Rozproszone systemy internetowe

Web serwisy
JAX-WS (SOAP web services)

cz.2

Sources and help:

Web services (javaTpoint)

Java Web Services - Book online

JAX-WS Tutorial by mkyong

Java EE Tutorial -Web services

JAX-WS Release Documentation

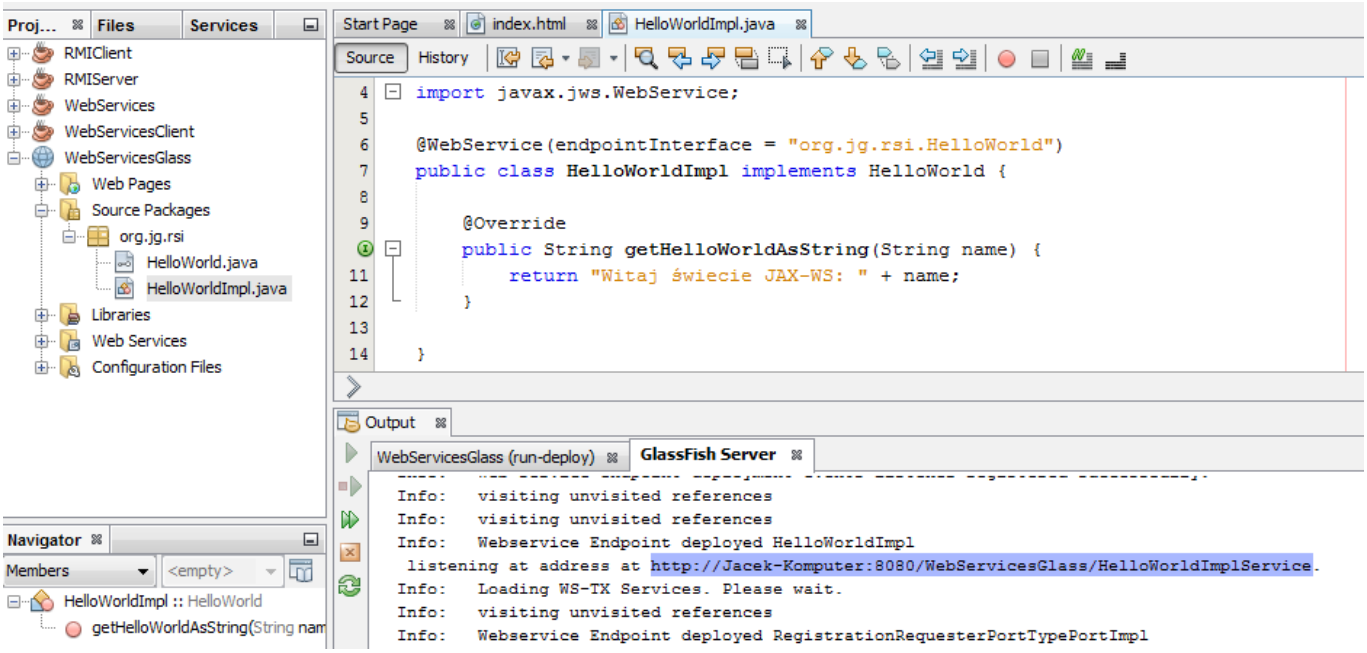
Metro User Guide

Ćwiczenie 1. Tworzenie web serwisu na serwerze Glassfish

Utwórz Web Aplication Project

Skopiuj do projektu pliki interfejsu i implementacji z poprzednich zajęć(HelloWorld.java, HelloWorldImpl.java)

Wdróż (Deploy) projekt na serwerze Glassfish



Sprawdź czy serwis działa.

http://localhost:8080/WebServicesGlass/HelloWorldImplService

http://localhost:8080/WebServicesGlass/HelloWorldImplService?WSDL

Web Services

Endpoint	Information
Service Name: {http://rsi.jg.org/}HelloWorldImplService Port Name: {http://rsi.jg.org/}HelloWorldImplPort	Address: http://localhost:8080/WebServicesGlass/HelloWorldImplService WSDL: http://localhost:8080/WebServicesGlass/HelloWorldImplService?wsdl Implementation class: org.jg.rsi.HelloWorldImpl

ZADANIA:

- Pod jaką nazwą został opublikowany serwis?
- Gdzie jest ta nazwa zdefiniowana w WSDL?
- Czym się różni WSDL serwisu wdrożonego na Glassfish od WSDL serwisu z poprzednich zajęć?
- Wywołaj „glassfishowego” testera serwisów http://localhost:8080/WebServicesGlass/HelloWorldImplService?Tester

getHelloWorldAsString Method invocation

Method parameter(s)

Type	Value
java.lang.String	test

Method returned

java.lang.String : "**Witaj świecie JAX-WS: test**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="ht
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:getHelloWorldAsString xmlns:ns2="http://rsi.jg.org/">
            <arg0>test</arg0>
        </ns2:getHelloWorldAsString>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="ht
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:getHelloWorldAsStringResponse xmlns:ns2="http://rsi.jg.org/">
            <return>Witaj Ł›wiecie JAX-WS: test</return>
        </ns2:getHelloWorldAsStringResponse>
    </S:Body>
</S:Envelope>
```

Ćwiczenie 2. Tworzenie aplikacji klienckiej dla zewnętrznego web serwisu

Napisz aplikację która skorzysta z **zewnętrznego serwisu** i pobierze nazwę kraju na podstawie wysłanego IP

GeoIPService enables you to easily look up countries by IP address / Context

WSDL serwisu:

http://wsgeoip.lavasoft.com/ipservice.asmx?WSDL

Strona info:

http://wsgeoip.lavasoft.com/ipservice.asmx

PS.

Można wybrać sobie inny serwis z tej strony np.

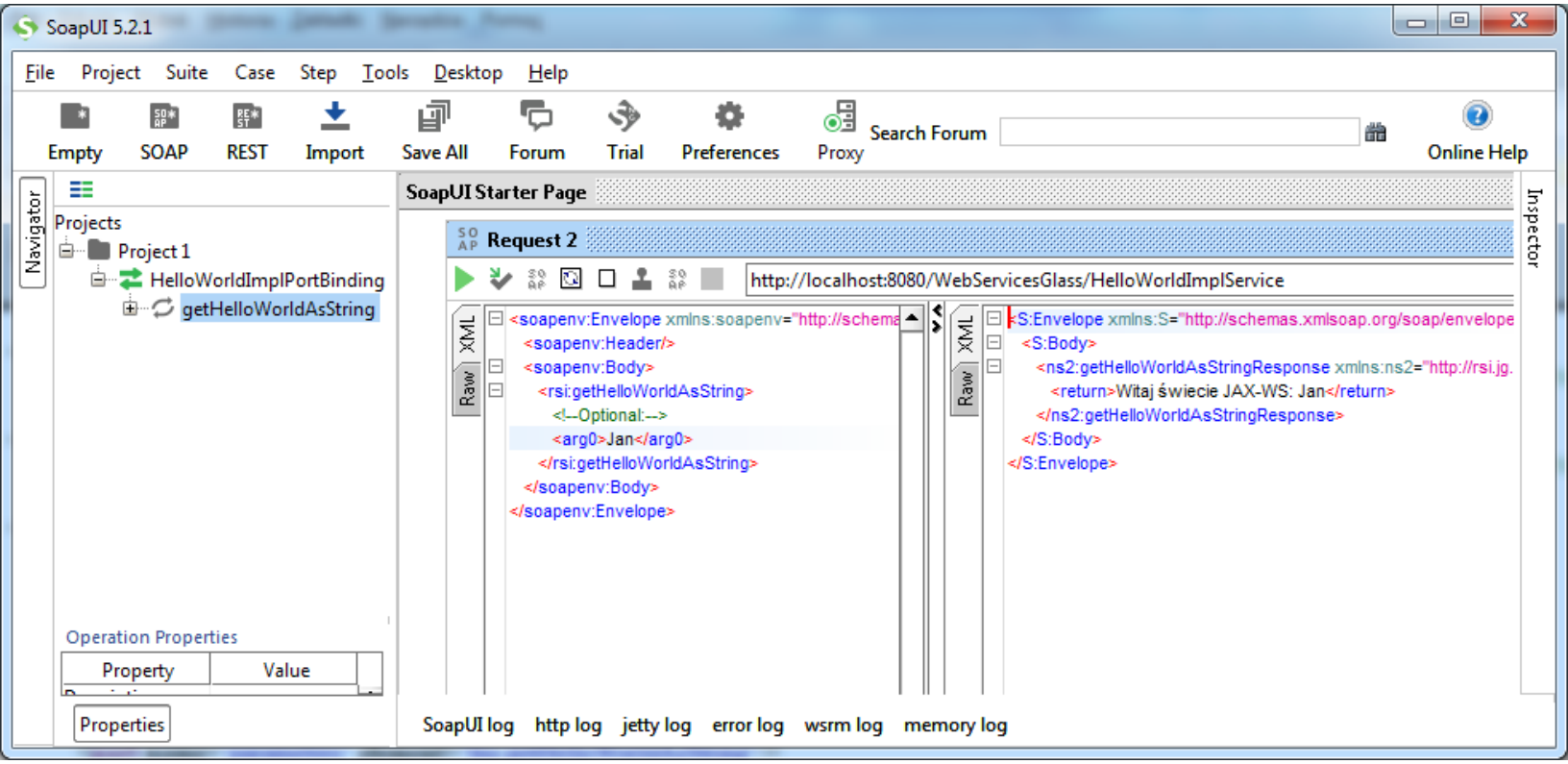
http://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSDL

http://www.xignite.com/xcurrencies.asmx?WSDL

Ćwiczenie 3. Użycie SoapUI do testowanie serwisów

Użyj **SoapUI** do testowanie serwisu (sugeruję zainstalowanie **wersji 4.6.4,** bo w nowszej HTTP monitor nie działa)

https://www.soapui.org/downloads/soapui/soapui-os-older-versions.html



Ćwiczenie 4. Rozbudowa web serwisu

Utwórz metodę w Web Serwisie, która zwróci listę produktów. Przetestuj serwis przy użyciu SoapUI. Przeanalizuj SOAP komunikaty. Przeanalizuj WSDL.

np.

…

```
@WebMethod

public List<Produkt> getProducts(){

}
```

```
public class Product(){

private String nazwa;

private String opis;

private int cena;
```

```
...

}
```

Ćwiczenie 5. Testowanie wpływu adnotacji na WSDL

- Przetestuj kolejno wpływ parametrów adnotacji @WebService na generowany WSDL

```
@WebService (name = "..." )

portName = "...", serviceName = "...", targetNamespace = "…"
```

- Przetestuj kolejno wpływ parametrów adnotacji @WebMethod na generowany WSDL

```
@WebMethod (action = "…")

, operationName = "...", exclude =
```

Ćwiczenie 6. Tworzenie aplikacji klienckiej w innym języku niż web serwis

Zaimplementować **klienta** z innym języku niż język w którym napisany został web serwis(np. 1) web serwis: Java, Klient: JavaScript, Python, C lub C#, Albo 2) web servis: C#, Klient: Java, itp) i połączyć się z utworzonym **web serwisem.**