SOAP

Hello!

Michał Nowakowski

lead software engineer @EPAM

michal@nowakowski.me.uk

Web Service

- Właściwość systemu informatycznego, polegająca na powtarzalnym wykonywaniu operacji na danych dostarczonych w uporządkowanej strukturze
- Zgodnie z zaleceniami W3C dane przekazywane powinny być w formacie XML przy użyciu protokołu HTTP
- Usługa Web Service – **samodokumentujący** się komponent, który może zostać wywołany zdalnie przez aplikacje klienckie

SOAP
Simple Object Access Protocol

- Protokół umożliwiający zdalne wywoływanie usług
- Oficjalny standard W3C
- Z technicznego punktu widzenia są to z reguły zwykłe żądania i odpowiedzi HTTP
- Przesyłane wiadomości zapisane są w formacie XML
- Do przesyłania wykorzystywany jest najczęściej protokół HTTP (inne: SMTP, FTP, JMS)

SOAP
Simple Object Access Protocol

- Protokół niezależny od języka programowania, systemu operacyjnego (Win, Linux, Java, .NET, PHP, C++)
- Możliwość wykorzystania różnych protokołów sieciowych do transportu danych
- Silnie sformalizowany (WSDL)
- Łatwy do rozszerzenia

SOAP
Czemu nie REST?

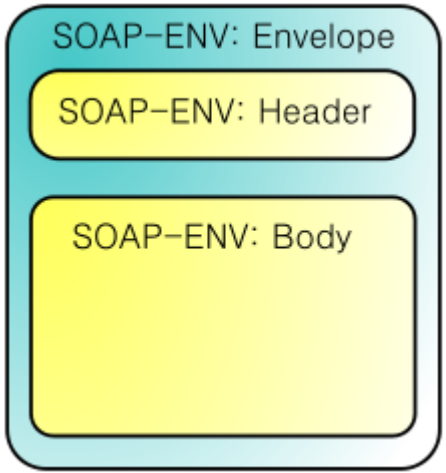
- Określony, wymuszony format wiadomości
- Wsparcie dla asynchronicznych operacji (jako część standardu)
- Zaprojektowany do operacji na rozbudowaych obiektach
- Wsparcie dla autoryzacji, szyfrowania danych (WS-Security)
- WS-AtomicTransaction – transakcyjność, jeden serwis wywołuje kolejny serwis (jako pojedyczna transakcja)
- WS-ReliableMessaging – potwierdzenie dostarczenia komunikatu
- Wbudowana obsługa błędów

SOAP
A może jednak REST?

- Łatwe cacheowanie (np. wszystkie metody GET)
- Bezstanowy (stateless)
- Łatwiejszy do zrozumienia!

SOAP
Struktura wiadomości

- Koperta (Envelope) – zawiera w sobie całą wiadomość (root node)
- Nagłówek (Header) – opcjonalnie, informacje dotyczące np. autoryzacji, adresu zwrotnego
- Ciało wiadomości (Body) – właściwa treść – wywołanie metody lub odpowiedź
- Informacja o błędzie (Fault) – opcjonalnie, jako część ciała wiadomości



SOAP
Wywołanie metody

- Adres serwisu (URL)
- Nazwa metody
- Nazwy i wartości parametrów (jeśli występują)
- Nagłówek (opcjonalnie)

SOAP
Przykładowe żądanie

```
POST http://localhost:9999/service HTTP/1.1
Content-Type: text/xml;charset=UTF-8
Content-Length: 314

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:ns="http://service.infoshareacademy.com/">
    <soapenv:Header/>
    <soapenv:Body>
        <ns:toUpperCase>
            <value>Ala ma kota.</value>
        </ns:toUpperCase>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP
Przykładowa odpowiedź

```
HTTP/1.1 200 OK
Date: Wed, 10 May 2017 19:50:59 GMT
Content-type: text/xml;charset=utf-8

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:ns2="http://service.infoshareacademy.com/">
    <soapenv:Body>
        <ns2:toUpperCaseResponse>
            <return>ALA MA KOTA.</return>
        </ns2:toUpperCaseResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP

Przykładowa odpowiedź (błąd)

```
HTTP/1.1 500 Internal Server Error
Date: Wed, 10 May 2017 19:50:59 GMT
Content-type: text/xml;charset=utf-8
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <soapenv:Fault xmlns:ns="http://www.w3.org/2003/05/soap-envelope">
            <faultcode>S:Server</faultcode>
            <faultstring>Unknown ID</faultstring>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP i Java
JAX-WS

- **J**ava **A**PI for **X**ML **W**eb **S**ervices
- Część Java Enterprise Edition
- Część Java Standard Edition od wersji 6
- Określa w jaki sposób mapowane są metody Java do metod web serwisu
- Inne implementacje zgodne z JAX-WS: Apache CXF, Apache Axis2

SOAP i Java
Opis serwisu

- Serwis deklarujemy jako interfejs
- Interfejs opisujący serwis oraz klasę implementującą interfejs oznaczamy adnotacją @***WebService***
- Metody serwisu oznaczamy adnotacją @***WebMethod*** (opcjonalne)
- Parametry metod serwisu oznaczamy adnotacją @***WebParam*** (opcjonalne)

SOAP i Java
@WebService

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
public @interface WebService {
    String name() default "";

    String targetNamespace() default "";

    String serviceName() default "";

    String portName() default "";

    String wsdlLocation() default "";

    String endpointInterface() default "";
}
```

SOAP i Java
@WebMethod

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD})
public @interface WebMethod {
    String operationName() default "";

    String action() default "";

    boolean exclude() default false;
}
```

SOAP i Java

@WebParam

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.PARAMETER})
public @interface WebParam {
    String name() default "";

    String partName() default "";

    String targetNamespace() default "";

    WebParam.Mode mode() default WebParam.Mode.IN;

    boolean header() default false;

    public static enum Mode {
        IN, OUT, INOUT;
        private Mode() {}
    }
}
```

SOAP i Java

Przykładowy interfejs

```
package com.infoshareacademy.service;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService
public interface AcademyService {

    @WebMethod
    String toUpperCase(@WebParam(name = "value") String value);
}
```

SOAP i Java

Przykładowa implementacja

```
package com.infoshareacademy.service;

import javax.jws.WebService;

@WebService(endpointInterface =
"com.infoshareacademy.service.AcademyService")
public class AcademyServiceImpl implements AcademyService {

    @Override
    public String toUpperCase(String value) {
        if (value != null) {
            return value.toUpperCase();
        }

        return null;
    }
}
```

SOAP i Java
Wystawienie serwisu

```
public static Endpoint publish(String address, Object implementor)
```

SOAP i Java
Wystawienie serwisu

```
package com.infoshareacademy.server;

import com.infoshareacademy.service.AcademyServiceImpl;
import javax.xml.ws.Endpoint;

public class Main {

    public static void main(String[] args) {
        Endpoint.publish("http://localhost:9999/service",
            new AcademyServiceImpl());
    }
}
```

SOAP i Java
Ćwiczenie 1

- $ git clone https://github.com/infoshareacademy/jjdd2-materialy-api-rest-soap

- Zaimplementuj serwis AcademyService prezentowany na poprzednich slajdach
- Wystaw usługę

- WSDL: http://localhost:9999/service?wsdl
- XSD: http://localhost:9999/service?xsd=1

SOAP i Java

Wywołanie serwisu (cURL)

