

Wprowadzenie do technologii Web Services: SOAP, WSDL i UDDI

Maciej Zakrzewicz
PLOUG

mzakrz@cs.put.poznan.pl

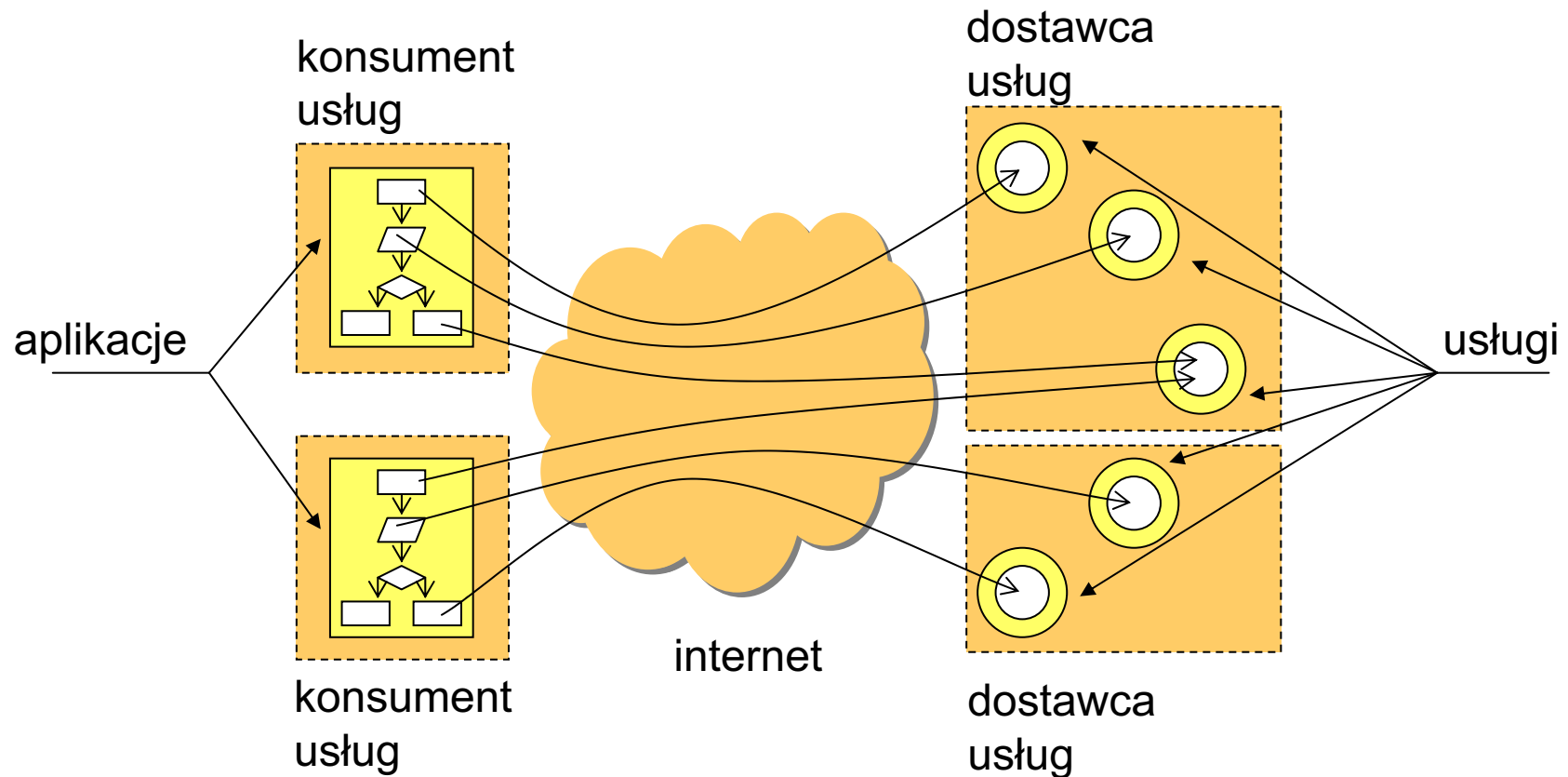
Plan prezentacji

- Wprowadzenie do architektury zorientowanej na usługi
- Charakterystyka technologii Web Services
 - protokół komunikacyjny SOAP
 - języku opisu interfejsu WSDL
 - rejestr usług UDDI
- Konstrukcja aplikacji Web Services za pomocą narzędzia Oracle JDeveloper

