1>
                  <customerData2>CustomerData2</customerData2>
                  <customerData3>CustomerData3</customerData3>
                  <sizeX>1</sizeX>
                  <sizeY>2</sizeY>
                  <sizeZ>3</sizeZ>
                  <weight>1.2</weight>
               </parcels>
               <parcels>
                  <content>Content</content>
                  <customerData1>CustomerData1</customerData1>
                  <customerData2>CustomerData2</customerData2>
                  <customerData3>CustomerData3</customerData3>
                  <sizeX>1</sizeX>
                  <sizeY>2</sizeY>
                  <sizeZ>3</sizeZ>
                  <weight>1.2</weight>
               </parcels>
```

Fragment odpowiedzi SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:generatePackagesNumbersV1Response xmlns:ns2="http://dpdservices.dpd.com.pl/">
         <return>
            <packages>
               <packageId>2562257</packageId>
               <parcels>
                  <parcelId>3464000</parcelId>
                  <status>OK</status>
                  <waybill>0001531901495A</waybill>
               </parcels>
               <parcels>
                  <parcelId>3464001</parcelId>
                  <status>OK</status>
                  <waybill>0002531901495A</waybill>
               </parcels>
               <status>OK</status>
            </packages>
```