

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/jaxws001.htm#BNAYN>

Do przećwiczenia 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.

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.

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/

Lub tutaj

<http://www.mastertheboss.com/javaee/jboss-web-services>

Przykład realizacji prostego Web Serwisu w JBoss można przeglądać także tutaj;

https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Developer_Studio/7.0/html/JBoss_SOAP_Web_Services_User_Guide/simple_web_service.html

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

<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>

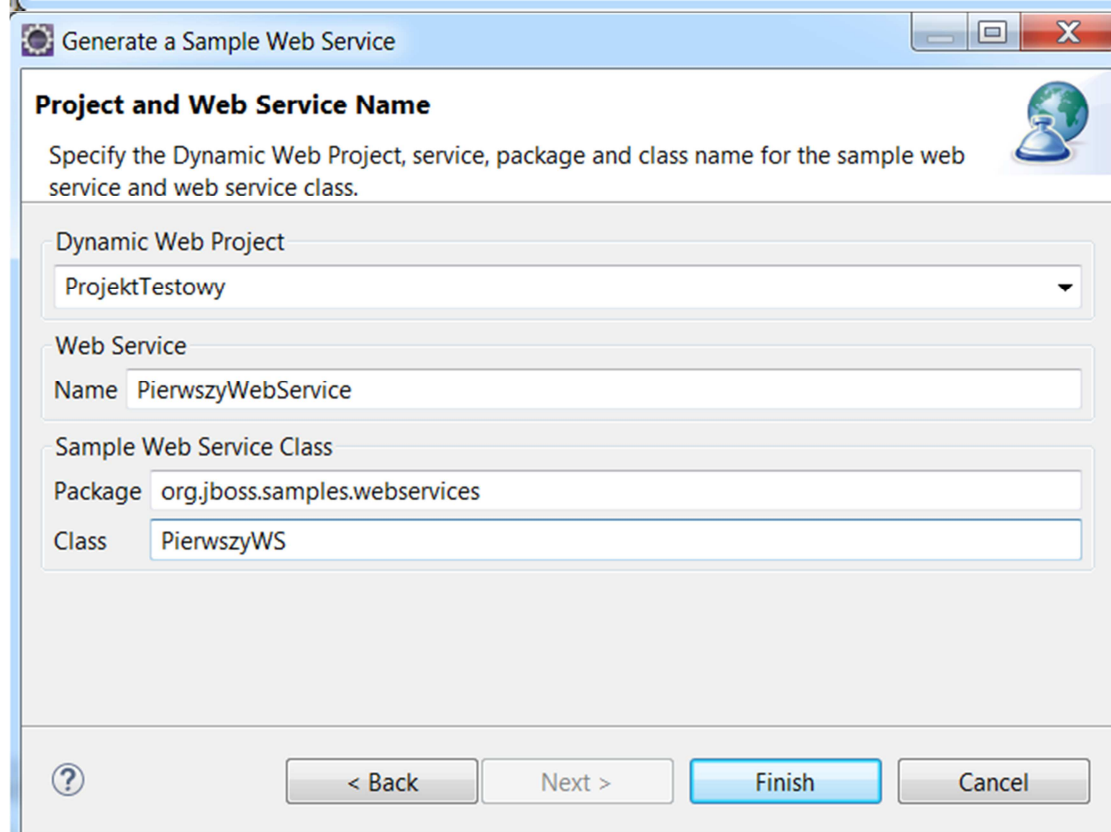
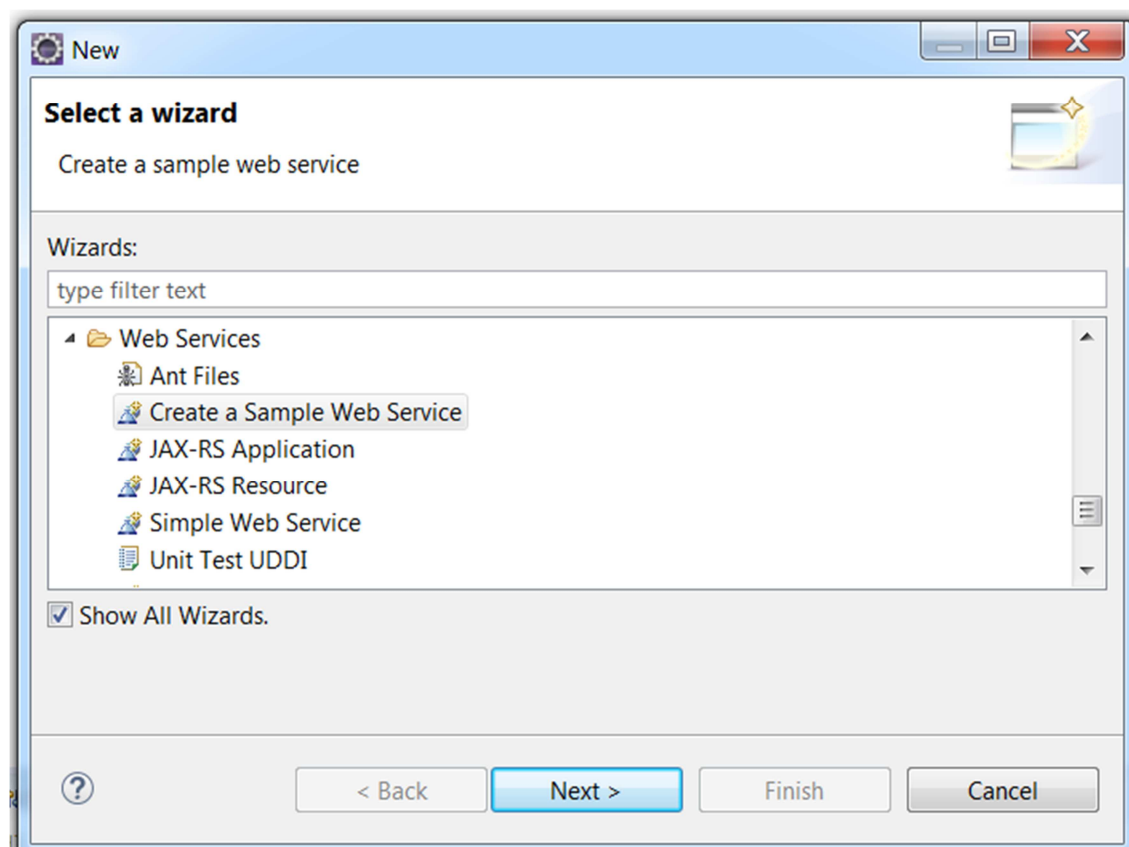
<http://www.eclipse.org/webtools/jst/components/ws/1.0/tutorials/WebServiceExplorer/WebServiceExplorer.html>

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

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

<http://javapostsforlearning.blogspot.in/2013/03/jaxws-web-service-eclipse-tutorial.html>


Krótki tutorial zdjęciowy jak tworzyć WS i go testować.




```
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: samplePierwszyWSPProxy Browse...

JSP folder: /ProjektTestowyClient/WebContent/samplePierwszyWSPProxy

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/samplePierwszyWSPProxy/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