Architektura zorientowana na usługi (SOA)

- Logika biznesowa nie stanowi monolitycznego programu; rozbita pomiędzy wiele rozproszonych komponentów usługowych
- Komponenty usługowe koordynowane przez centralną aplikację sterującą
- Nowy model biznesu IT, bazujący na odpłatnym oferowaniu dostępu do biznesowych komponentów usługowych

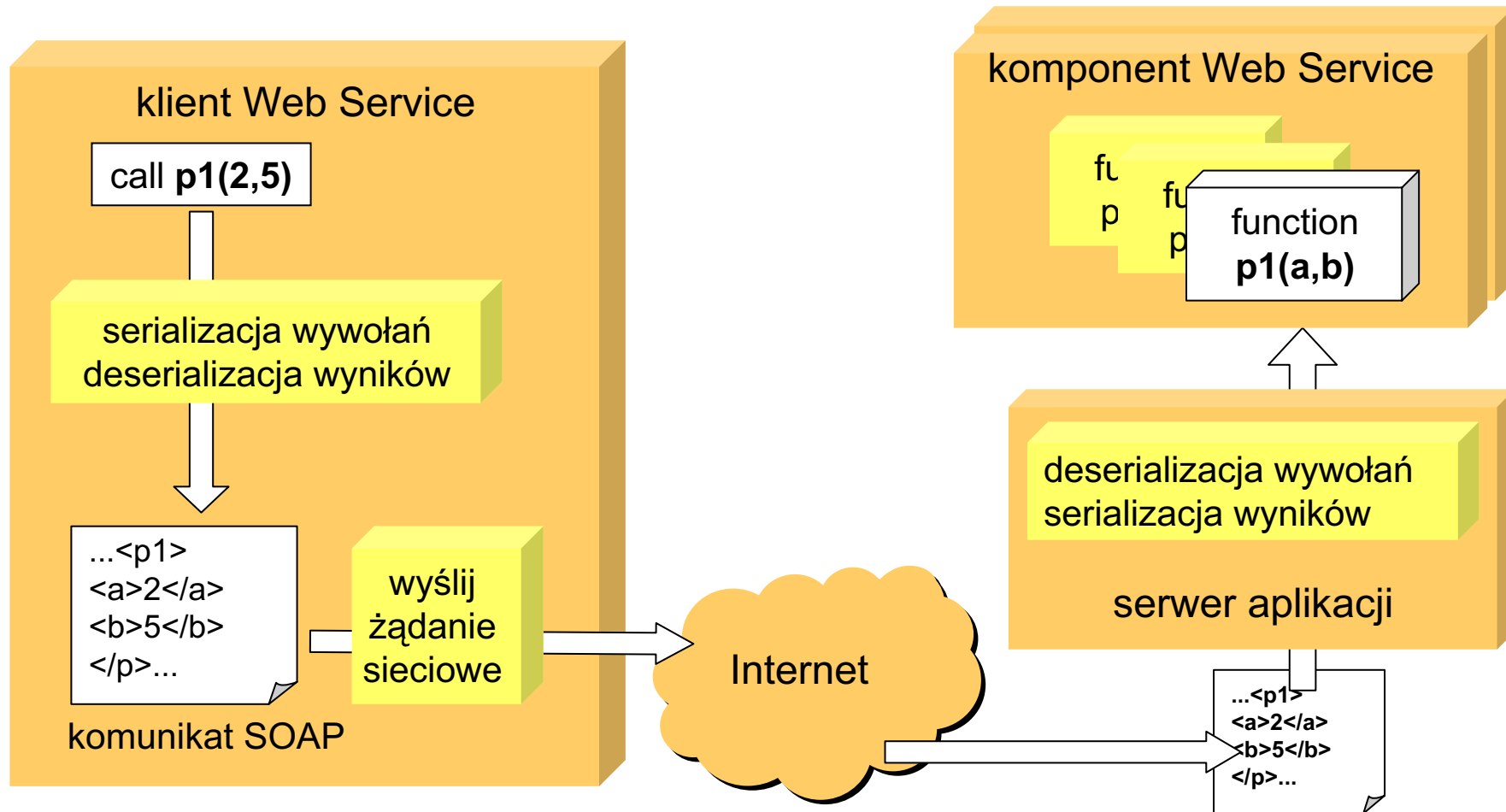
Architektura zorientowana na usługi (SOA)