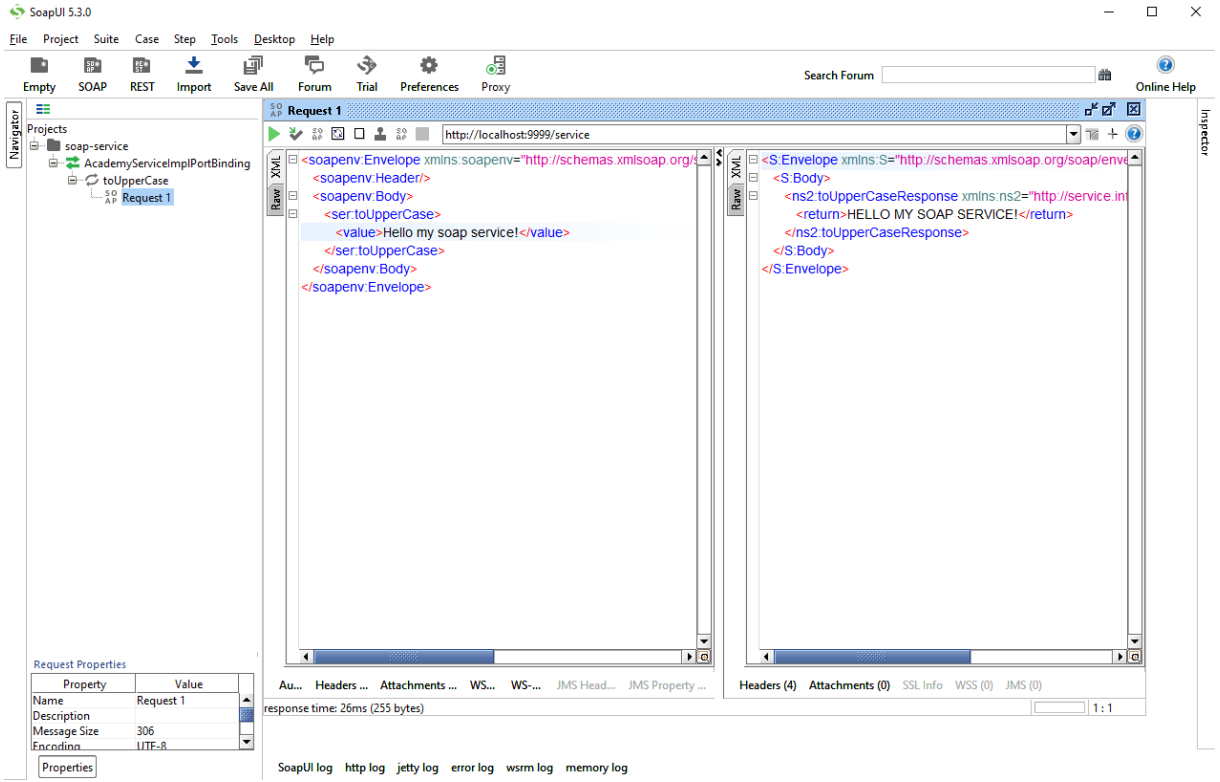
```
$ curl --header "Content-Type: text/xml;charset=UTF-8"
--data @test-data/example-request-1.xml
http://localhost:9999/service
```

SOAP i Java
SOAP UI

- Darmowe narzędzie do testowania serwisów SOAP i REST (OpenSource)
- 100% Java (wieloplatformowość)
- https://www.soapui.org/downloads/soapui.html

SOAP i Java

Wywołanie serwisu (SOAP UI)



SOAP i Java
Ćwiczenie 2

- Dodaj do web serwisu metodę, która zwróci największy wspólny dzielnik dwóch podanych liczb
- Przetestuj metodę przy użycia cURLa lub SOAP UI

SOAP
XML Schema Definition

- XML Schema Definition dla naszego serwisu: http://localhost:9999/service?xsd=1

XSD
XML Schema Definition

- Standard definiowania struktury dokumentu XML
- Dokument XSD sam w sobie jest dokumentem XML

XML Schema Definition

Typy danych

- Typy proste (`<xs:element>`)
- Typy złożone (`<xs:complex>`)

XML Schema Definition

Typy proste

- Element dokumentu XML, który może zawierać tylko pojedynczą wartość (string, int, date, etc.)
- Nie może zawierać innych elementów ani atrybutów

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

XML Schema Definition
Typy złożone

- Elementy dokumentu XML, które mogą zawierać inne elementy oraz atrybuty

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname"
         type="xs:string"/>
      <xs:element name="lastname"
         type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

XML Schema Definition

Typy danych – XSD i Java

XML Schema Type	Java Data Type
xsd:string	java.lang.String
xsd:integer	java.math.BigInteger
xsd:int	int
xsd.long	long
xsd:short	short
xsd:decimal	java.math.BigDecimal
xsd:float	float
xsd:double	double
xsd:boolean	boolean
xsd:byte	byte
xsd:QName	javax.xml.namespace.QName

XML Schema Definition

Typy danych – XSD i Java

XML Schema Type	Java Data Type
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]
xsd:unsignedInt	long
xsd:unsignedShort	int
xsd:unsignedByte	short
xsd:time	javax.xml.datatype.XMLGregorianCalendar
xsd:date	javax.xml.datatype.XMLGregorianCalendar
xsd:g	javax.xml.datatype.XMLGregorianCalendar
xsd:anySimpleType	java.lang.Object
xsd:anySimpleType	java.lang.String
xsd:duration	javax.xml.datatype.Duration

XML Schema Definition
XMLGregorianCalendar

- Implementacja typów reprezentujących datę i czas ze specyfikacji W3C XML Schema 1.0

