

ARCHITEKTURA ZORIENTOWANA NA USŁUGI

SOA: SERVICE ORIENTED ARCHITECTURE

SOAP: SIMPLE OBJECT ACCESS PROTOCOL

WSDL: WEB SERVICES DESCRIPTION LANGUAGE

2

SOA: Service Oriented Architecture

Architektura zorientowana na usługi

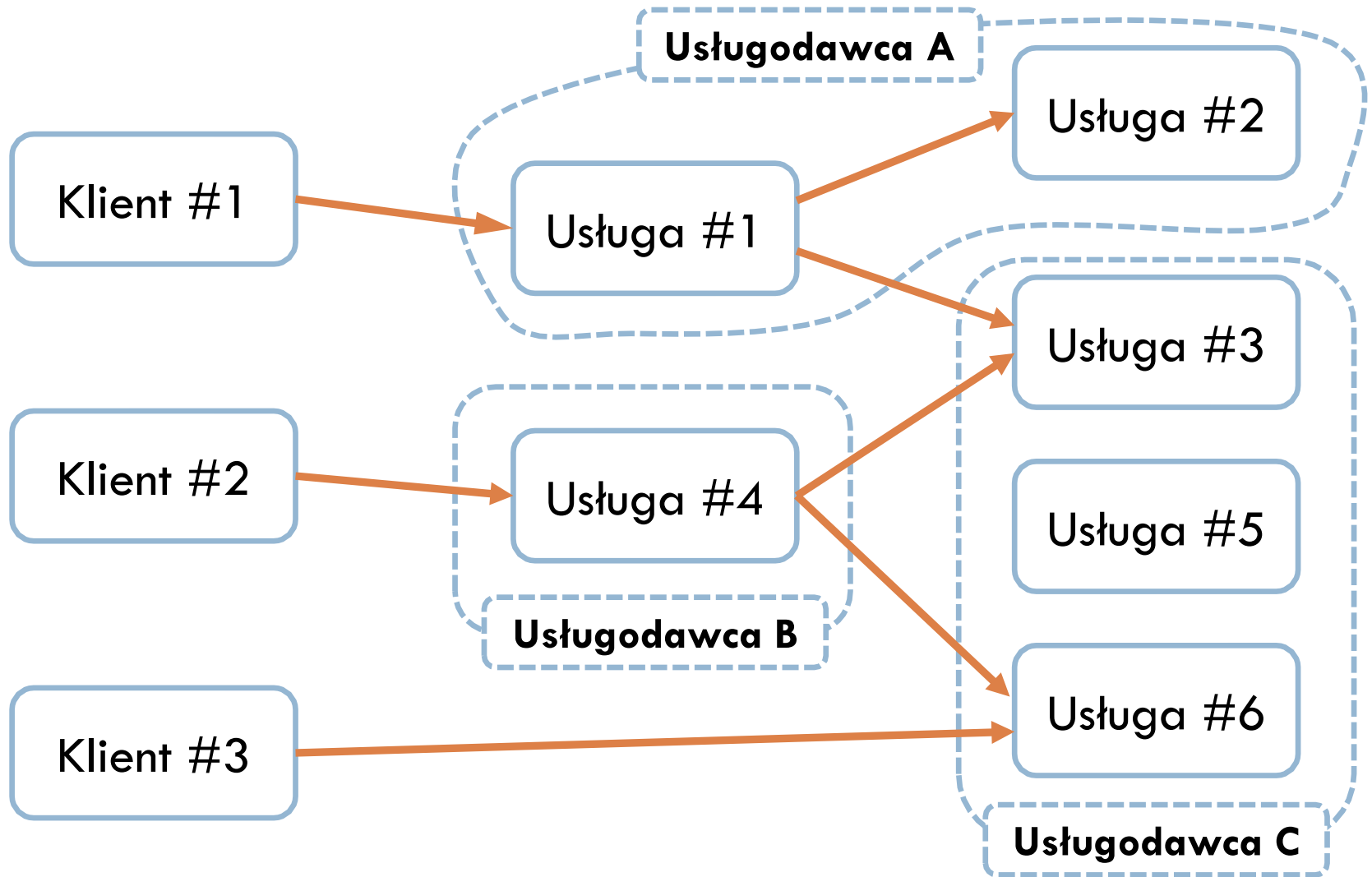
Service Oriented Architecture (SOA)

