Paweł Rajba
pawel@ii.uni.wroc.pl
http://www.itcourses.eu/

Web Services

Agenda

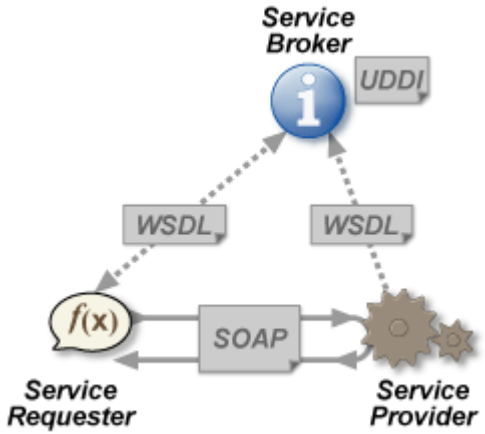
- Wprowadzenie
- SOAP RPC web services
  - Architektura
  - SOAP
  - WSDL
  - Rejestracja i wyszukiwanie usług
- RESTful web services
  - Co to jest? Wymagania
  - SOAP vs. REST
  - Zalety
  - Czego brakuje w REST?
- Co wybrać?

Wprowadzenie

- Komunikacja między aplikacjami
  - Dawniej
    - Dedykowane rozwiązania, brak standardów (np. DCOM, .NET Remoting)
    - Brak interoperatywności
  - Obecne oczekiwanie: międzyplatformowość
    - Jedno z rozwiązań: usługi sieciowe
- Usługi sieciowe
  - Oparte o standardowe protokoły (np. HTTP), formaty wiadomości (np. XML), itd..
- Dwa podejścia: SOAP RPC i REST

SOAP RPC web services

- Architektura
  - Aktorzy
    - Dostawca
    - Konsument
  - Komunikacja
    - SOAP
    - WSDL
    - Discovery



Źródło: http://en.wikipedia.org/wiki/Web_service

SOAP RPC web services

- Specyfikacje
  - Opisują poszczególne elementy rozwiązania
  - Pogrupowane tematycznie
- Kilka przykładów
  - Messaging Specification
    - SOAP, WS-Addressing
  - Metadata Exchange Specification
    - WS-Policy, WS-Discovery, WSDL 2.0
  - Security Specification
    - WS-Trust, WS-Federation, Security Assertion Markup Language (SAML)
  - Reliable Messaging Specifications
    - WS-Reliability
- Ciekawe podsumowanie: http://en.wikipedia.org/wiki/List_of_web_service_specifications

SOAP RPC web services

- SOAP
  - SOAP to Simple Object Access Protocol
  - Protokół komunikacyjny oparty o XML
  - Niezależny od warstwy transportowej
  - Schemat wiadomości

```
<soap:Envelope xmlns:"...">
    <soap:Header>
        <!-- info o securiy, trans, etc. -->
    </soap:Header>
    <soap:Body>
        <InvoiceRequest>...</InvoiceRequest>
        <soap:Fault>...</soap:Fault>
    </soap:Body>
</soap:Envelope>
```

SOAP RPC web services

- SOAP, przykładowe żądanie:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Źródło: http://www.w3schools.com/webservices/ws_soap_example.asp

SOAP RPC web services

- SOAP, przykładowa odpowiedź:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