- Konwertowanie obiektu LocalDateTime do XMLGregorianCalendar:

```
private XMLGregorianCalendar toXMLDate(LocalDateTime localDateTime)
        throws DatatypeConfigurationException {
    DateTimeFormatter dtf = DateTimeFormatter.ISO_DATE_TIME;
    return DatatypeFactory.newInstance()
            .newXMLGregorianCalendar(
                localDateTime.truncatedTo(ChronoUnit.SECONDS).format(dtf));
}
```

XML Schema Definition
XMLGregorianCalendar

- Konwertowanie obiektu XMLGregorianCalendar do LocalDateTime:

```
private LocalDateTime toLocalDateTime(XMLGregorianCalendar xmlGregorianCalendar) {
    return LocalDateTime.ofInstant(
            xmlGregorianCalendar.toGregorianCalendar().getTime().toInstant(),
            ZoneId.systemDefault());
}
```

XML Schema Definition

Przykład

```
<xs:element name="note">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

XML Schema Definition

Przykład (Java Bean) - @XmlElement

```
public class Note {

    @XmlElement
    private String to;

    @XmlElement
    private String from;

    @XmlElement
    private String heading;

    @XmlElement
    private String body;

    // constructor
}
```

XML Schema Definition

Przykład (Java Bean) - @XmlAccessorType

```
@XmlAccessorType(XmlAccessType.FIELD)
public class Note {

    private String to;
    private String from;
    private String heading;
    private String body;

    // constructor
}
```

XML Schema Definition

Przykład (Java Bean) – settery i gettery

```
public class Note {

    private String to, from, heading, body;

    // constructor

    public String getHeading() {
        return heading;
    }

    public void setHeading(String heading) {
        this.heading = heading;
    }

    // setters and getters
}
```

SOAP
Web Services Description Language

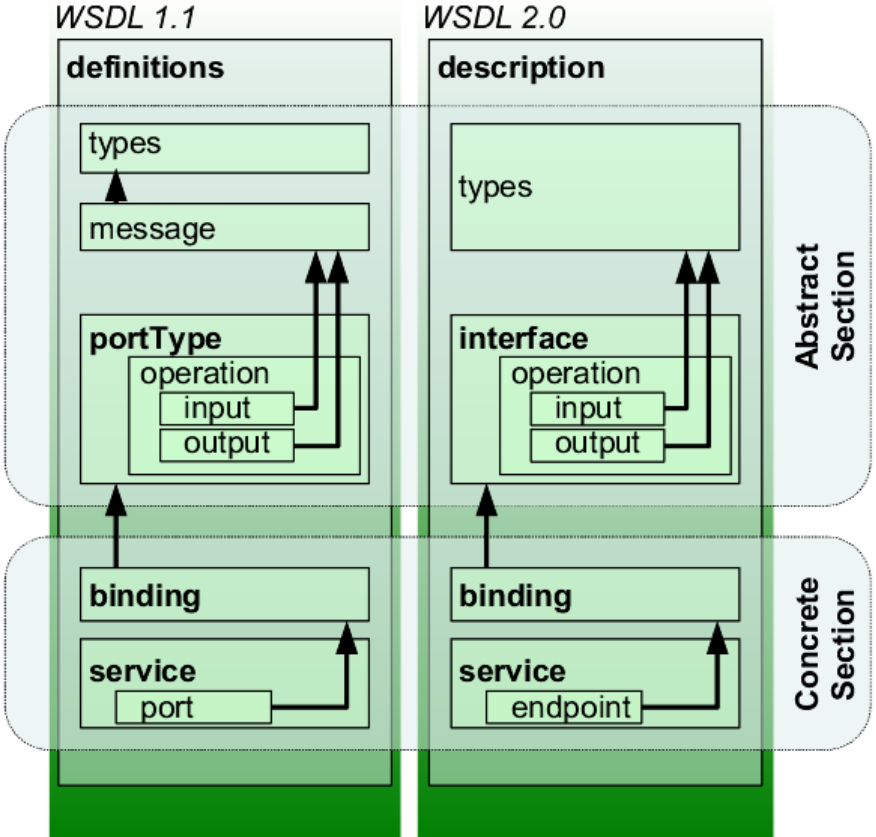
- Dokument WSDL dla naszego serwisu:
  http://localhost:9999/service?wsdl

WSDL
Web Services Description Language

- Kompletny opis serwisu w formacie XML
- Definiuje usługę sieciową razem ze wszystkimi wykorzystywanymi strukturami danych
- Na podstawie pliku WSDL aplikacja kliencka wie, w jaki sposób przesyłać i interpretować dane
- Programista koncentruje się na logice aplikacji, a nie szczegółach transportu danych czy formacie danych

WSDL

Elementy WSDL



WSDL
Elementy WSDL

- `<definitions>` – element nadrzędny (root tag)
- `<service>` i `<port>` – adresy punktów dostępowych (URLe)
- `<binding>` - metoda kodowania parametrów wejściowych i wyjściowych
- `<portType>` - deklaracja funkcji biznesowych oferowanych przez usługę (grupuje metody)
- `<operation>` - deklaracja operacji (metody w Javie), jej atrybuty wejścia i wyjścia (`<input>` i `<output>`)
- `<message>` - struktura zapytań i odpowiedzi
- `<types>` – typy danych wykorzystywane przez usługę (XSD)

WSDL

Przykładowy plik WSDL (1)