3

- Usługodawcy
 - ▣ Usługa prognozy pogody
 - ▣ Usługa płatności (np. kartą kredytową)
 - ▣ Usługa rozsyłania wiadomości SMS
- Klienci – aplikacje klienckie
 - ▣ Sklep internetowy
 - wykorzystuje płatności
 - ▣ Aplikacja na urządzenie mobilne (np. smartphone)
 - wykorzystuje serwis pogodowy
 - ▣ Aplikacja okienkowa w przychodni
 - wykorzystuje usługę SMSową (np. przypomnienia o wizytach)
 - ▣ Komponent w środowisku rozproszonym

Service Oriented Architecture (SOA)

4



Service Oriented Architecture (SOA)

5

- Klient nie musi znać szczegółów działania usługi – ani implementacyjnych ani biznesowych/branżowych
 - ▣ Usługa pogodowa
 - Szczegóły branżowe: skąd pochodzą dane meteorologiczne (stacje, czujniki), jaki model prognozowania został zastosowany
 - Szczegóły implementacyjne: jak dane są składowane
 - Klient chce tylko wiedzieć, czy będzie padać!
 - ▣ Usługa płatności
 - Szczegóły biznesowe: umowy z bankami, sesje rozliczeniowe
 - Szczegóły implementacyjne: komunikacja z systemem bankowym
 - Klient – np. sklep internetowy – chce tylko wiedzieć, czy zamówienie zostało opłacone!

Service Oriented Architecture (SOA)

6

- Architektura SOA pozwala klientom usług skupić się na własnych działaniach biznesowych
- Operacje niezwiązane bezpośrednio z biznesem klienta zostają oddelegowane do usługodawców, którzy się w nich specjalizują
 - ▣ Forma *outsourcingu*
- Usługa może wywoływać inne usługi (orkiestracja) w celu zaspokojenie potrzeb klienta
 - ▣ Usługa staje się klientem innych usług, które mogą pochodzić od innego usługodawcy

Manifest SOA (ang. *SOA Manifesto*)

7

- Business value over technical strategy
- Strategic goals over project-specific benefits
- Intrinsic interoperability over custom integration
- Shared services over specific-purpose implementations
- Flexibility over optimization
- Evolutionary refinement over pursuit of initial perfection

October 2009

Architektura zorientowana na usługi

8

- Usługodawca i klient to często dwa całkowicie odrębne podmioty
 - ▣ Odrębne aplikacje, odrębne firmy
 - ▣ Różne języki implementacji i platformy technologiczne
 - Java, C#/.NET, PHP, JavaScript, Objective-C, Swift
 - ▣ Różne środowiska działania aplikacji
 - Windows, Linux, OS X, Android, iOS
 - 32-bitowe/64-bitowe, LittleEndian/BigEndian
- **Heterogeniczne środowisko**
 - ▣ Różnorodne komponenty wchodzące w interakcje

Komunikacja

Jak połączyć ze sobą heterogeniczne komponenty?

Gniazda sieciowe (ang. *sockets*)

10

- Wydajność (+)
- Konieczność zdefiniowania własnego protokołu komunikacyjnego między klientem i usługodawcą (-)
 - ▣ Operacje (komendy), dane i ich format
 - Bajty w pamięci: 00 | 2A
Java: 0x002A (42, BigEndian) → .NET 0x2A00 (10752, LittleEndian)
 - ▣ Każdy usługodawca definiuje własny protokół...
- Brak wsparcia narzędziowego (-)
 - ▣ Źmudne wytwarzanie usług
 - ▣ Kłopotliwa konsumpcja usług
- Trudne wersjonowanie API (-)