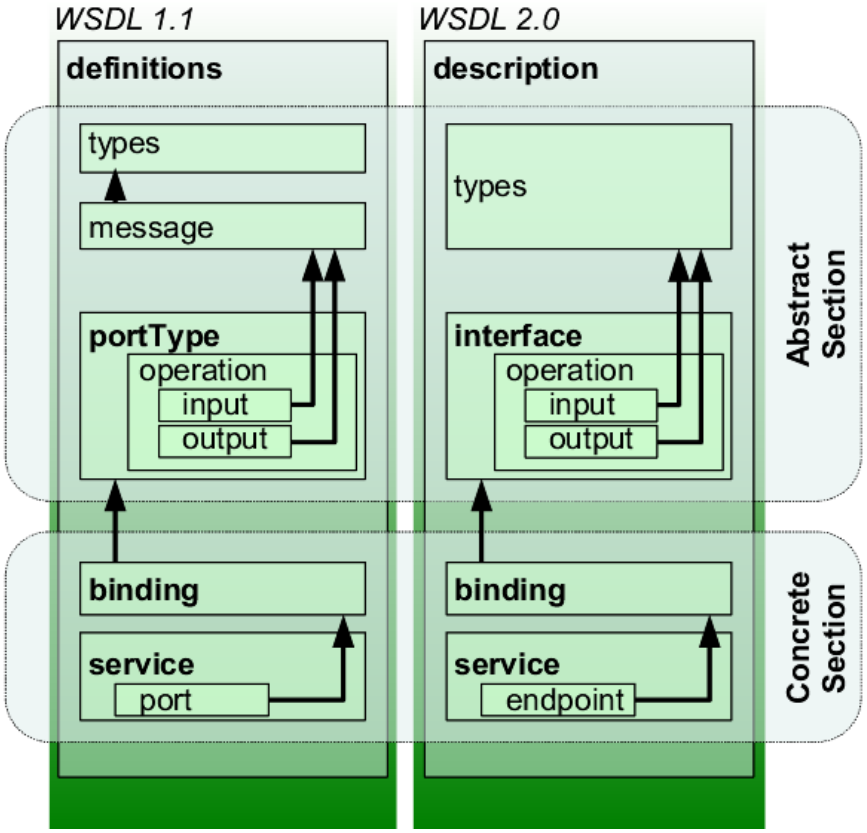
Źródło: http://www.w3schools.com/webservices/ws_soap_example.asp

SOAP RPC web services

- WSDL
  - WSDL to Web Services Description Language
  - Umożliwia opis WS i sposób dostępu
  - Oparty na XML, wersja 2.0 składa się z sekcji
    - <description> - główny element
    - <types> - typy danych użyte w usłudze (XSD)
    - <interfaces> - opis operacji wraz z typami przesyłanych komunikatów (możliwe odwołania do <types>)
    - <binding> - opis jak usługa jest dostępna
      - zwykle jest to powiązanie z protokołem HTTP
    - <service> - opis gdzie usługa jest dostępna
      - zwykle jest to jakiś adres URL
    - <documentation> - zawiera opis usługi; opcjonalny
    - <import> - pozwala na dołączenie zewnętrznych XSD lub WSDL; opcjo.

SOAP RPC web services

- WSDL: architektura



Źródło: http://en.wikipedia.org/wiki/Web_Services_Description_Language

SOAP RPC web services

- WSDL, przykładowe description

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns=            "http://www.w3.org/ns/wsdl"
    targetNamespace= "http://jenkov.com/MyService"
    xmlns:tns=        "http://jenkov.com/MyService"
    xmlns:stns =      "http://jenkov.com/MyService/schema"
    xmlns:wsoap=      "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap=       "http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx=      "http://www.w3.org/ns/wsdl-extensions" >

</description>
```

Źródło: http://tutorials.jenkov.com/wsdl/index.html

SOAP RPC web services

- WSDL, przykładowe types

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://jenkov.com/MyService"
    xmlns:tns=       "http://jenkov.com/MyService"
    xmlns:stns=     "http://jenkov.com/MyService/schema"
    . . . >

  ...

  <types>

    <xs:schema
        xmlns:xs=        "http://www.w3.org/2001/XMLSchema"
        targetNamespace= "http://jenkov.com/MyService/schema"
        xmlns:tns=       "http://jenkov.com/MyService/schema"
    >

        <xs:element name="latestTutorialRequest"
                    type="typeLatestTutorialRequest"/>

        <xs:complexType name="typeLatestTutorialRequest">
          <xs:sequence>
            <xs:element  name="date"   type="xs:date"/>
          </xs:sequence>
        </xs:complexType>

        <xs:element name="latestTutorialResponse" type="xs:string"/>

        <xs:element name="invalidDateError" type="xs:string"/>

    </xs:schema>

  </types>
  . . .
</description>
```