Technologia Web Services

- Technologia konstrukcji rozproszonych komponentów usługowych, stanowiących podstawę dla realizacji aplikacji biznesowych w architekturze SOA
- Usługa Web Service: zwarty, samodokumentujący się komponent programowy, który może być przez swojego twórcę zarejestrowany w sieci komputerowej, a następnie przez twórcę aplikacji-konsumenta odkryty i wywołany w trybie zdalnego wykonania

Technologia Web Services



Protokół SOAP

- Simple Object Access Protocol
- Prosty protokół komunikacyjny oparty na języku XML, umożliwiający przekazywanie wywołań zdalnych komponentów Web Services
- Może współdziałać z dowolnym niskopoziomowym sieciowym mechanizmem transportowym, np. HTTP, HTTPS, SMTP, JMS, RMI
- Dwa tryby wywołań: Remote Procedure Call i dokumentowy (document-oriented)

Protokół SOAP

```
<?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="demo">
    <m:multiply>
      <m:val1>3</m:val1>
      <m:val2>2</m:val2>
    </m:multiply>
  </soap:Body>
</soap:Envelope>
```

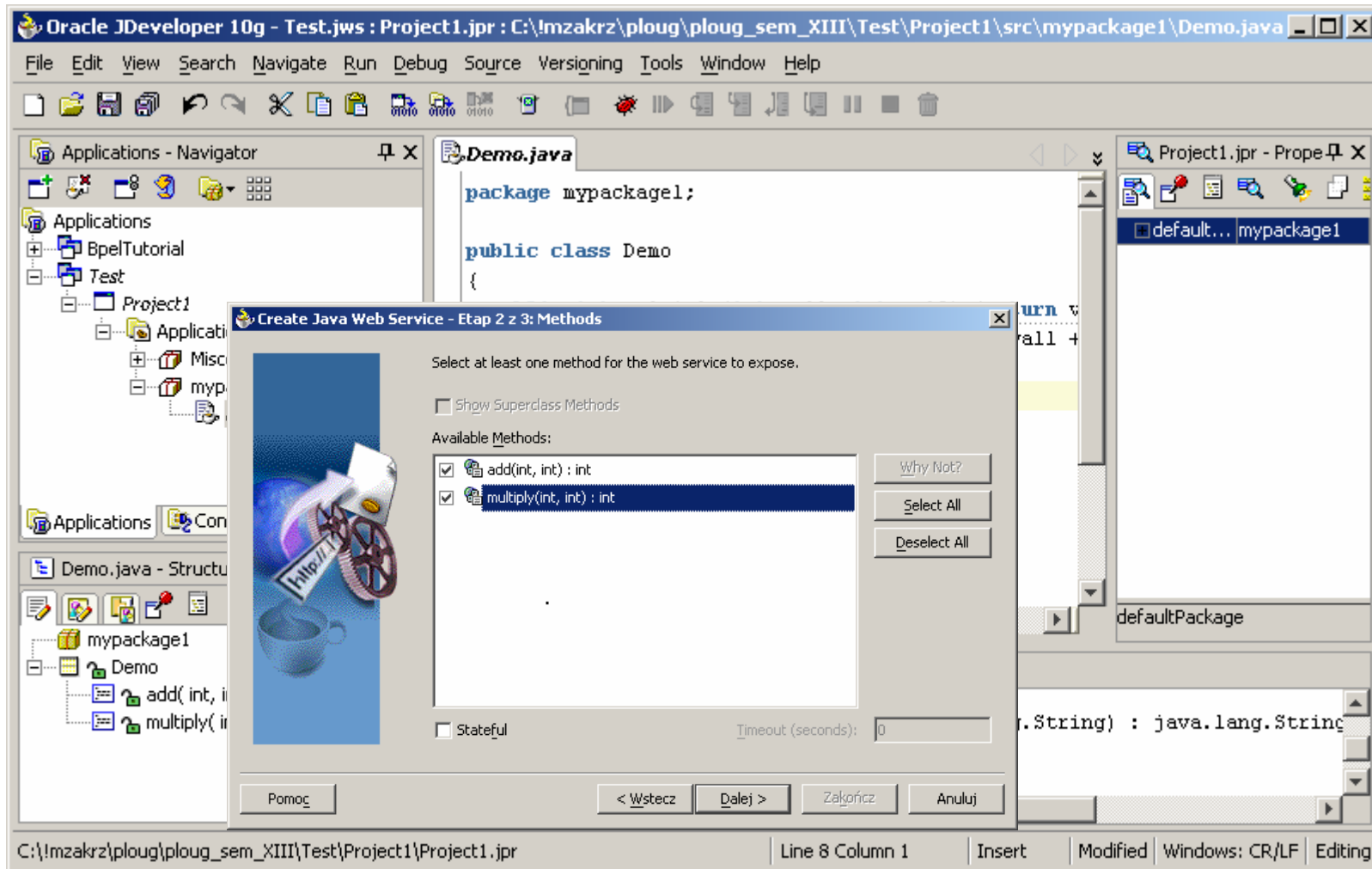
Żądanie:
multiply(2,3)

Odpowiedź:
6

```
<m:multiplyResponse>
  <m:result>6</m:result>
</m:multiplyResponse>
</soap:Body>
</soap:Envelope>
```

```
.../12/soap-envelope"
...ap-encoding">
```

Implementacja usługi Web Services



1. Utwórz klasę Java zawierającą metody logiki biznesowej
2. Skorzystaj z kreatora Java web Service

Instalowanie usługi Web Services

The screenshot shows the Oracle JDeveloper 10g IDE. The main editor displays the content of 'IDemo.wSDL', which is an XML file defining a web service. The XML content includes a namespace definition for 'Demo' and a target namespace for 'http://mypackagel/Demo.wSDL'. The 'Applications - Navigator' on the left shows a project structure with 'WebServices.deploy' selected. A context menu is open over 'WebServices.deploy', with 'Deploy to' selected, and a sub-menu showing 'LocalOC4J' as the target. The 'Property Inspector' on the right is empty.

```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<!--Generated by the Oracle JDeveloper 10g Web Serv
<!--Date Created: Tue Apr 25 14:56:17 CEST 2006-->
<definitions
  name="Demo"
  targetNamespace="http://mypackagel/Demo.wSDL"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap"
  xmlns:tns="http://mypackagel/Demo.wSDL"
  http://mypackagel/IDemo.xsd">
```

WebServices.deploy - Structure

Type: WAR File

WAR: C:\!mzakrz\ploug\ploug_se

ESAP: C:\!mzakrz\ploug\ploug_se

Deploy to LocalOC4J

Deploy to

Deploy to WAR file

Deploy to EAR file

Compare With

Properties...

LocalOC4J

New Connection...

SOAP-ENC="http://schemas.xmlsoap.org

Testowanie zainstalowanej usługi

Demo endpoint

For a formal definition, please review the S

Demo service

The following operations are supported.

- [add](#)
- [multiply](#)

oc4j client

The java proxy is packaged in a .jar either

- [Proxy Jar](#)
- [Proxy Source](#)

Demo Web Service - Microsoft Internet Explorer

Click [here](#) for a complete list of operations.

add

Test

To test the operation using the HTTP GET protocol, click the 'Invoke' button.

Parameter	Type	Value
val1	int	<input type="text" value="2"/>
val2	int	<input type="text" value="3"/>

Implementacja klienta Web Services

```
import org.apache.soap.*;
import oracle.soap.transport.http.*;
...
Call call = new Call();
call.setTargetObjectURI("demo");
call.setMethodName("multiply");
call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
Vector params = new Vector();
params.addElement(new Parameter("val1", Integer.class, 2, null));
params.addElement(new Parameter("val2", Integer.class, 3, null));
call.setParams(params);
Response resp = call.invoke(new URL("http://miner/demo"), "");
Parameter ret = resp.getReturnValue();
Object value = ret.getValue();
System.out.println(value);
```