Architektura zorientowana na usługi

11

**W architekturze SOA potrzeba standardów,
nie własnościowych rozwiązań!**

Komunikacja – wymagania

12

- Jednolity, standardowy i rozszerzalny protokół
 - ▣ Wspólny dla wielu dostawców – łatwiejsze konsumowanie
 - ▣ Definiujący strukturę operacji i sposób ich wywoływania
 - Umożliwiający definiowanie własnych operacji w ramach tej struktury
 - ▣ Definiujący formaty danych i sposób ich kodowania
 - *Endianness*, formaty liczb całkowitych, zmiennoprzecinkowych, dat, struktur
 - Umożliwiający definiowanie własnych złożonych, hierarchicznych struktur danych na bazie składowych typów prostych
- Wsparcie narzędziowe
 - ▣ Łatwość wytwarzania i konsumpcji usług

SMTP

13

- Określa standardowe sposoby kodowania danych (+)
 - ▣ Nagłówki, typy MIME, BASE64
 - ▣ Brak hierarchicznych struktur danych (-)
- Brak definicji operacji na poziomie protokołu (-)
 - ▣ Tylko przesyłanie komunikatów
- Jednokierunkowe komunikaty nadawca → odbiorca (-)
 - ▣ Przeznaczony do obsługi poczty...
- Potrzebny model komunikacji: żądanie → odpowiedź
 - ▣ Lepiej dopasowany do relacji klient-usługodawca

HTTP

14

- Definiuje standardowe operacje (+)
 - ▣ GET, POST, PUT, DELETE
 - ▣ Brak możliwości definiowania własnych (-)
- Definiuje standardowe kody odpowiedzi (+)
 - ▣ 200, 303, 401, 500
 - ▣ Brak możliwości definiowania własnych... (-)
- Definiuje sposoby kodowania danych (+)
 - ▣ Typy MIME, tekstowe reprezentacje: *url-encoded*, BASE64
 - ▣ Brak zdefiniowanych formatów np. dla dat, struktur złożonych
 - Konieczne wprowadzanie zewnętrznych formatów, np. JSON, XML
- Działa w kontekście zasobów nie operacji
 - ▣ Konieczne mapowanie modelu RPC na model zasobów
- Wykorzystywany jako protokół dla usług REST
 - ▣ Szersze omówienie na dalszych wykładach

CORBA

Common Object Request Broker Architecture

15

- Model RPC (*Remote Procedure Call*)
- Niezależny od platformy, przenośny (+)
 - ▣ CORBA 1.x – tylko na poziomie IDL
 - ▣ CORBA 2.x (1997) – również *on-the-wire*
- IDL – Interface Definition Language (+)
- Ugruntowana pozycja na rynku (od 1991 roku) (+)
 - ▣ Przestarzały... (-)
- Nadmiernie rozbudowany standard, niespójny (-)
- Problemy na poziomie zapór *firewall* (-)
 - ▣ Brak standardowych numerów portów usług

SOAP Web Services

16

- Standard mający spełnić wymagania architektury zorientowanej na usługi (SOA)
 - ▣ ...i rozwiązać problemy standardu CORBA
- Pierwsza wersja – rok 2000
- Wsparcie ze strony przodujących firm w branży, m.in.: IBM, Microsoft, Sun, SAP
- Projektowany z myślą o automatyzacji i wsparciu narzędziowym
- Wykorzystuje m.in. HTTP jako protokół transportowy
 - ▣ Co jest takiego szczególnego w protokole HTTP?