Źródło: http://tutorials.jenkov.com/wsdl/index.html

SOAP RPC web services

- WSDL, przykładowe interface

```
<interface  name = "latestTutorialInterface" >

  <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>

  <operation name="latestTutorialOperation"
          pattern="http://www.w3.org/ns/wsdl/in-out"
          style="http://www.w3.org/ns/wsdl/style/iri"
          wsdlx:safe = "true">

    <input    messageLabel="In"  element="stns:latestTutorialRequest" />
    <output   messageLabel="Out" element="stns:latestTutorialResponse" />
    <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />

  </operation>

</interface>
```

Źródło: http://tutorials.jenkov.com/wsdl/index.html

SOAP RPC web services

- WSDL, przykładowe binding

```
<binding name="latestTutorialSOAPBinding"
        interface="tns:latestTutorialInterface"
        type="http://www.w3.org/ns/wsdl/soap"
        wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

  <fault ref="tns:invalidDateFault" wsoap:code="soap:Sender"/>

  <operation ref="tns:latestTutorialOperation"
    wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

</binding>
```

Źródło: http://tutorials.jenkov.com/wsdl/index.html

SOAP RPC web services

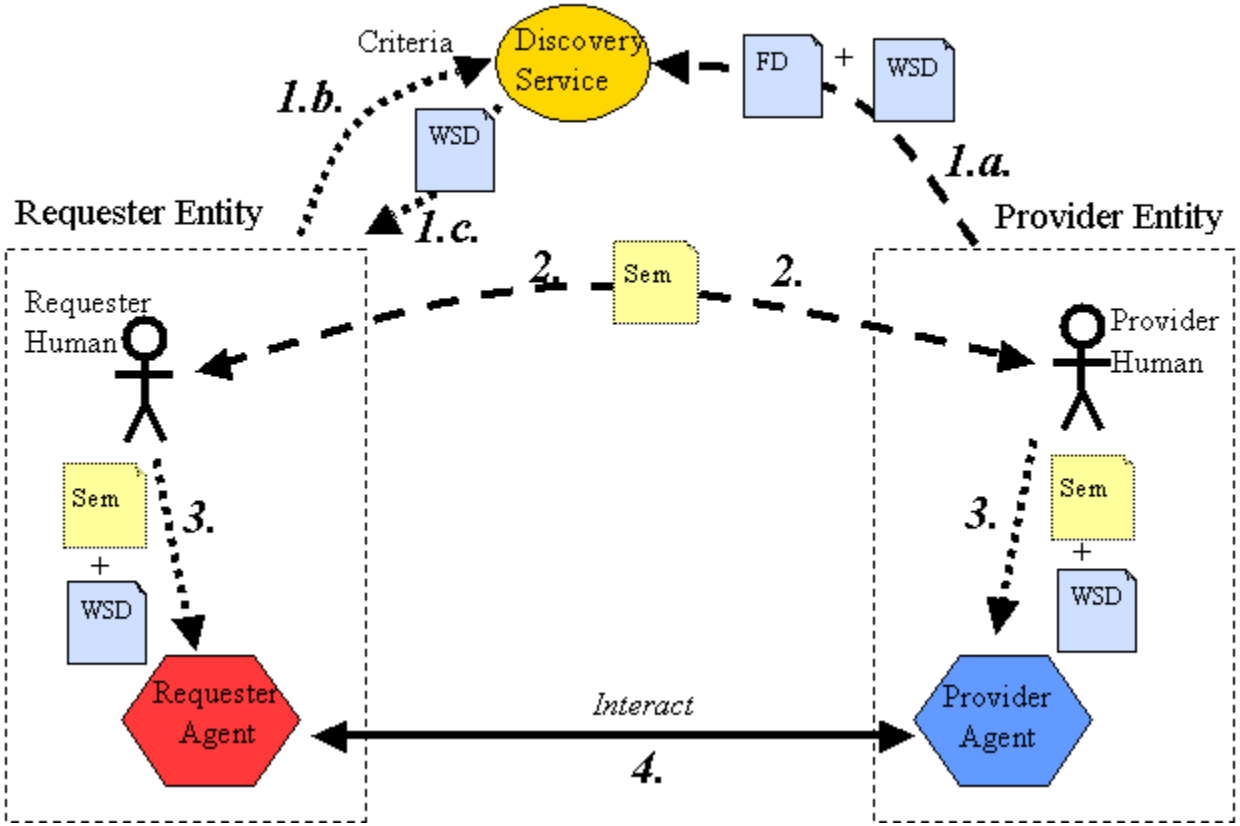
- WSDL, przykładowe service

```
<service
     name     ="latestTutorialService"
     interface="tns:latestTutorialInterface">

   <endpoint name ="latestTutorialEndpoint"
          binding ="tns:latestTutorialSOAPBinding"
          address ="http://jenkov.com/latestTutorial"/>

</service>
```