```
<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>
```

WSDL

Przykładowy plik WSDL (2)

```
<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

Przykładowy plik WSDL (3)

```
<service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
        <soap:address location="http://example.com/stockquote"/>
    </port>
</service>
```

SOAP i Java
Kolekcje

- JAX-WS / JAXB poradzi sobie z tablicami, np. String[], ale co z kolekcjami takimi jak List, Set, Map?

- Jak wygląda pole typu List w zwracanym XMLu?

```
public class GradesListWrapper {

    private ArrayList<Integer> grades = new ArrayList<>();

    // setter
    // getter
}
```

SOAP i Java
Ćwiczenie 3

- Stwórz klasę opisującą wyniki w nauce pojedycznego ucznia (imię, nazwisko, lista ocen)
- Stwórz serwis SOAP, który pobierze listę uczniów i zwróci nazwiska uczniów razem ze średnią ocen dla każdego ucznia, tj. pary nazwisko – średnia
- Przetestuj serwis przy użyciu SOAP UI
- Przeanalizuj generowany plik WSDL i XSD

SOAP i Java
Klient SOAP

- Interfejs serwisu
- Klasy używane przez serwis
- URL serwisu

SOAP i Java
Klient SOAP

```
private static final String SERVICE_URL = "http://localhost:9999/service";

public static void main(String[] args) throws MalformedURLException {

    URL url = new URL(SERVICE_URL);
    QName qName = new QName("http://service.infoshareacademy.com/",
            "AcademyServiceImplService");

    Service service = Service.create(url, qName);
    AcademyService academyService = service.getPort(AcademyService.class);

    System.out.println(academyService.toUpperCase("Hello from my SOAP Client!"));
}
```

SOAP i Java
Ćwiczenie 4

- Napisz klienta serwisu stworzonego w Ćwiczeniu 3

SOAP i Java
Generowanie kodu z XSD

- `$ xjc -d src/main/java/ -p com.infoshareacademy.soap.model src/main/resources/stock-quote.xsd`

- -p PACKAGE – pakiet dla klas wynikowych
- -d DESTINATION – ścieżka zapisu plików wynikowych

SOAP i Java
Ćwiczenie 5

- Stwórz dokument XSD opisujący cenę akcji na giełdzie (identyfikator, pełna nazwa, cena, data notowania, wolumen)
- Zamodeluj także obiekt żądania i odpowiedzi dla metody zwracającej opisany wyżej obiekt
- Metoda powinna pobierać jako argument identyfikator akcji
- Przetestuj serwis przy użyciu SOAP UI

SOAP i Java
Klient na podstawie WSDL

- Klasy opsujące web serwis generowane są na podstawie dokumentu WSDL

- `$ wsimport -s src/main/java WSDL_URL`

```
Service service = new CurrencyServiceImplService();
currencyService = service.getPort(CurrencyService.class);
double rate = currencyService.getConversionRate(Currency.USD,
        Currency.PLN);
```

SOAP i Java
Ćwiczenie 6

- WSDL serwisu zwracającego kurs wymiany wybranych walut: http://176.122.224.213:21370/service?wsdl

- Napisz aplikację webową, która przekonwertuje podaną kwotę w danej walucie na inną wybraną walutę przy użyciu web serwisu podanego powyżej (np. 100 PLN do GBP)

Thanks!!

Q&A