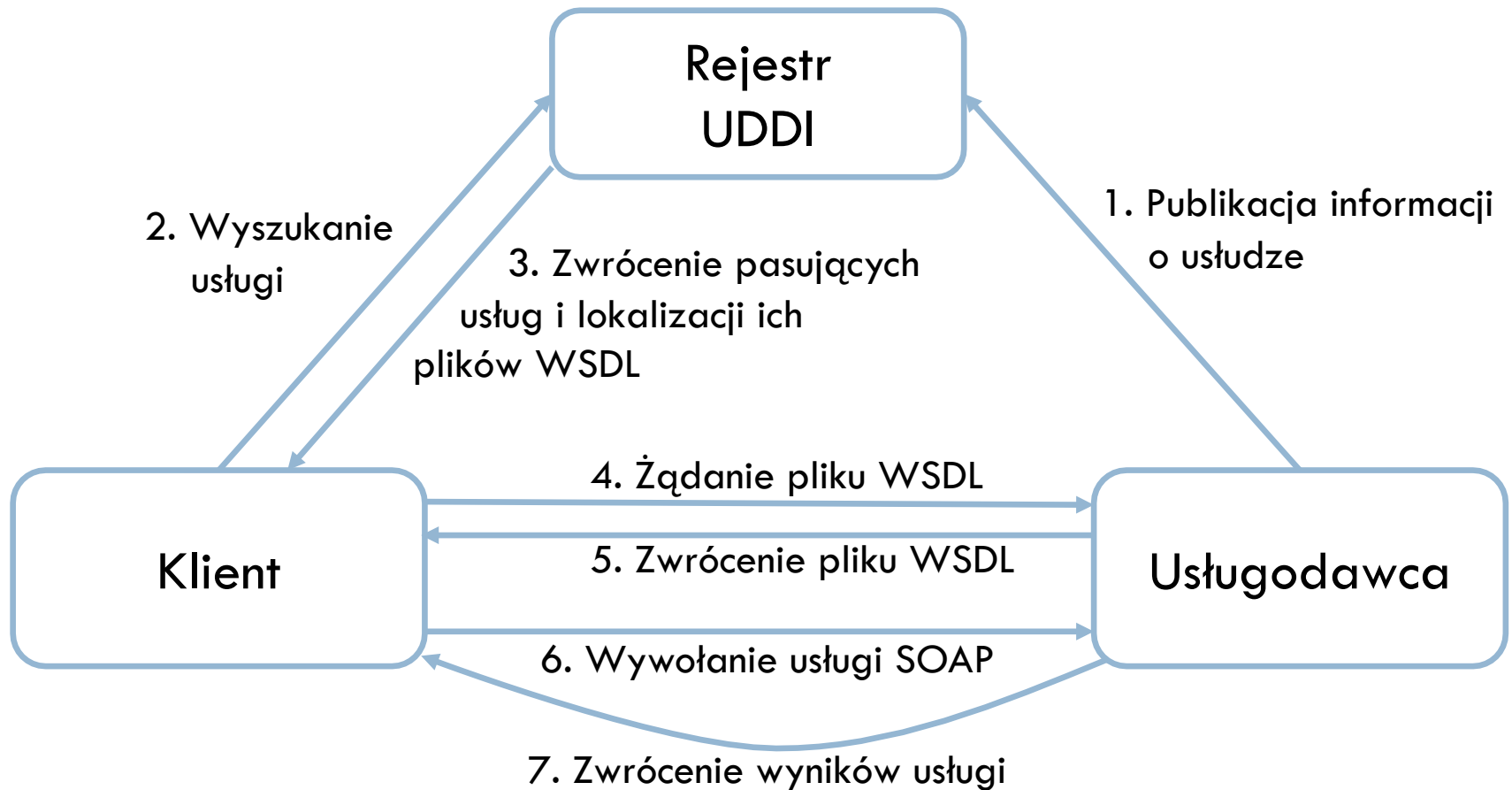
SOAP

17

- ❑ Publikowanie luźno powiązanych usług z różnych lokalizacji, przy użyciu różnorodnych platform i języków programowania
- ❑ Opublikowane usługi opisane w rejestrze UDDI, wraz z informacją o usługodawcach
- ❑ Szczegóły usługi opisane w pliku WSDL
 - ❑ Operacje, argumenty, typy danych (również złożone), lokalizacje usług, protokoły
- ❑ Dynamiczne usługi – mogą pojawiać się i być wyłączane w dowolnym czasie
- ❑ Mogą być łączone w skomplikowane procesy
 - ❑ Jedna usługa wywołuje kolejne (orkiestracja usług)

Usługi internetowe SOAP

18



Rys. 1. Oryginalna idea przebiegu interakcji pomiędzy klientami, rejestrem dostępnych usług i usługodawcami.

