Laboratorium 9 - SOA.
Tematyka: JAX- WS

Tworzenie i korzystanie z serwisów SOAP-owych w Javie.

Celem laboratorium jest zaznajomienie z Web Service SOAP-owymi tworzonymi w oparciu o bibliotekę JAX-WS.

Uwaga. Ważna informacja – Od wersji jdk 11 kilka do tej pory kluczowych pakietów zostało usunięte z pakietu. Jednym z tych pakietów jest JAX- WS. Oznacza to że musimy ręcznie dociągnąć brakujące pakiety z repozytorium Mavena.

Więcej na ten temat można znaleźć np. tutaj -> https://openjdk.java.net/jeps/320

Proponuje wiec rozpoczęcie jakichkolwiek prac od stworzenia szkielety projektu z dociągniętymi następującymi pakietami:

```
<dependency>
   <groupId>javax.xml.bind</groupId>
   <artifactId>jaxb-api</artifactId>
   <version>2.2.8</version>
</dependency>
<dependency>
   <groupId>com.sun.xml.bind</groupId>
   <artifactId>jaxb-core</artifactId>
   <version>2.3.0</version>
</dependency>
<dependency>
   <groupId>com.sun.xml.bind</groupId>
   <artifactId>jaxb-impl</artifactId>
   <version>2.3.2</version>
</dependency>

<dependency>
   <groupId>com.sun.xml.ws</groupId>
   <artifactId>jaxws-ri</artifactId>
   <version>2.3.0</version>
   <type>pom</type>
</dependency>3.
```

Istnieją dwa podejścia do tworzenia usług sieciowych typu SOAP:

1. Podejście z góry na dół — strategia ta oznacza tworzenie usługi sieciowej na podstawie pliku WSDL. To podejście stosowane najczęściej w sytuacji, gdy usługa sieciowa powstaje całkowicie od podstaw. Jest podejściem preferowanym przez inżynierów lubiących „czyste” usługi sieciowe bazujące tylko i wyłącznie na wymaganiach biznesowych, bo kontrakt definiują osoby od strony biznesowej, a oprogramowanie musi się do tego dostosować.

2. Podejście z dołu do góry — w tej strategii plik WSDL powstaje na podstawie interfejsów programistycznych. Stosuje się je najczęściej, gdy zadanie polega na udostępnieniu usługi sieciowej dla już istniejącej aplikacji. Ponieważ podejście to nie wymaga głębokiej znajomości składni WSDL, jest łatwiejsze w realizacji, bo zamiana klas Javy i komponentów EJB na usługę sieciową może odbyć się automatycznie.

Przed rozpoczęciem prac implementacyjnych proponuje zaznajomić się z tutorialami Intelij opisującymi dokładnie jak pracować z Web Service oraz jak generować Clienta dla stworzonego Web Service.

Obługa JAX-WS web service

I. „Ręczne" tworzenie Web Serwisu - podejście 2 (Bottom-Up (Contract-Last))

1) Stwórz nowy moduł w którym samodzielnie zaimplementujesz (lub skorzystasz z wbudowanego generatora przykładu typu HelloWord) prostą klasę o podstawowej funkcjonalności.

Stworzymy interfejs publiczny wystawiany poprzez Endpoint orz jego implementacje. Plik WSDL zostanie wygenerowany w momencie publikacji Web Serwisu.

Zaczynamy od stworzenia klasy z modelem daych np *Pracownik*.

//Pracownik - klasa modelu danych:

```
public class Pracownik {
    private int id;
    private String firstName;

    //  gettery i settery
}
```

Web Service - Endpoint Interface

Następnie definujemy interfejs dostępowy dla przyszłej usługi.

```
package pl.agh.kis.soa.webservice;
```

```
import javax.jws.WebMethod;

import javax.jws.WebService;


@WebService
public interface PracownikService {
    @WebMethod
    Pracownik getPracownik(int id);

    @WebMethod
    Pracownik updatePracownik(int id, String name);

    @WebMethod
    boolean deletePracownik(int id);

    @WebMethod
    Pracownik addPracownik(int id, String name);

    // ...
}
```

Wewnątrz adnotacji @WebService można określić dodatkowe elementy, na przykład element targetNamespace, który deklaruje przestrzeń nazw używaną przez elementy WSDL dla generowanej usługi sieciowej. Jeżeli wspomniany element nie pojawi się, kontener usługi sieciowej użyje jako domyślnej przestrzeni nazw XML nazwy pakietu.

Element serviceName służy do określenia nazwy usługi. Nazwa wskazana w elemencie służy również do generowania atrybutu name w elemencie Service interfejsu WSDL. Brak tego elementu spowoduje użycie nazwy domyślnej, którą jest nazwa klasy ziarna wraz z tekstem Service.

Kolejny wiersz przy użyciu adnotacji @javax.jws.SOAPBinding wskazuje, że usługa sieciowa jest typu RPC (Remote Procedure Call). lub DOCUMENT (domyślnie).

Dołączenie adnotacji @WebMethod do metody publicznej oznacza, iż chce się, by ta metoda stała się częścią usługi sieciowej.

Adnotacja @WebParam służy do wskazania nazw parametrów prezentowanych w WSDL. Zawsze warto rozważyć użycie wspomnianej adnotacji, szczególnie w przypadku metod wieloargumentowych, ponieważ w przeciwnym razie WSDL użyje domyślnych nazw parametrów (na przykład arg0), które użytkownikom usługi nic nie mówią.

Adnotacja @WebResult przypomina adnotację @WebParam w tym sensie, że również umożliwia wskazanie nazwy dla wartości zwracanej przez WSDL.

Web Service Implementation

```
@WebService(endpointInterface = "pl.agh.kis.soa.PracownikService")
public class Pracownik ServiceImpl implements PracownikService {

    @Inject
    private PracownikRepository PracownikRepositoryImpl;
```

```
    @WebMethod
    public Pracownik getPracownik(int id) {
        return PracownikRepositoryImpl.getPracownik(id);
    }

    @WebMethod
    public Pracownik updatePracownik(int id, String name) {
        return PracownikRepositoryImpl.updatePracownik(id, name);
    }

    @WebMethod
    public boolean deletePracownik(int id) {
        return PracownikRepositoryImpl.deletePracownik(id);
    }

    @WebMethod
    public Pracownik addPracownik(int id, String name) {
        return PracownikRepositoryImpl.addPracownik(id, name);
    }

    // ...
}
```

Opublikowanie Web Service Endpoints

Kolejnym krokiem jest napisanie implementacji klasy udostępniającej usługę.
Aby opublikować usługę należy podac jako parametry metody publish z pakietu *javax.xml.ws.Endpoint* addres oraz instancje implementacji Web Serwisu

```
public class PracownikServicePublisher {
    public Endpoint.publish("http://localhost:8080/Pracownikservice",
                new PracownikServiceImpl());
    }
}
```

Uruchom napisaną aplikację -> w jego wyniku web serwis został opublikowany

Można ją szybko przetestować wpisując w adresie
http://localhost:8080/Pracownikservice?wsdl

Samodzielne stworzenie aplikacji klienckiej dla udostępnionej usługi JAXWS Client

Tworzymy nowy moduł na potrzeby stworzenia aplikacji klienckiej.

Teraz potrzebujemy wygenerowania szkieletu plików potrzebnych do implementacji aplikacji klienckiej tzw. client stubs.

Można to zrobić np. tak:

Z linii komend użyj polecenia wsimport:

cd %project_home%/src

wsimport -s . http://localhost:8080/Pracownikservice?wsdl

w jego wyniku wygenerowane zostaną w katalogu src projektu pliki potrzebne do uruchomienia klienta

napiszmy teraz implementacje l **pl.agh.soa.webservice**

```
public class PracownikServiceClient {
   public static void main(String[] args) throws Exception {
     URL url = new URL("http://localhost:8080/Pracownikservice?wsdl");

 PracownikService_Service PracownikService_Service =
           new PracownikService_Service(url);
 PracownikService PracownikServiceProxy =
          PracownikService_Service.getPracownikServiceImplPort();

     List<Pracownik> allPracowniks
              = PracownikServiceProxy.getAllPracowniks();
    }
}
```

Uruchom aplikację i sprawdź otrzymany wynik w konsoli tekstowej.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Do przećwiczenia na zajęciach są następujące zagadnienia:

Zadanie 1.

1. Tworzenie prostego Web Serwisy składającego się z jednej metody udostępniającej informacje o ilości dni do początku wakacji (przyjmijmy że jest nim 1.07) .

2. Po stworzeniu serwisu przetestowanie go bezpośrednio w przeglądarce.

3. Wygenerowanie na podstawie WSDL aplikacji klienckiej w celu przetestowania użycia serwisu.

Uwaga . Aplikacja kliencka może być stworzona na dwa sposoby; ręcznie lub za pomocą kreatora. Proszę spróbować obu wersji.

Zadanie 2.

Stworzenie Web Serwisy, który wymaga podania parametrów. Niech nasz Web Service obejmuje funkcjonalność kantoru walutowego. Powinien mieć następujące funkcjonalności: Informacja o aktualnym kursie (jako parametr podajemy symbol waluty), sprzedaż waluty (parametry to waluta(symbol), ilość waluty) zwracana wartość to ilość PLN.

Następnie proszę przetestować stworzony Web Service pisząc aplikację kliencką. (jako oddzielny deploy)

Zadanie 3 .

Generowanie usług Web Services na podstawie opisu WSDL.

Celem zadanie jest opracowanie opisu serwisu w postaci pliku WSDL oraz wygenerowanie aplikacji serwera i klienta na podstawie tego pliku. Zadaniem serwisu będzie analiza statystyczna podanego jako argument ciągu znaków i zwrócenie informacji o: ilości znaków, ilości białych znaków, ilości dużych liter, ilości małych liter, ilości cyfr w ciągu znaków.