Źródło: http://tutorials.jenkov.com/wsdl/index.html

SOAP RPC web services

- Rejestracja i wyszukiwanie usług



Źródło: http://www.w3.org/TR/ws-arch/

RESTful services

- REST to REpresentational State Transfer
- Podstawowe składowe
  - Zasoby reprezentowane przez URI
    - Adres
    - Zasób
    - Parametry
  - Metody HTTP jako operacje
  - Znaczenie nagłówków HTTP
    - Content negotation
    - Status codes
    - Security, etag, custom headers
  - HTTP body
- Format danych to zwykle: JSON lub XML
- Przykłady:
  - POST http://example.com/srv/invoice/123
  - GET http://example.com/srv/invoice/123

RESTful services

- SOAP vs. REST
  - SOAP jest
    - Skupiony na operacjach
      - getUser(), addUser(), getAddress(), updateAddress()
    - Niezależny od warstwy transportowej
      - Np. może być powiązany z HTTP, MSMQ, TCP, itd.
  - REST
    - Skupiony na zasobach
      - User
      - Address
        - Poprzez metody protokołu HTTP wykonujemy operacje na zasobach
    - Silnie powiązany z HTTP (czyli warstwą transportową)

RESTful services

- Istotnym jest reprezentacja zasobu
  - Zasób może mieć różne reprezentacje stanu
  - Do reprezentacji zwykle używamy JSON lub XML
  - Przykładowo
    - Resource: osoba (Pawel)
    - Service: dane kontaktowe (pobierane przez GET)
    - Representation:
      - Imie, nazwisko, ulica, miasto, kod, telefon
        - Jako formatu użyjemy np. JSON-a

RESTful services

- Wymagane zasady obowiązujące w REST
  - Uniform interface
  - Stateless
  - Client-server
  - Cacheable
  - Layered system

RESTful services

- Uniform interface
  - Metody HTTP i ich interpretacja
    - GET – pobranie
    - POST – wstawienie nowego zasobu
    - PUT – aktualizacja
    - DELETE – usunięcie
  - URI zasobu
- Stateless
  - Server nie przechowuje stanu
  - Każde żądanie zawiera cały kontekst potrzebny do jego przetworzenia
    - tzw. *Self-descriptive messages*
  - Stan jeśli jest, pamiętany jest po stronie klienta

RESTful services

- Client-server
  - W tym modelu działa RESTful service
  - Chodzi też o silny rozdział odpowiedzialności
    - np. przechowywanie danych
- Cacheable
  - Odpowiedzi muszą mieć możliwość cache'owania
  - Może to być określone jawnie lub niejawnie
- Layered system
  - Klient nie może zakładać bezpośredniego połączenia
    - Być może odpowiedź jest z cache'a
    - Być może po drodze jest loadbalancer lub inny pośrednik (np. security)
  - Dla klienta istotne jest dostępne API
  - Takie podejście poprawia skalowalność rozwiązania