Dokument WSDL

```
<definitions name="demo" ...>
  <message name="multiply0Request">
    <part name="val1" type="xsd:int"/>
    <part name="val2" type="xsd:int"/>
  </message>
  <message name="multiply0Response">
    <part name="return" type="xsd:int"/>
  </message> ...
  <binding name="DemoBinding" type="tns:DemoPortType">
    <soap:binding style="rpc" ...>
    <operation name="multiply">
      <input name="multiply0Request"> ... </input>
      <output name="multiply0Response"> ... </output>
    </operation>
  </binding>
  <service name="http://miner/Demo">
    <port name="DemoPort" binding="tns:DemoBinding">
      <soap:address location="http://miner/Demo"/>
    </port>
  </service>
</definitions>
```

WSDL to język znaczników XML służący do opisu technicznych parametrów połączenia sieciowego aplikacji-klienta z komponentem Web Service

Generowanie klasy pośrednika (Stub/Proxy)

The screenshot displays the Oracle JDeveloper 10g IDE interface. The main editor window shows the WSDL file `IIsbnFinder.wsdl` with the following content:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<!--Generated by the Oracle JDeveloper 10g Web Services WSDL Generator-->
<!--Date Created: Sun Apr 23 21:06:38 CEST 2006-->
<definitions
  name="IsbnFinder"
  targetNamespace="http://mypackage2/IsbnFinder.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/"
  xmlns:tns="http://mypackage2/IsbnFinder.wsdl"
  xmlns:finder="http://schemas.xmlsoap.org/finder.wsdl"
  xmlns:finderxsd="http://schemas.xmlsoap.org/finder.xsd">
  <part name="param1" type="xsd:string"/>
</message>
<message name="getIsbn0Response">
  <part name="return" type="xsd:string"/>
</message>
</definitions>
```

A context menu is open over the WSDL file, with the following options:

- Generate Web Service Stub/Skeleton...
- Open
- Check XML Syntax
- Validate WSDL
- Make (Ctrl+Shift+F9)
- Compare With
- WS-I Analyze WSDL...
- Create Data Control

The Applications - Navigator on the left shows the project structure:

- Project1
 - Application Sources
 - Miscellaneous Packages
 - IIsbnFinder.wsdl

A zoomed-in view of the project structure and the context menu is shown below:

- Project1
 - Application Sources
 - Miscellaneous Packages
 - IIsbnFinder.wsdl

The context menu options are:

- Generate Web Service Stub/Skeleton...
- Open

The status bar at the bottom indicates "Saved nodes(1)" and "Editing".

Klasa-pośrednik (Stub/Proxy)

```
public class DemoStub
{
    public DemoStub() {...}
    public String getEndpoint() {...}
    public void setEndpoint(String endpoint) {...}
    public Integer add(Integer val1, Integer val2) throws Exception {...}
    public Integer multiply(Integer val1, Integer val2) throws Exception {...}
    public void setMaintainSession(boolean maintainSession) {...}
    public boolean getMaintainSession() {...}
    public void setTransportProperties(Properties props) {...}
    public Properties getTransportProperties() {...}
}

public class DemoClient
{
    public static void main(String[] args) throws Exception
    {
        DemoStub ds = new DemoStub();
        System.out.println(ds.multiply(new Integer(2), new Integer(3)));
    }
}
```

```
public class Demo
{
    public int multiply(int val1, int val2) {return val1 * val2;}
    public int add(int val1, int val2) {return val1 + val2;}
}
```

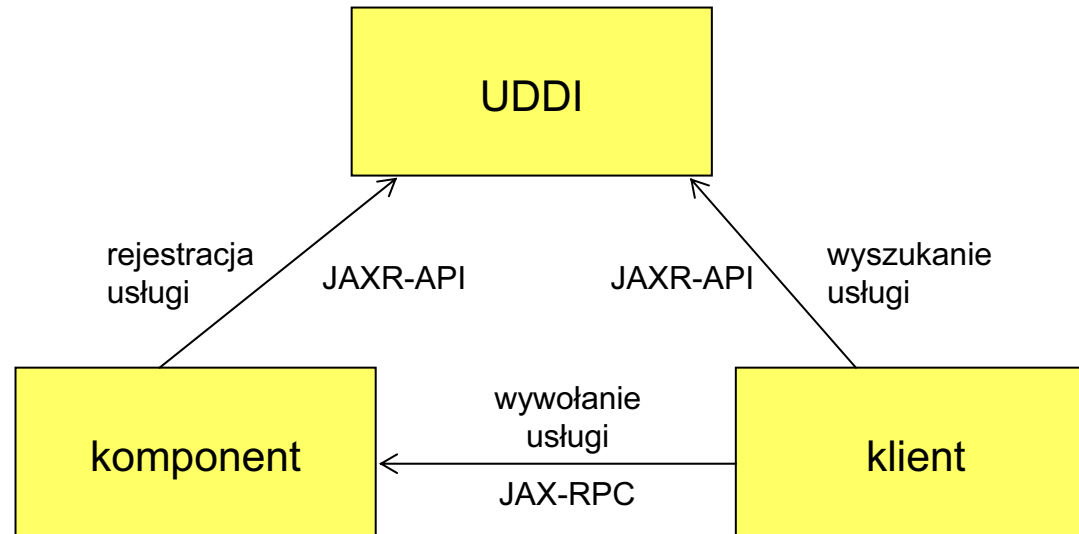
Dynamiczne generowanie klasy- pośrednika (Stub/Proxy)

```
WebServiceProxyFactory factory = new WebServiceProxyFactory();  
WebServiceProxy proxy =  
    factory.createWebServiceProxy("http:...demo.wsdl");  
WebServiceMethod method = proxy.getMethod("multiply");  
String paramNames[] = { "val1", "val2"};  
Object paramValues[] = new Integer[2];  
paramValues[0] = new Integer(2);  
paramValues[1] = new Integer(3);  
Object response = method.invoke(paramNames,paramValues);  
...
```

Universal Description, Discovery, and Integration - UDDI

- Specyfikacja bazy danych, w której twórcy usług Web Services dokonują ich rejestracji (przy użyciu WSDL)
- Twórcy aplikacji klienta mogą przeszukiwać bazy danych UDDI w celu znalezienia potrzebnej im usługi sieciowej; przeszukiwanie jest oparte o nazwy, adresy, klasyfikacje NAICS, ISO-3166, UNSPSC
- Pobrany z bazy danych UDDI dokument WSDL służy twórcom aplikacji klienta do wygenerowania Client Proxy, używanego następnie podczas budowy aplikacji końcowej
- Przykładowe bazy danych UDDI - <http://uddi.microsoft.com>, <http://uddi.sap.com>, <http://uddi.ibm.com>
- Udostępniane usługi obejmują prognozy pogody, wyszukiwarki, kursy akcji, kursy walut, oferty sklepów internetowych, itd.

Cykl życia usługi Web Service



Zestaw programisty WSDP

- Java API for XML Processing (JAXP): podstawowe biblioteki przetwarzania danych XML, zgodne z rekomendacją W3C,
- Java API for XML Messaging (JAXM): biblioteka sieciowej komunikacji opartej na języku XML,
- SOAP with Attachments API for Java (SAAJ): biblioteka sieciowej komunikacji w oparciu o protokół SOAP,
- Java API for XML-based RPC (JAX-RPC): biblioteka służąca do realizacji zdalnych wywołań komponentów usługowych w trybie RPC,
- Java API for XML Registries (JAXR): biblioteka służąca do dostępu do XML-owych rejestrów usług Web Services np. UDDI,
- Tomcat – zredukowany serwer J2EE umożliwiający instalowanie i uruchamianie komponentów usługowych Web Services,
- JavaServer Pages Standard Tag Library (JSTL): standardowa biblioteka znaczników JSP,
- Registry Server: rejestr UDDI

Podsumowanie

- Podstawowe kanony implementacji środowisk Web Services:
 - implementację komponentu usługowego w dowolnym języku programowania
 - sporządzenie pliku opisu interfejsu komponentu (WSDL)
 - zgłoszenie komponentu do centralnego rejestru (UDDI)
 - wyszukanie komponentu w centralnym rejestrze (UDDI)
 - wygenerowanie kodu komunikacyjnego dla klienta (Web Service Stub)
 - implementację klienta Web Service