SOAP

19

- Protokół wymiany wiadomości pomiędzy klientem i usługodawcą przez możliwe węzły pośrednie
- Wykorzystuje inne protokoły jako warstwę transportową, m.in.:
 - ▣ HTTP – najpopularniejszy transport
 - ▣ SMTP
 - ▣ MSMQ
 - ▣ Gniazda TCP/IP

SOAP

20

- Wykorzystuje XML do opisu przesyłanych wiadomości
 - ▣ Interoperacyjność – biblioteki do obsługi formatu XML w każdym znaczącym języku programowania
 - ▣ Narzut na przetwarzanie w stosunku do protokołów binarnych (np. CORBA)
- Projektowany jako protokół niezależny od języka, platformy, systemu operacyjnego
- Istnieje wiele różnych implementacji
 - ▣ .NET – WCF, Java – JAX-WS, AXIS, PHP, Python...

SOAP – dwa modele komunikacji

21

- RPC (*Remote Procedure Call*)
 - ▣ Wywołania zdalnych procedur
- Document – *Conversational Message Exchanges*
 - ▣ Przekazane dane traktowane jako dokument do przetworzenia
- Wybrany model określany w pliku WSDL

WSDL

22

- Web Services Description Language
- Opisuje strukturę usługi internetowej
 - ▣ Interfejs
 - Metody
 - Parametry
 - Typy danych
 - ▣ Protokoły używane w wywołaniach
 - ▣ Lokalizacja usługi (np. URL)
- Stanowi opis *kontraktu* pomiędzy klientem i usługą
- Specyfikacja: <http://www.w3.org/TR/wsdl>

WSDL – kompletny plik

23

```
<?xml version="1.0"?>
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
```

```
<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>

<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>

</definitions>
```

WSDL – przestrzenie nazw

24

```
<?xml version="1.0"?>  
<definitions name="StockQuote"  
    targetNamespace="http://example.com/stockquote.wsdl"  
    xmlns:tns="http://example.com/stockquote.wsdl"  
    xmlns:xsd1="http://example.com/stockquote.xsd"  
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  
    xmlns="http://schemas.xmlsoap.org/wsdl/">
```


WSDL – typy danych (*types*)

25

```
<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>
```

WSDL – wiadomości (*message*)

26

```
<message name="GetLastTradePriceInput">  
  <part name="body" element="xsd1:TradePriceRequest"/>  
</message>
```

```
<message name="GetLastTradePriceOutput">  
  <part name="body" element="xsd1:TradePrice"/>  
</message>
```

Definicja usługi jako zbioru operacji (*portType*)

27

```
<portType name="StockQuotePortType">  
  <operation name="GetLastTradePrice">  
    <input message="tns:GetLastTradePriceInput"/>  
    <output message="tns:GetLastTradePriceOutput"/>  
  </operation>  
</portType>
```

Wiązanie usługi z protokołem transportowym (*binding*)


28

```
<binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation
      soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

WSDL – lokalizacja usługi

29

```
<service name="StockQuoteService">  
  <documentation>My first service</documentation>  
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">  
    <soap:address location="http://example.com/stockquote"/>  
  </port>  
</service>  
  