RESTful services

- Zalety
  - Caching
    - Wśród wymagań
    - Oparcie o HTTP niejako automatycznie to implementuje
  - Skalowalność
    - Bezstanowość znacznie ułatwia skalowalność
  - Idempotent
    - Wielokrotne wywołanie tej samej operacji na zasobie powinno być równoważne jednokrotnemu wywołaniu
    - W kontekście błędów z siecią, problemów z architekturą rozproszoną, taka cecha jest bardzo przydatna
  - Interoperability
    - Chociaż SOAP wspierał tą cechę, zastosowanie tylko HTTP jeszcze bardziej ją podwyższyło
    - Niektóre języki nie miały toolkita do SOAP, nie wszystkie elementy WS-* były zawsze wspierane
      - Przykładowo większość klientów mobilnych nie zaimplementowało WS-*
  - Simplicity – wyniki ze specyfikacji REST i jest jego główną zaletą

RESTful services

- HATEOAS
  - czyli Hypermedia as the Engine of Application State
  - W założeniu klient może operować z serwisem bez wcześniejszej wiedzy o nim

```
GET /account/12345 HTTP/1.1
Host: somebank.org
Accept: application/xml
```

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
   <account_number>12345</account_number>
   <balance currency="usd">100.00</balance>
   <link rel="deposit" href="/account/12345/deposit" />
   <link rel="withdraw" href="/account/12345/withdraw" />
   <link rel="transfer" href="/account/12345/transfer" />
   <link rel="close" href="/account/12345/close" />
 </account>
```

RESTful services

- Czego trochę brakuje w REST?
  - Opisu usług, chociaż są standardy, które można wykorzystywać
    - WADL
    - XRD: http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html
    - Swagger: http://swagger.io/
  - Wsparcia dla authN & authZ tak dobrego jak w SOAP
    - Jest co prawda OAuth i OpenID Connect, ale nie jest to tak dojrzałem jak rozwiązania w SOAP
  - Dla JSON-a brakuje odpowiednika XSD
    - Znowu są pewne rozwiązania, np. JSON Schema, ale nie są one jeszcze powszechnie zaimplementowane

Co wybrać?

- Wszystko zależy od scenariusza i potrzeb
- SOAP+WS-*
  - Więcej możliwości, bardzo bogata specyfikacja WS-*
  - Większa kontrola, bardziej złożone zastosowania
  - Ogromne wsparcie narzędzi
- REST
  - Bardzo prosty w implementacji, łatwa interoperacyjność
  - Wielu dostawców udostępnia usługi przez REST (Amazon, Yahoo, …)
  - Mniej funkcjonalności, brak wsparcia narzędzi

Literatura

- Web Services

  http://www.w3.org/TR/ws-arch/
  http://www.w3schools.com/webservices/
  http://www.w3.org/2002/ws/
  http://en.wikipedia.org/wiki/List_of_web_service_specifications
- WSDL 2.0

  http://www.w3.org/TR/wsdl20/
  http://en.wikipedia.org/wiki/Web_Services_Description_Language
  http://tutorials.jenkov.com/wsdl/index.html
- WADL

  https://wadl.java.net/
  http://www.w3.org/Submission/wadl/
  http://en.wikipedia.org/wiki/Web_Application_Description_Language
  http://stackoverflow.com/questions/2215646/difference-between-wsdl-2-0-wadl-xrd
- UDDI

  http://uddi.xml.org/
  http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration
- RESTful services

  http://en.wikipedia.org/wiki/Representational_state_transfer
  http://www.restapitutorial.com/lessons/whatisrest.html
  http://msdn.microsoft.com/en-us/magazine/dd315413.aspx
  http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven