

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.

Wprowadzenie do tematyki można znaleźć tutaj:

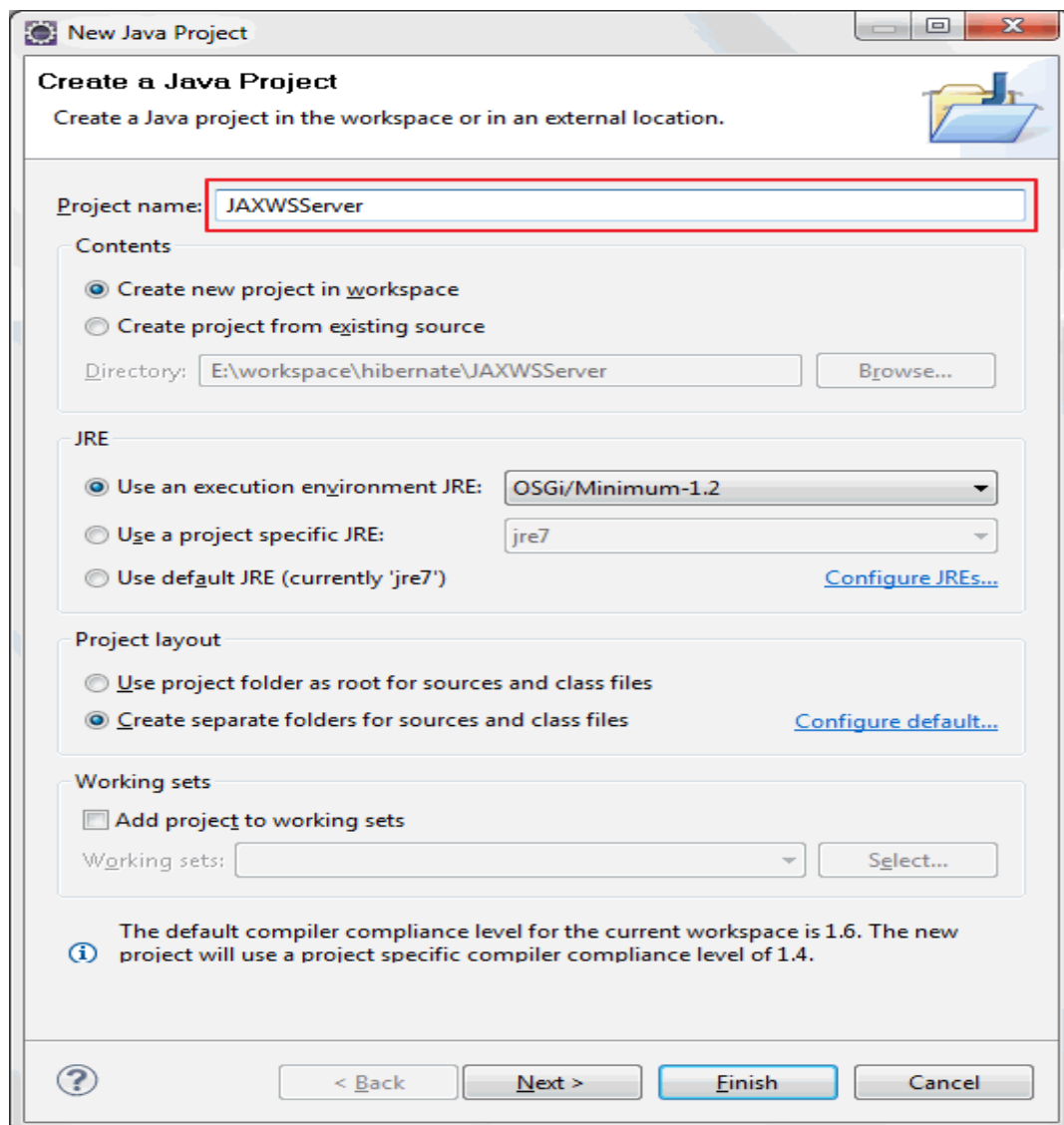
<https://docs.oracle.com/javaee/7/tutorial/jaxws.htm#BNAYL>

Obluga JAX-WS web service

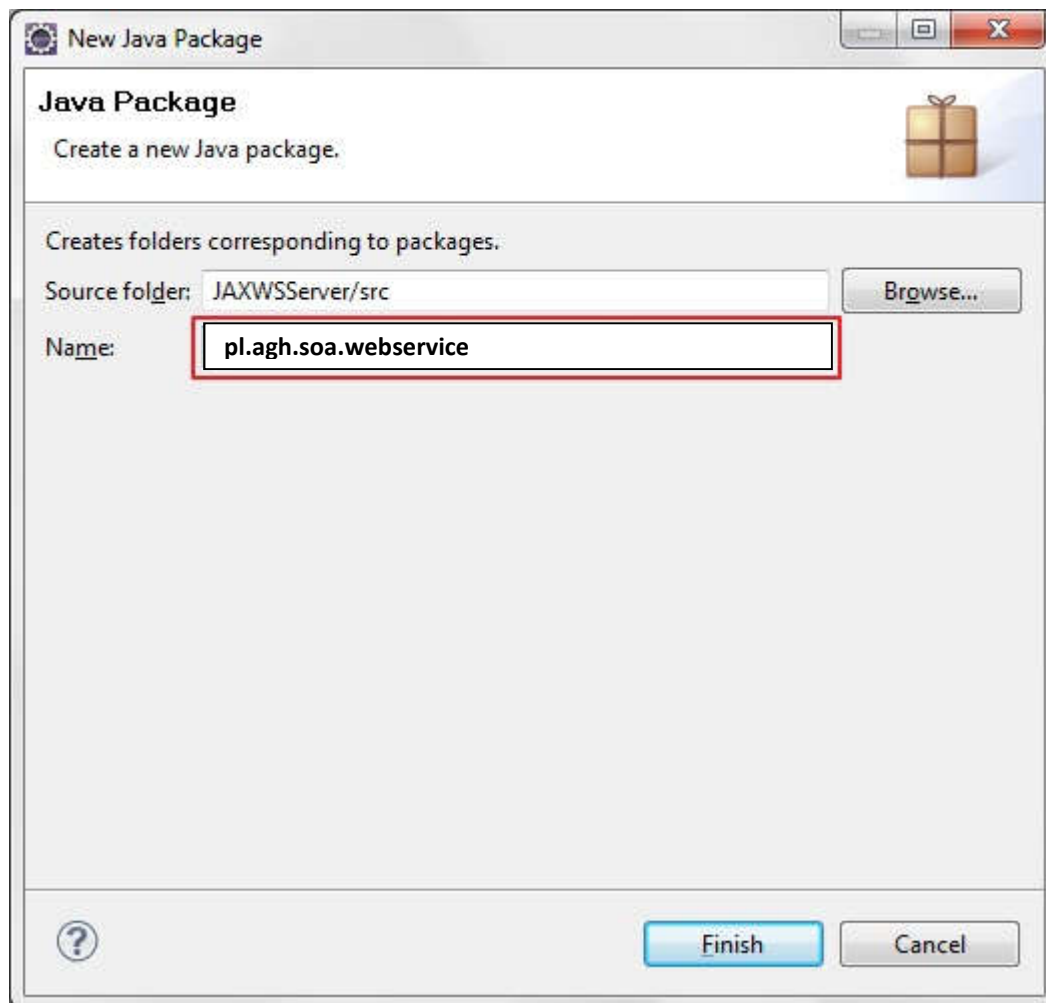
I. Reczne tworzenie Web Serwisu

Tworzenie usługi JAX-WS webservice

- 1) Uruchom Eclipse IDE
- 2) Stwórz zwykły project w Java np. "JAXWSServer"



3) Stwórz nowy pakiet "pl.agh.soa.webservice"



4) Stwórz Interfejs dla przyszłej usługi.

HelloWorld.java

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

```
import javax.xml.ws.WebMethod;
```

```
import javax.xml.ws.WebService;
```

```
@WebService
```

```
public interface HelloWorld {
```

```
    @WebMethod public String helloWorld(String name);
```

```
}
```

5) Stwórz implementację interfejsu.

HelloWorldImpl.java

```

package pl.agh.soa.webservice;

import javax.jws.WebService;

@WebService(endpointInterface="pl.agh.soa.webservice.HelloWorld")

public class HelloWorldImpl implements HelloWorld {

    public String helloWorld(String name) {

        return "Pozdrowienia od "+name;

    }

}

```

6) Napisz implementację klasy udostępniającej usługę.

HelloWorldWSPublisher.java

```

package pl.agh.soa.webservice;

import javax.xml.ws.Endpoint;

public class HelloWorldWSPublisher {

    public static void main(String[] args) {

        Endpoint.publish("http://localhost:8080/WS/HelloWorld", new HelloWorldImpl());

    }

}

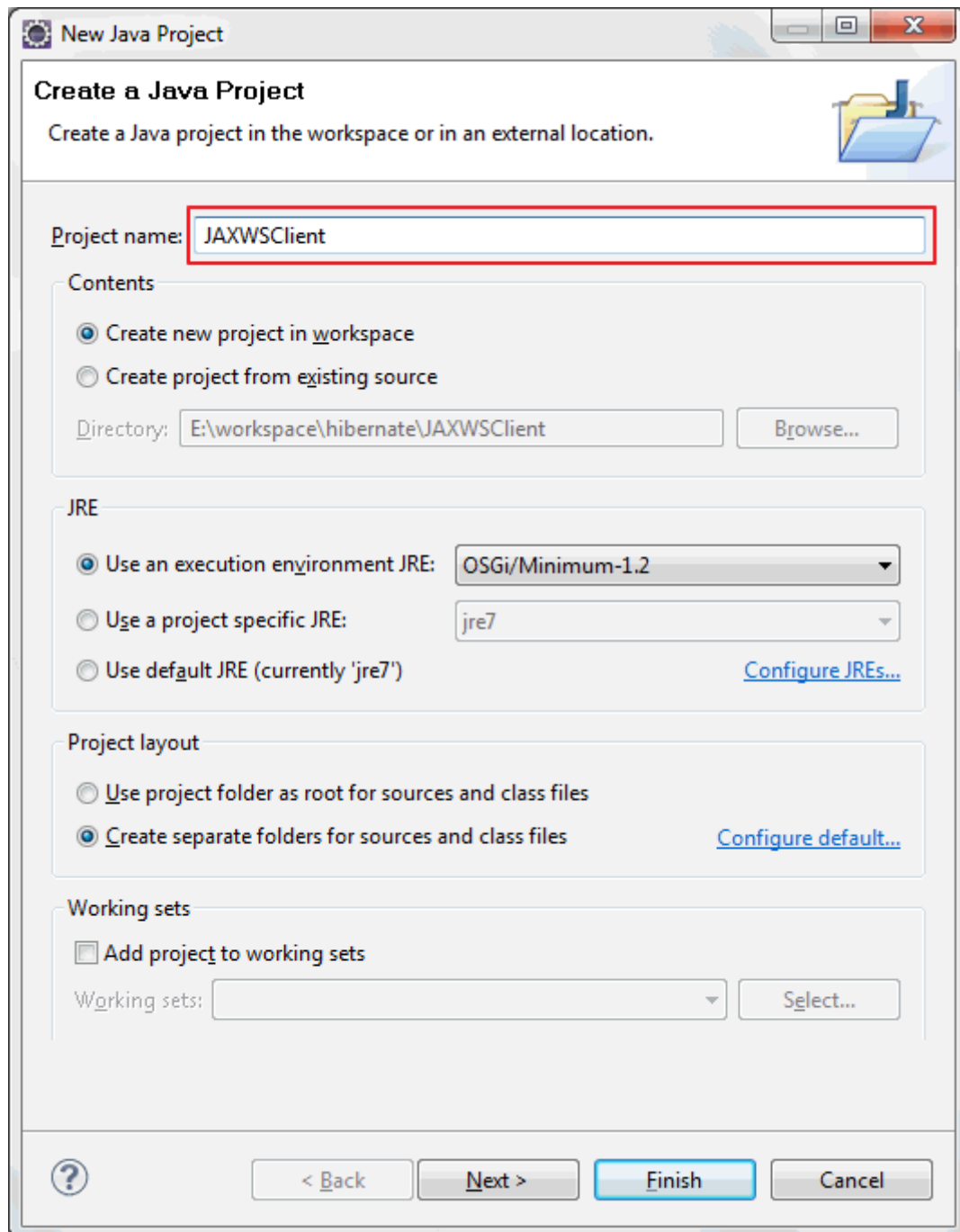
```

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

8). Można ją szybko przetestować wpisując w adresie
<http://localhost:8080/WS/HelloWorld?wsdl>

Samodzielne stworzenie aplikacji klienckiej dla udostępnionej usługi JAXWS Client

1) Stwórz nowy project w Eclipse o nazwie JAXWSClient.



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

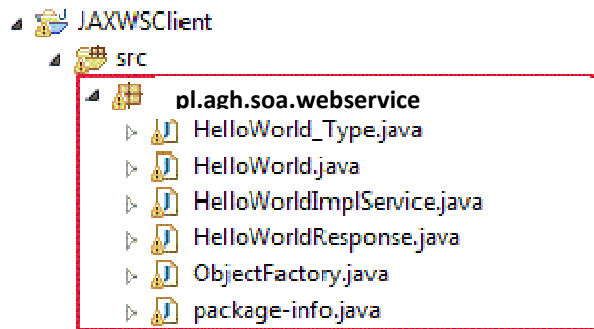
Można to zrobić np. tak:

Z linii komend użyj polecenia `wsimport`:

```
cd %project_home%/src
```

```
wsimport -s . http://localhost:8080/WS>HelloWorld?wsdl
```

w jego wyniku wygenerowane zostaną w katalogu `src` projektu następujące pliki:



4) napiszmy teraz implementacje klienta

Stwórz JAXWSCClient.java w pakiecie pl.agh.soa.webservice.client

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

import pl.agh.soa.webservice.HelloWorld;

import pl.agh.soa.webservice.webservice.HelloWorldImplService;

public class JAXWSCClient {

    public static void main(String[] args) {

        HelloWorldImplService helloWorldService = new HelloWorldImplService();

        HelloWorld helloWorld = helloWorldService.getHelloWorldImplPort();

        System.out.println(helloWorld.helloWorld("Grzegorza"));

    }

}
```

5) Uruchom aplikacje i sprawdz otrzymany wynik w konsoli tekstowej.

II. Deployment Web Serwisu za pomocą serwera aplikacyjnego

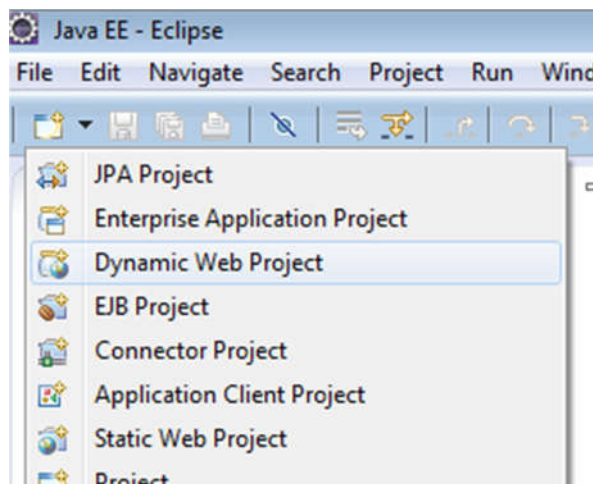
Do wykonania ćwiczenia potrzebne jest zintegrowane środowisko programistyczne Eclipse wraz z serwerem aplikacji oraz zainstalowanym i skonfigurowanym środowiskiem wykonawczym dla web serwisów. W omawianym przykładzie wykorzystano Axis2.

Web Service tworzyć można na dwa sposoby: *bottom up* (najpierw klasy Java, a potem WSDL), lub *top down* (inaczej zwane *contract first* – najpierw WSDL, a następnie implementacja).

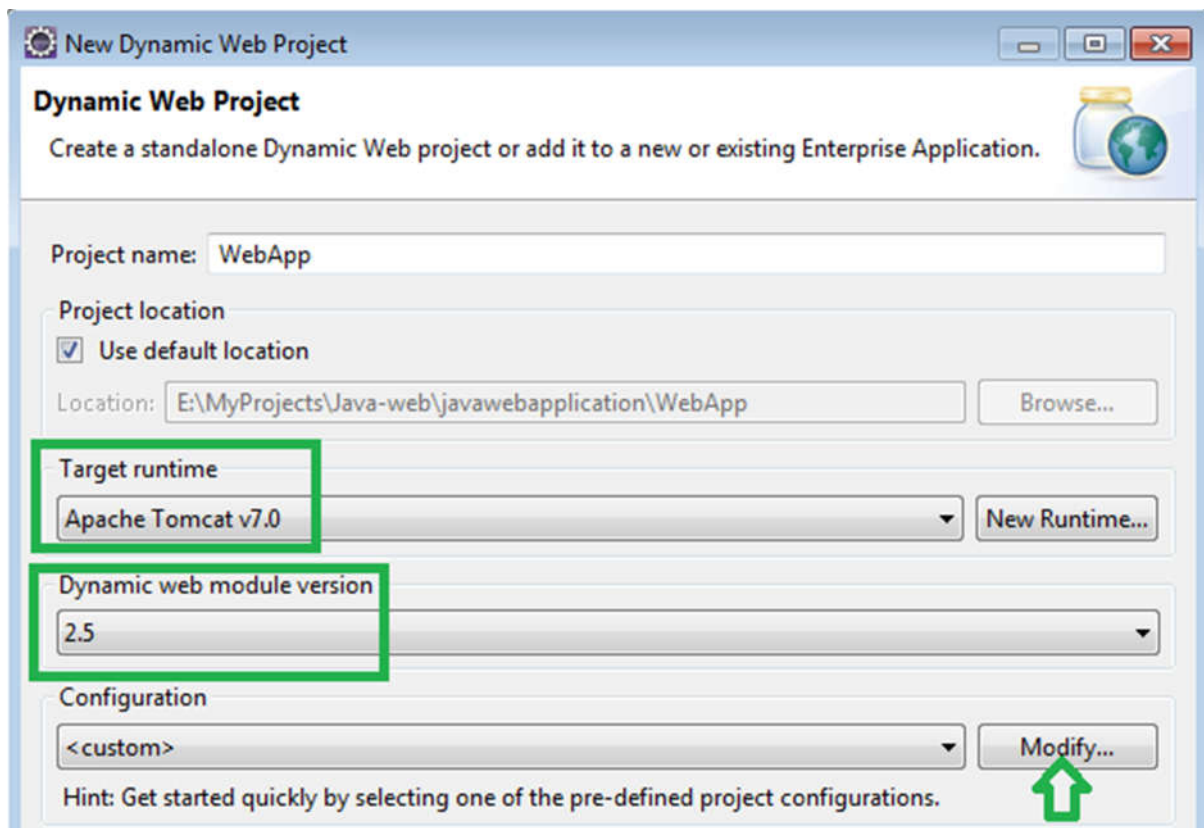
Tworzenie Web Service metodą Bottom-Up

Celem ćwiczenia jest przygotowanie usługi sieciowej w oparciu o klasę Java.

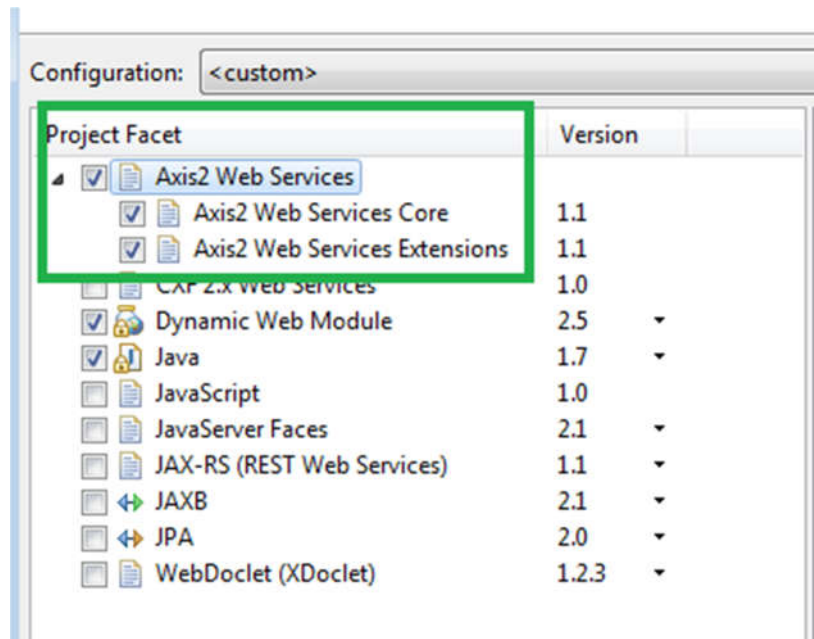
1. Uruchom środowisko Eclipse.
2. Utwórz nowy projekt typu Dynamic Web Project.



Ustaw i wypełnij okno projektowe tak jak na poniższym rysunku (proszę zmienić tylko serwer Tomcat na WildFly)



- Wybierz silnik obsługi serwisów - Axis2 web services.

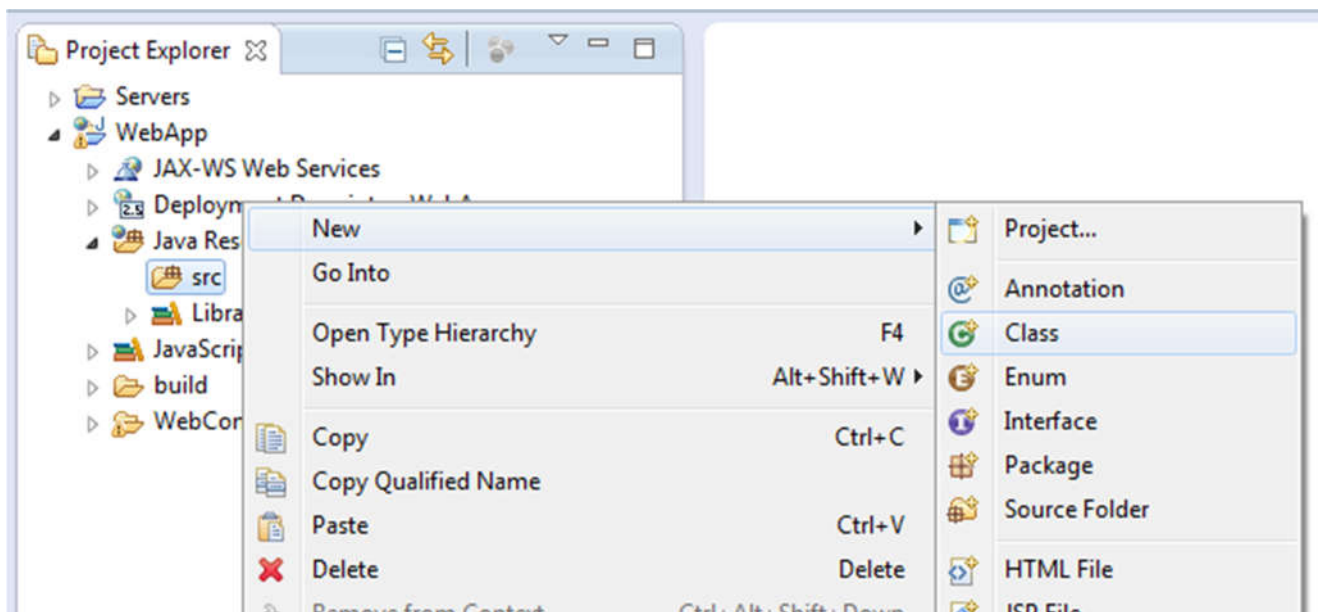


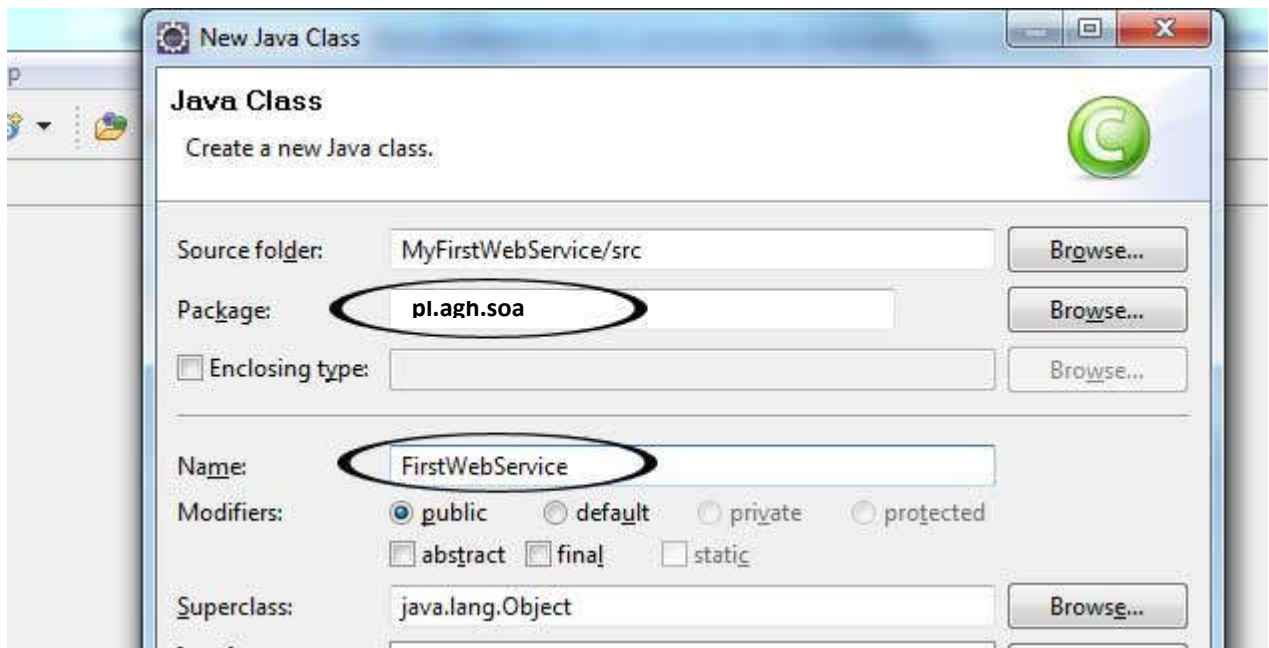
Naciśnij OK a następnie Next . Wybierz folder i zamknij kreatora.

3. Stworzenie i implementacja klasy WebSerwisu.

Teraz należy stworzyć klasę w Javie, którą chcemy udostępnić jako Web Service. Klasa nazywać się będzie FirstWebService i zawierać publiczną metodę addTwoNumbers o dwóch parametrach typu integer zwracającą wynik operacji dodawania wartości parametrów.

3.1 Prawym klawiszem myszki nacisnąć na folderze `src` i wybrać `New >> Class`





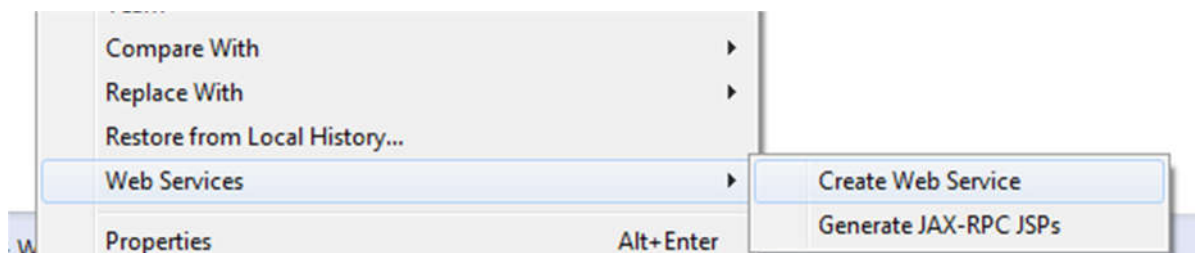
Implementacja klasy poniżej:

```
package pl.agh.soa;
public class FirstWebService {
    public int addTwoNumbers(int firstNumber, int secondNumber) {
        return firstNumber + secondNumber;
    }
}
```

3.2 Następnie dla klasy FirstWebService czyli klas, która będzie funkcjonowała jako usługa wykonaj poniższe czynności.

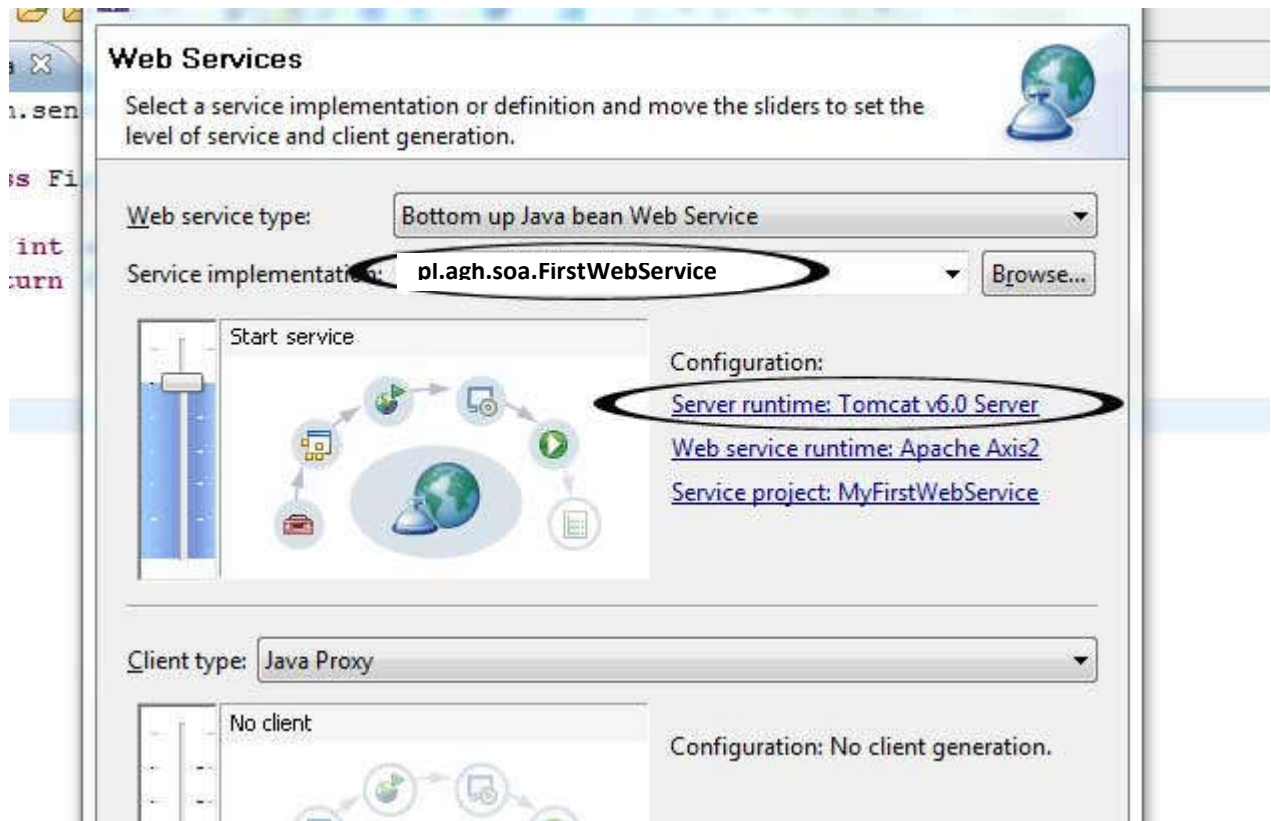
- Zaznacz klasę, która ma funkcjonować jako usługa.
- Prawym klawiszem wywołaj menu kontekstowe i wybierz:

Web Services >> Create Web Service

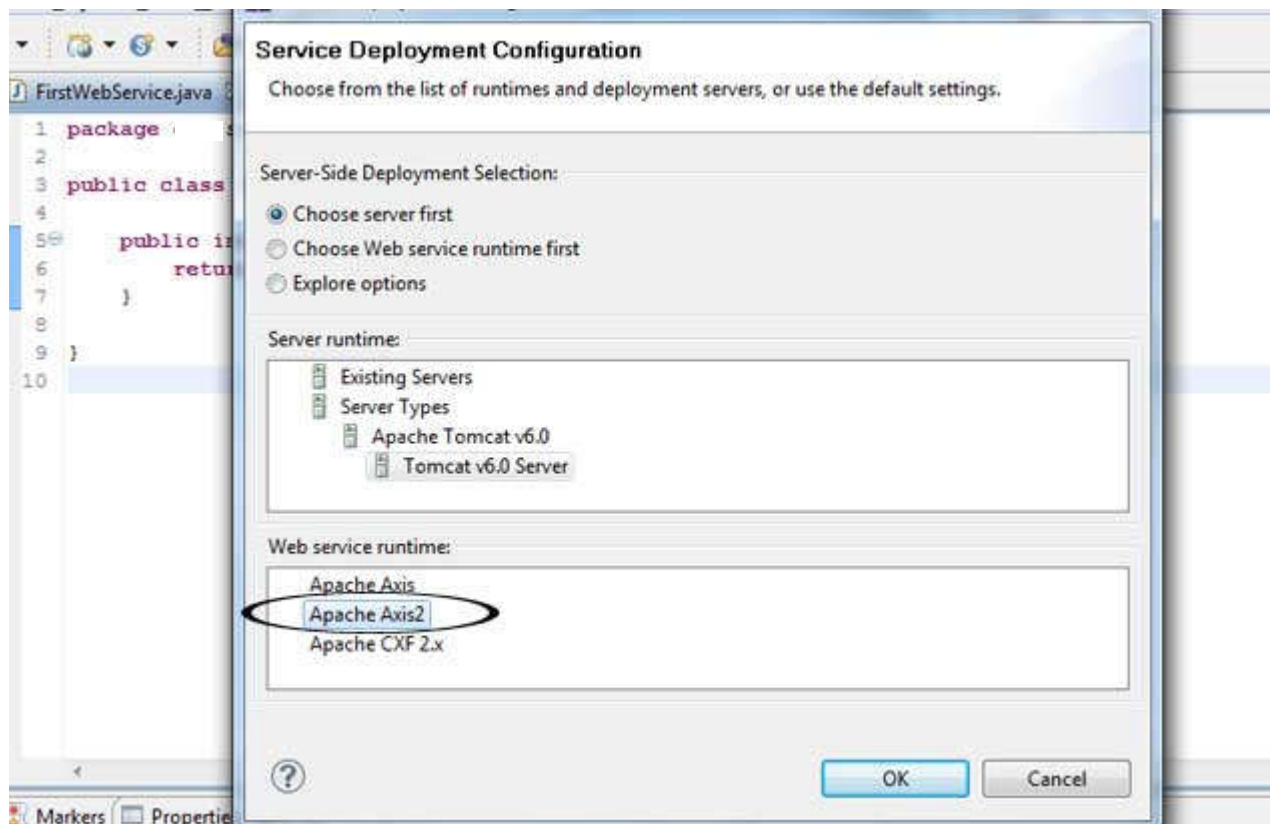


3.3 Rejestrowanie klasy jako usługi sieciowej

- Przejdź do następnego kroku przyciskiem Next. Kliknij na link umożliwiającą konfigurację środowiska uruchomieniowego dla usług.

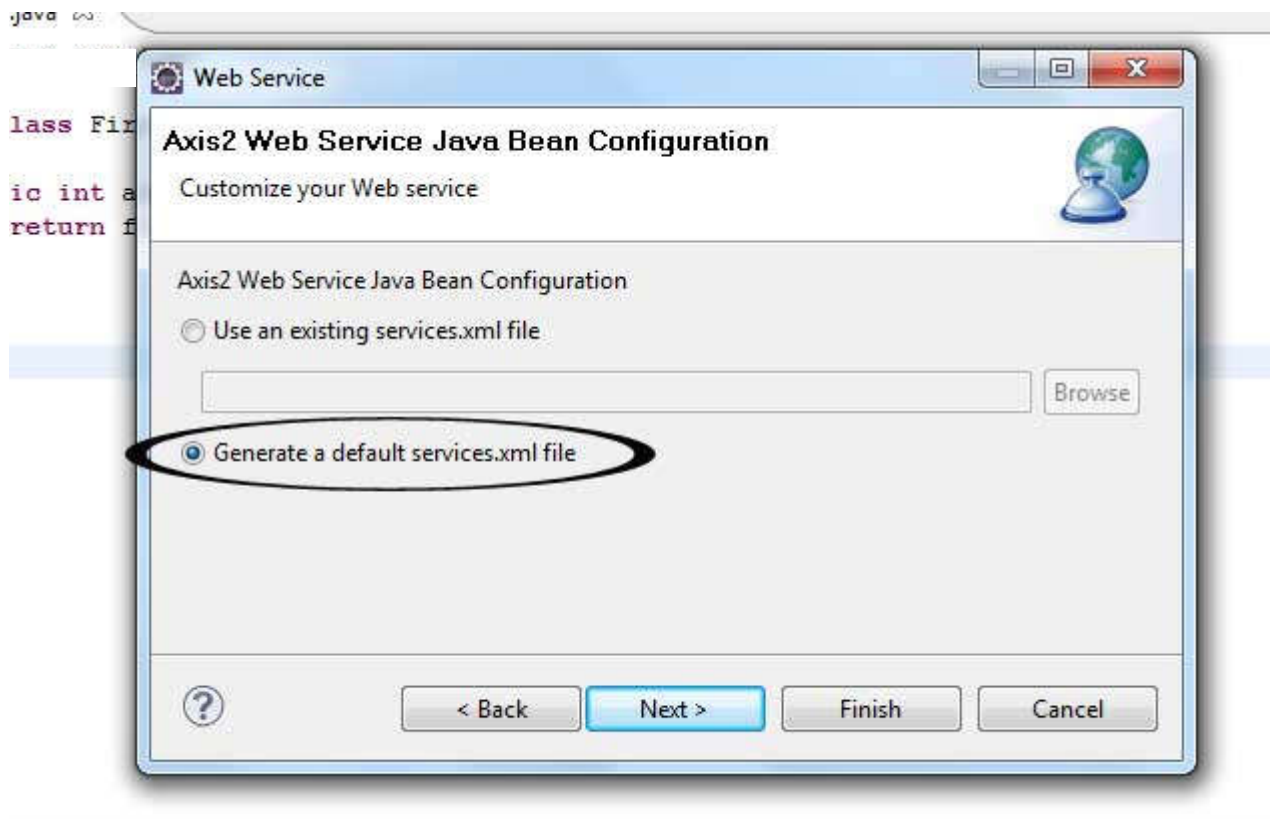


3.4 Ustawić Web Service runtime jako Axis2 i zatwierdzić Ok.



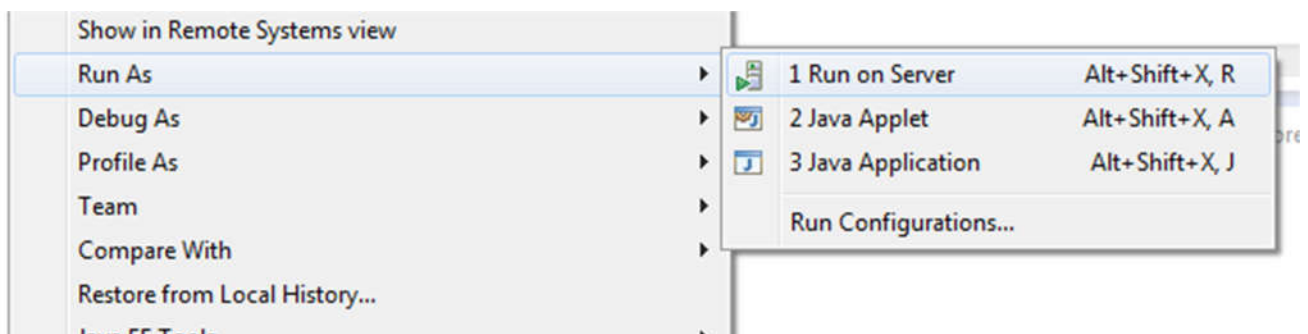
3.5 Rejestrowanie klasy jako usługi sieciowej

- Pozostaw opcję generacji domyślnego pliku services.xml.
- W kolejnym kroku uruchom serwer WildFly korzystając z przycisku *Start server*. Po zakończeniu procesu uruchamiania przejdź do następnego kroku przyciskiem *Next>*.
- Zakończ tworzenie usługi przyciskiem *Finish*. Nie publikuj usług w rejestrach UDDI.

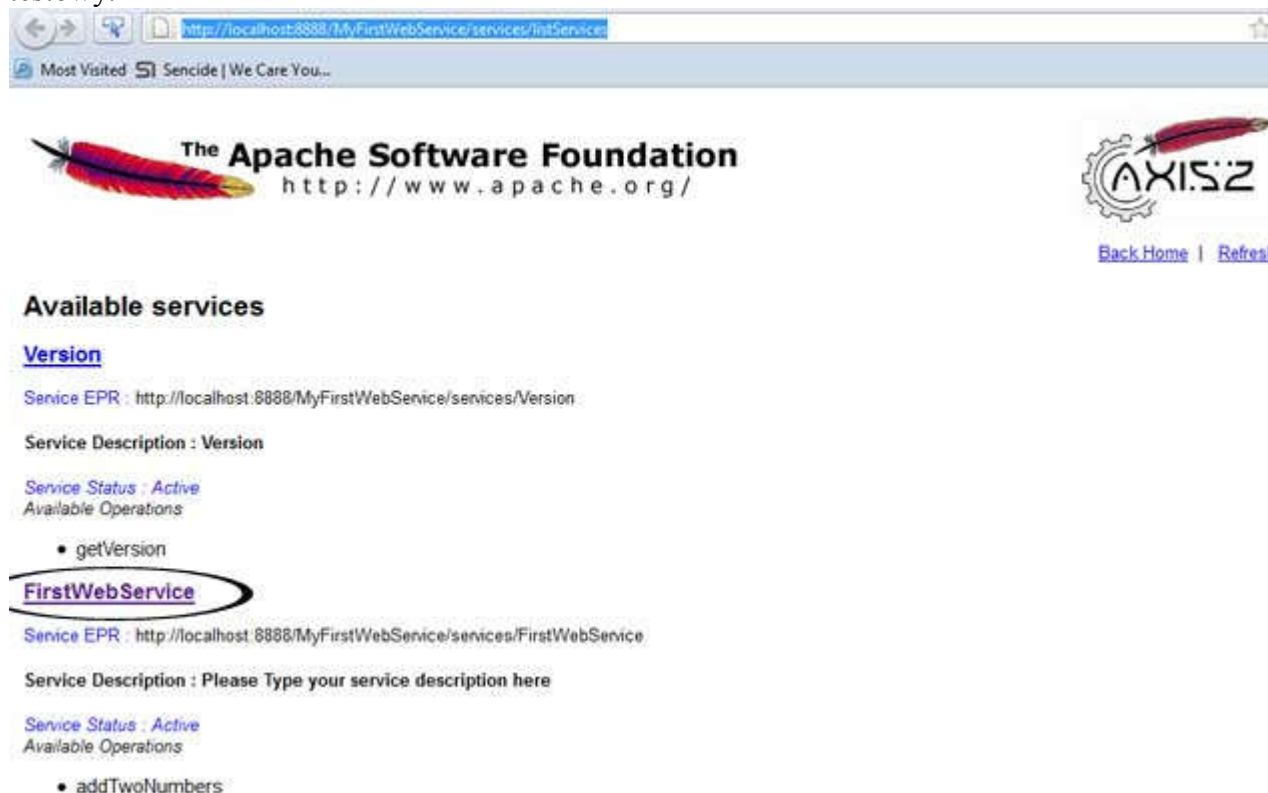


4. Testowanie Web Serwisu

Aby przetestować stworzony Web Serwis z menu kontekstowego projektu wybierz **Run As** >> **Run on Server**.

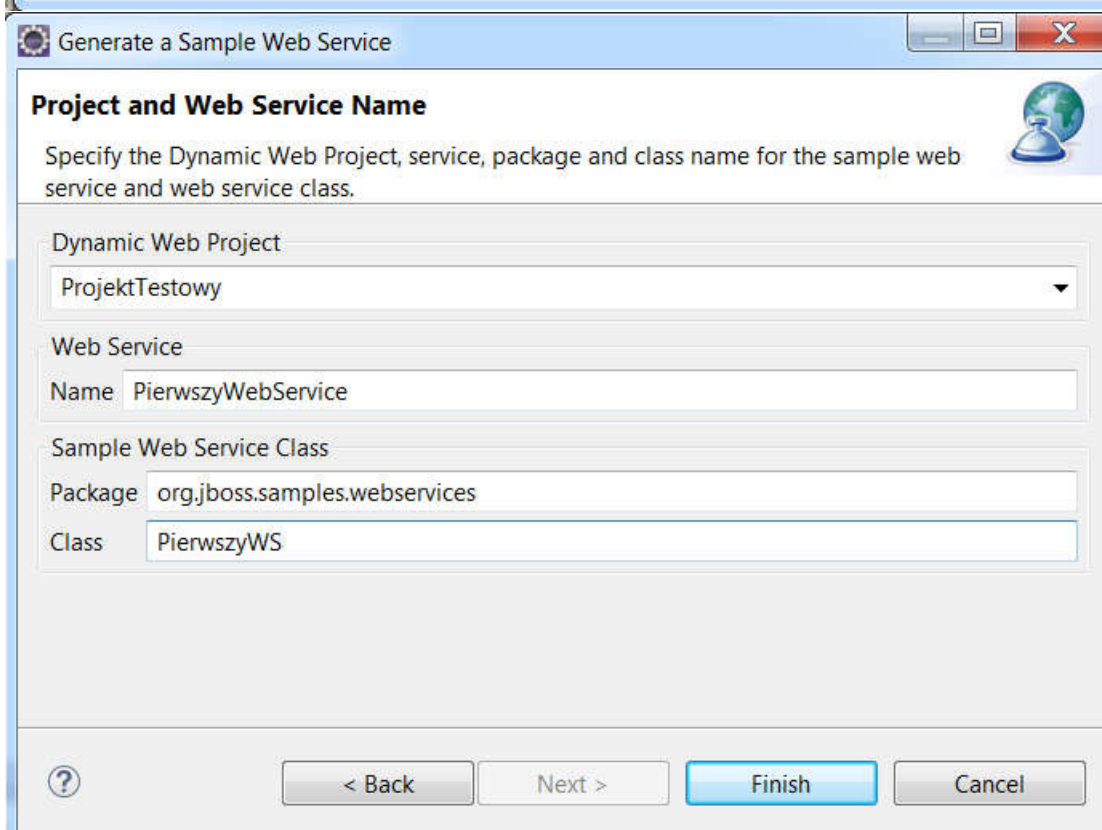
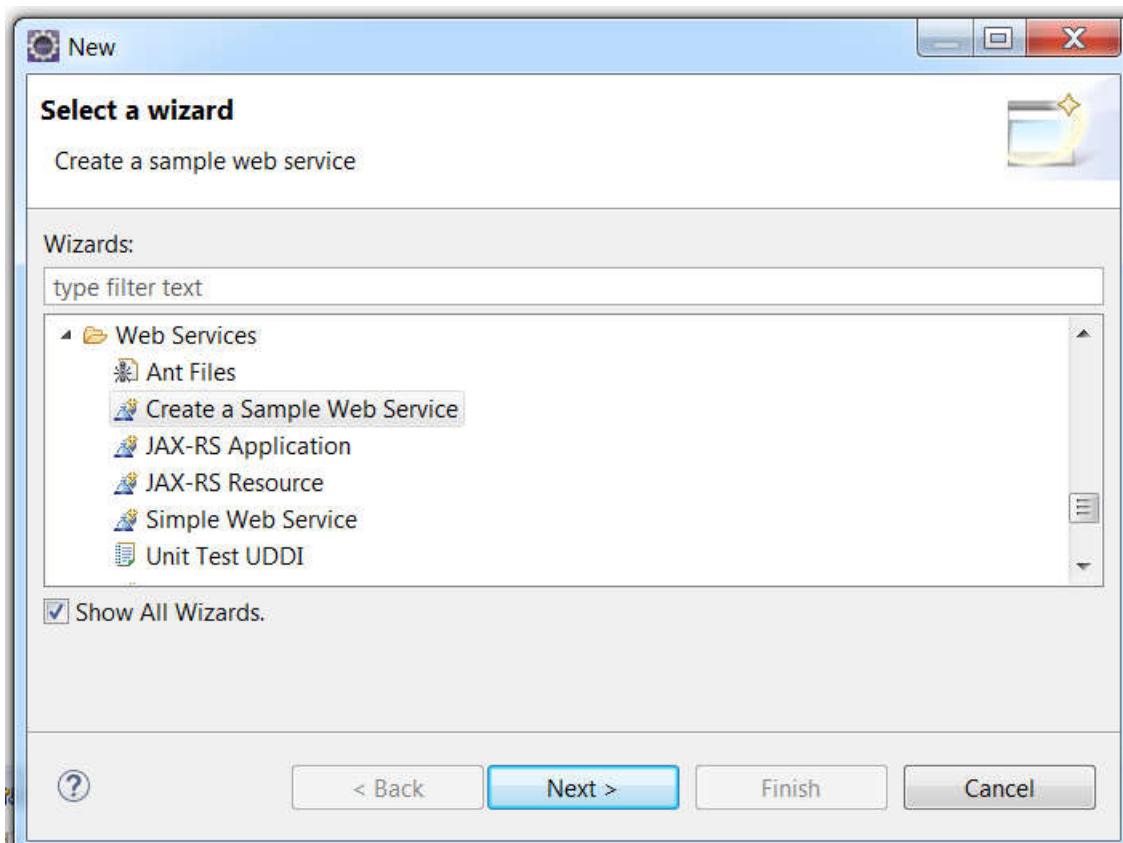


Jesli wszystko jest OK – w wbudowanej przeglądarce testowej Eclipse pojawi się klient testowy.



I. Automatyczna generacja Web Serwisu oraz Client Web Serwisowego.

(na podstawie JBoss WebService Example)



PierwszyWS.java

```
1 package org.jboss.samples.webservices;
2
3 import javax.jws.WebMethod;
4
5
6 @WebService()
7 public class PierwszyWS {
8
9     @WebMethod()
10    public String sayHello(String name) {
11        System.out.println("Hello: " + name);
12        return "Hello " + name + "!";
13    }
14 }
15
```

Project Explorer

- ProjektTestowy
 - Deployment Descriptor: ProjektTestowy
 - JAX-WS Web Services
 - Java Resources
 - src
 - org.jboss.samples.webservices
 - PierwszyWS.java
 - Libraries
 - JavaScript Resources
 - build
 - WebContent

Web Service

Web Services


Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

Service implementation: org.jboss.samples.webservices.PierwszyWS

Browse...


Start service



Configuration:
[Server runtime: WildFly 10.x](#)
[Web service runtime: Apache Axis](#)
[Service project: ProjektTestowy](#)
[Service EAR project: ProjektTestowyEAR](#)

Client type: Java Proxy

Test client



Configuration:
[Server runtime: WildFly 10.x](#)
[Web service runtime: Apache Axis](#)
[Client project: ProjektTestowyClient](#)
[Client EAR project: ProjektTestowyClientEAR](#)

☐ Publish the Web service

☒ Monitor the Web service

☒ Overwrite files without warning

?

< Back

Next >

Finish

Cancel

Web Service

Web Service Java Bean Identity

Configure the Java bean as a Web service.

WSDL file: PierwszyWS.wsdl

Methods

☒ sayHello(java.lang.String)

Select All

Deselect All

Style and use

☒ document/literal (wrapped)

☐ document/literal

☐ RPC/encoded

☐ Define custom mapping for package to namespace.

?

< Back

Next >

Finish

Cancel

Web Service

Web Service Client Test

Do you want to test the generated proxy?

☒ Test the generated proxy

Test facility: JAX-RPC JSPs

JSP project: ProjektTestowyClient

EAR project: ProjektTestowyClientEAR

Folder: samplePierwszyWSProxy Browse...

JSP folder: /ProjektTestowyClient/WebContent/samplePierwszyWSProxy

Methods

- ☒ getEndpoint()
- ☒ setEndpoint(java.lang.String)
- ☒ getPierwszyWS()
- ☒ sayHello(java.lang.String)

Select All Deselect All

☒ Run test on server

? < Back Next > Finish Cancel

Web Services Test Client

localhost:8080/ProjektTestowyClient/samplePierwszyWSProxy/TestClient.jsp?endpoint=http://localhost:11270/ProjektTestowy/services/PierwszyWS

Methods

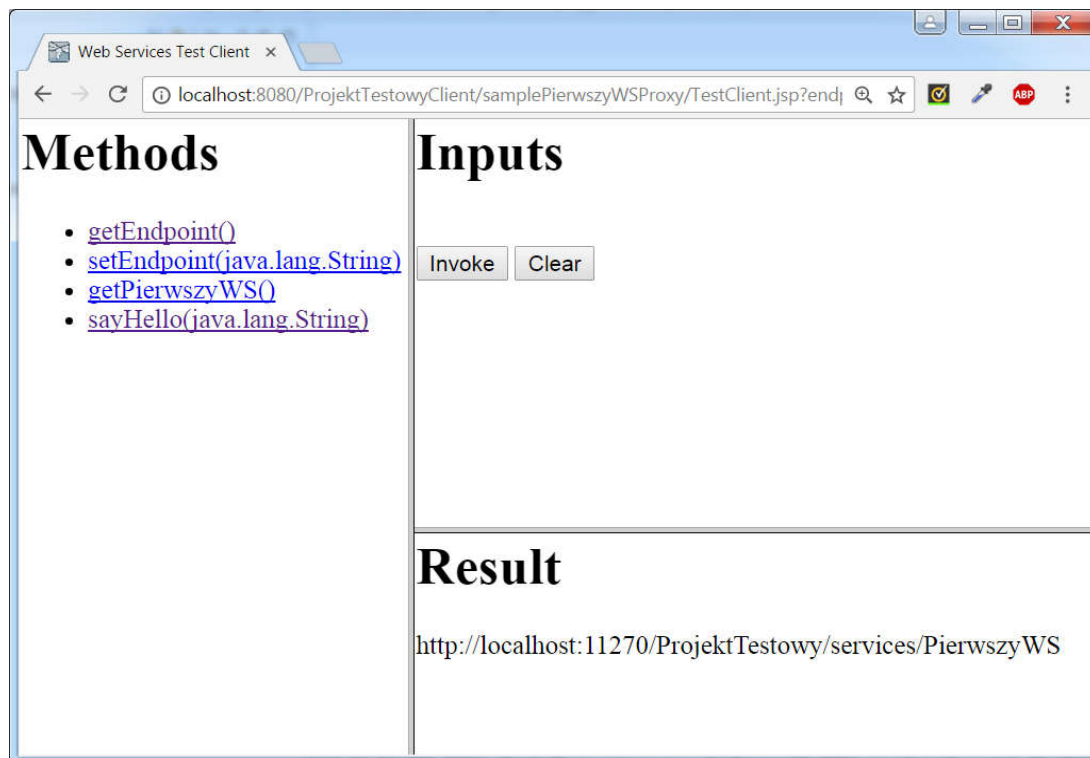
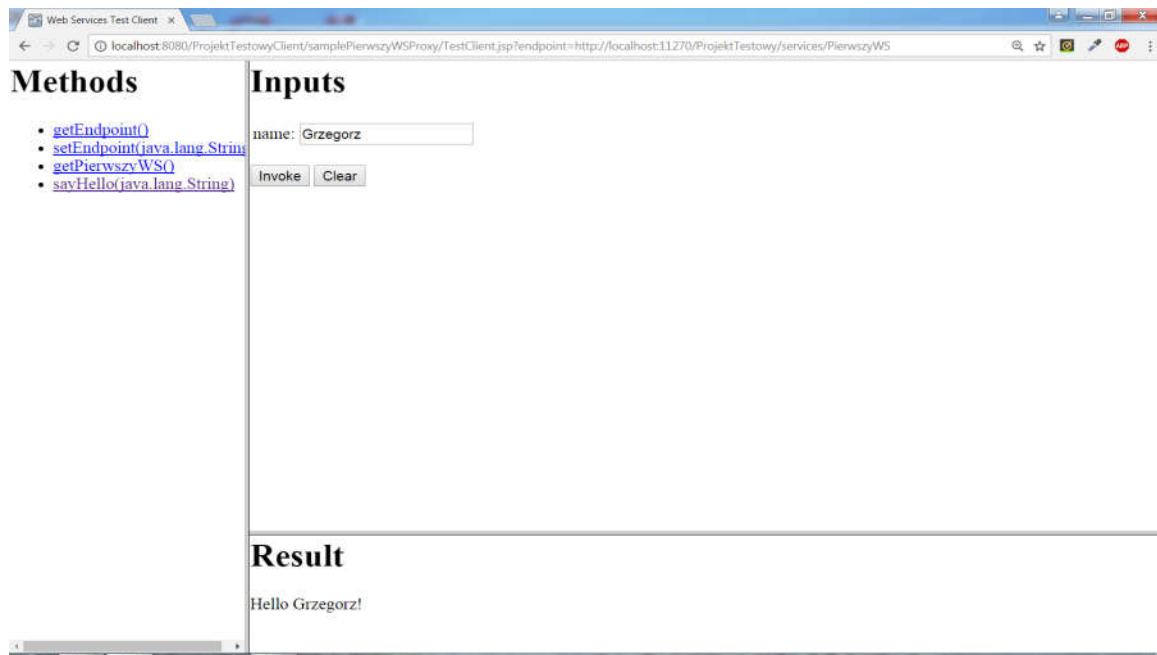
- [getEndpoint\(\)](#)
- [setEndpoint\(java.lang.String\)](#)
- [getPierwszyWS\(\)](#)
- [sayHello\(java.lang.String\)](#)

Inputs

Select a method to test.

Result

result: N/A



Do przeciwiczenia na zajęciach są następujące zagadnienia:

Zadanie 1.

1. Tworzenie prostego Web Serwisu 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 Serwisu, 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.

Gdyby były ciągle problemy z realizacją zadań jako dodatkowe wsparcie proponuje zapoznać się z:

W celu ułatwienia realizacji zadań proponuje zaznajomić się z następującymi przykładami:

Opis realizacji Web Serwisów w JBoss :

http://docs.jboss.org/tools/latest/en/ws_soap_reference/html/

Opis realizacji Web Serwisów w Eclipse możecie państwo znaleźć pod poniższymi adresami;

http://www.eclipse.org/webtools/community/tutorials/BottomUpAxis2WebService/bu_tutorial.html

<http://www.eclipse.org/webtools/jst/components/ws/1.5/tutorials/BottomUpWebService/BottomUpWebService.html>

<http://www.eclipse.org/webtools/jst/components/ws/1.5/tutorials/TopDownWebService/TopDownWebService.html>

<http://www.eclipse.org/webtools/jst/components/ws/1.5/tutorials/WebServiceClient/WebServiceClient.html>

<http://www.eclipse.org/webtools/jst/components/ws/1.0/tutorials/WebServiceExplorer/WebServiceExplorer.html>

Dodatkowo ciekawe tutoriale na temat tworzenia Web Serwisów można znaleźć tutaj:

<http://www.mkyong.com/tutorials/jax-ws-tutorials/>