</definitions>
```



adres usługi

WSDL – wytwarzanie opisu usługi

30

□ *Contract first*

- Zaczynamy od przygotowania pliku WSDL
- Na podstawie WSDL generujemy szkielet kodu usługi do wypełnienia
- Ułatwia uzyskanie interoperacyjności

□ *Code first*

- Zaczynamy od implementacji usługi
- W oparciu o istniejącą implementację generowany jest plik WSDL
- Szybsze wytworzenie usługi
- Zwalnia z konieczności znajomości szczegółów formatu WSDL

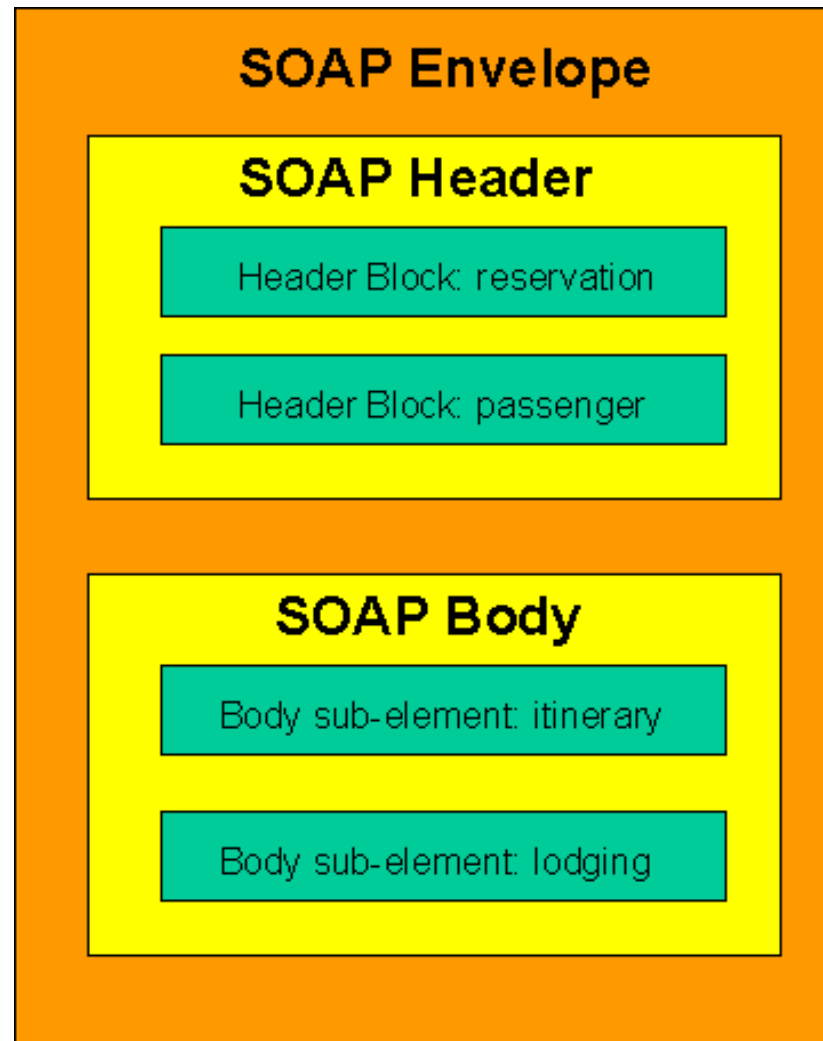
Konsumowanie usług

31

- Wywołanie usług polega na przesłaniu koperty SOAP, zawierającej:
 - ▣ Nagłówki (np. dane uwierzytelniające)
 - ▣ Informacje o metodzie do wywołania
 - ▣ Parametry
- Koperta jest wysyłana pod adres usługi, zdefiniowany w pliku WSDL
- Zawartość kopert SOAP wynika bezpośrednio z definicji usług w pliku WSDL
 - ▣ Dane, operacje, zwracane wartości itd.

Koperta SOAP

32



Żądanie SOAP RPC

33

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
      env:encodingStyle="http://example.com/encoding"
      env:mustUnderstand="true" >5</t:transaction>
  </env:Header>
  <env:Body>
    <m:chargeReservation
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:m="http://travelcompany.example.org/">
      <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
        <m:code>FT35ZBQ</m:code>
      </m:reservation>
      <o:creditCard xmlns:o="http://mycompany.example.com/financial">
        <n:name xmlns:n="http://mycompany.example.com/employees">
          Åke Jógvan Øyvind
        </n:name>
        <o:number>123456789099999</o:number>
        <o:expiration>2005-02</o:expiration>
      </o:creditCard>
    </m:chargeReservation>
  </env:Body>
</env:Envelope>
```

Żądanie SOAP RPC – nagłówki

34

```
<?xml version='1.0' ?>
<env:Envelope
xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
      env:encodingStyle="http://example.com/encoding"
      env:mustUnderstand="true" >5</t:transaction>
  </env:Header>
```

Żądanie SOAP RPC - ciało

35

```
<env:Body>
  <m:chargeReservation
    env:encodingStyle=http://www.w3.org/2003/05/soap-encoding
    xmlns:m="http://travelcompany.example.org">
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
      <m:code>FT35ZBQ</m:code>
    </m:reservation>
    <o:creditCard xmlns:o="http://mycompany.example.com/financial">
      <n:name xmlns:n="http://mycompany.example.com/employees">
        Åke Jógvan Øyvind
      </n:name>
      <o:number>123456789099999</o:number>
      <o:expiration>2005-02</o:expiration>
    </o:creditCard>
  </m:chargeReservation>
</env:Body>
</env:Envelope>
```

Odpowiedź SOAP RPC

36

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
      env:encodingStyle="http://example.com/encoding"
      env:mustUnderstand="true">5</t:transaction>
  </env:Header>
  <env:Body>
    <m:chargeReservationResponse
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:m="http://travelcompany.example.org/">
      <m:code>FT35ZBQ</m:code>
      <m:viewAt>
        http://travelcompany.example.org/reservations?code=FT35ZBQ
      </m:viewAt>
    </m:chargeReservationResponse>
  </env:Body>
</env:Envelope>
```

Odpowiedź SOAP RPC – nagłówki

37

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
      env:encodingStyle="http://example.com/encoding"
      env:mustUnderstand="true">5</t:transaction>
  </env:Header>
```

Odpowiedź SOAP RPC – ciało

38

```
<env:Body>
  <m:chargeReservationResponse
    env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
    xmlns:m="http://travelcompany.example.org/">
    <m:code>FT35ZBQ</m:code>
    <m:viewAt>
      http://travelcompany.example.org/reservations?code=FT35ZBQ
    </m:viewAt>
  </m:chargeReservationResponse>
</env:Body>
</env:Envelope>
```

Konsumowanie usług

39

- Na szczęście kopert SOAP nie trzeba budować ręcznie...
- Plik WSDL jest kompletnym i precyzyjnym opisem usługi
- Automatyczne generowanie klientów
- Wygenerowany klient może być używany jak zwykłe klasy lokalne
- Cała komunikacja jest ukryta przed programistą
 - ▣ Budowanie koperty SOAP
 - ▣ Kodowanie danych
 - ▣ Zapytanie HTTP

40

UDDI

UDDI

Universal Description, Discovery and Integration

41

- Standard opisujący interfejsy dla rejestrów usług internetowych
- Rejestry UDDI miały gromadzić dane o usługach publikowanych przez usługodawców
- Hierarchiczne repozytorium usług, np.:
 - ▣ Kategorie
 - Usługodawcy
 - Usługi
 - Operacje

- Możliwość przeglądania katalogu wg różnych kategorii (podobnie jak w książce telefonicznej)
 - ▣ *White pages* – katalog osobowy – informacje na temat usługodawców, nazwy firm, adresy, informacje kontaktowe
 - ▣ *Yellow pages* – katalog branżowy – informacje na temat rodzajów usług biznesowych oferowanych przez usługodawców
 - ▣ *Green pages* – opisują sposoby dostępu do usług i ich lokalizacje

UDDI

43

- Rejestr UDDI umożliwia wyszukiwanie usług pożądanых przez klienta
- Klient określa kryteria wyszukiwania
 - ▣ np. usługa realizacji płatności
- Rejestr zwraca listę usług spełniających kryteria
 - ▣ *Google dla web serviców*
- Klient wybiera usługę z listy i może ją wykorzystać
- UDDI miało ułatwiać odkrywanie usług, zwiększać ich dostępność dla potencjalnych klientów
 - ▣ Idea brzmi pięknie...

UDDI w praktyce

44

- Nigdy nie powstały publiczne rejestry UDDI o dużej skali i popularności
- Próby wykreowania takich rejestrów zakończyły się fiaskiem i ich zamknięciem po kilku latach działania
 - ▣ Rejestr UDDI firmy Microsoft
 - ▣ Rejestr UDDI firmy SAP
- Wygląda na to, że nikt nie potrzebuje Googla dla usług internetowych

UDDI w praktyce

45

- Usługodawcy nie chcą udostępniać usług przypadkowym klientom
 - ▣ Kontrakty na dostęp do usług, opłaty za korzystanie
- Klienci nie chcą korzystać z usług przypadkowo wyszukanych usługodawców
 - ▣ Problemy w działaniu usług mogą przekładać się na realne straty finansowe klienta
 - ▣ Klienci wolą wiarygodnych dostawców
 - ...z którymi mogą podpisać kontrakty z gwarancją dostępności usług – np. uptime 99,99% – i karami w razie braku dostępności

UDDI w praktyce

46

- Usługi są często zbyt specyficzne, aby warto było je promować w publicznych rejestrach
- Istnieją prywatne rejestry UDDI w zamkniętych środowiskach
- Rejestry UDDI funkcjonują najczęściej jako wewnętrzne komponenty w kompleksowych rozwiązaniach, nie jako samodzielnie usługi
 - ▣ np. rejestry UDDI wykorzystywane wewnętrznie przez serwer Microsoft BizTalk (ESB)

Bezpieczeństwo usług

47

- Rozwiązania na poziomie warstwy transportowej
 - ▣ Transport HTTP:
 - Szyfrowany kanał HTTPS
 - Uwierzytelnianie HTTP BASIC
- Rozwiązania na poziomie protokołu SOAP
 - ▣ WS-Security
 - Podpisywanie wiadomości (integralność, niezaprzeczalność)
 - Szyfrowanie (poufność)
 - Uwierzytelnianie
 - Login i hasło
 - Certyfikaty
 - Tokeny protokołu Kerberos i inne

48

SOAP Dziś

Krytyka

49

- Problemy z interoperacyjnością
 - ▣ np. Java ↔ .NET
- Kłopotliwa obsługa z poziomu języków skryptowych
 - ▣ Brak wsparcia narzędziowego w tych językach
- Przesadnie rozbudowane opisy usług (WSDL)
- Narzut kopert SOAP (XML)
 - ▣ Narzut na dane
 - ▣ Narzut na przetwarzanie formatu XML
 - W szczególności na urządzeniach mobilnych
- Pojedynczy punkt wejścia (*endpoint*) – black-box
- Problematiczne *cachowanie* wyników wywołań
- Brak wsparcia narzędziowego dla aplikacji mobilnych

SOAP dziś

50

- Wciąż obecny na rynku aplikacji klasy *enterprise* (rynek korporacyjny)
 - ▣ ...i jeszcze długo tam pozostanie
 - Dobrze zdefiniowane kontrakty między klientem i usługą – pożądane w środowiskach korporacyjnych
 - Wiele rozbudowanych narzędzi, ułatwiających pracę z usługami SOAP
 - Wiele istniejących już systemów, które będą działały jeszcze przez wiele lat
 - ▣ Historia protokołu SOAP przypomina nieco historię protokołu CORBA...

SOAP dziś

51

- Na rynku aplikacji konsumenckich ustępuje usługom REST
 - ▣ Łatwiejsze konsumowanie w językach z dynamicznym typowanie
 - Nie wymagają dedykowanych narzędzi, ani generowania klientów – wystarczy klient HTTP
 - ▣ Łatwiejsze konsumowanie na urządzeniach mobilnych
 - ▣ Mniejsze narzuty na przesyłane dane i ich przetwarzanie

52

Pytania?