Rozproszone Systemy Informatyczne

Raport - Ćwiczenie 3

Paweł Kluska, 260391

Katsiaryna Ziatsikava, 245891

Stworzyliśmy Java Maven project, w pliku pom dodaliśmy zależność jaxws-rt z com.sun.xml.ws.

```xml
<dependencies>
    <dependency>
        <groupId>com.sun.xml.ws</groupId>
        <artifactId>jaxws-rt</artifactId>
        <version>3.0.2</version>
    </dependency>
</dependencies>
```

Została zaimplementowana klasa **Person**, w której zdefiniowaliśmy atrybuty klasy, konsruktory, gettery i settery.

```java
public class Person {
    4 usages
    private int id;
    4 usages
    private String firstName;
    4 usages
    private int age;

    public Person() {
    }
    public Person(int id, String firstName, int age) {
        this.id = id;
        this.firstName = firstName;
        this.age = age;
    }
    4 usages
    public int getId() { return id; }
    no usages
    public void setId(int id) { this.id = id; }
    no usages
    public String getFirstName() { return firstName; }
    1 usage
    public void setFirstName(String firstName) { this.firstName = firstName; }
    no usages
    public int getAge() { return age; }
    1 usage
    public void setAge(int age) { this.age = age; }
```

Zdefiniowaliśmy klasy wyjątki **PersonExistsEx**, **PersonNotFoundEx** z adnotacją @WebFault, która została zaimportowana z pakietu jakarta.xml.ws.

```java
import jakarta.xml.ws.WebFault;

7 usages
@WebFault
public class PersonExistsEx extends Exception {
    9 usages
    public PersonExistsEx() {
        super("This person already exists");
    }
}
```

```java
import jakarta.xml.ws.WebFault;

15 usages
@WebFault
public class PersonNotFoundEx extends Exception {
    14 usages
    public PersonNotFoundEx() {
        super("The specified person does not exist");
    }
}
```

Został zdefiniowany interfejs **PersonRepositoty,** który zawiera opercje CRUD, plus aktualny rozmiar arraylisty oraz wykaz zawartości całej arraylisty.

```java
import java.util.List;

2 usages  1 implementation
public interface PersonRepository {
    1 usage  1 implementation
    List<Person> getAllPersons();
    1 usage  1 implementation
    Person getPerson(int id) throws PersonNotFoundEx;
    1 usage  1 implementation
    Person updatePerson(int id, String name, int age) throws PersonNotFoundEx;
    1 usage  1 implementation
    boolean deletePerson(int id) throws PersonNotFoundEx;
    1 usage  1 implementation
    Person addPerson(int id, String name, int age) throws PersonExistsEx;
    1 usage  1 implementation
    int countPersons();
}
```

Została zdefiniowana klasa **PersonRepositoryImpl** implementująca interfejs **PersonRepository.**

```
import java.util.ArrayList;
import java.util.List;
1 usage
public class PersonRepositoryImpl implements PersonRepository {
    11 usages
    private List<Person> personList;
    1 usage
    public PersonRepositoryImpl() {
        personList = new ArrayList<>();
        personList.add(new Person( id: 1,  firstName: "Mariusz",  age: 9));
        personList.add(new Person( id: 2,  firstName: "Andrzej",  age: 10));
    }
    1 usage
    public List<Person> getAllPersons() {
        return personList;
    }
    1 usage
    public Person getPerson(int id) throws PersonNotFoundEx {
        for (Person thePerson : personList) {
            if (thePerson.getId() == id) {
                return thePerson;
            }
        }
        throw new PersonNotFoundEx();
    }
```

```
public Person addPerson(int id, String name, int age) throws PersonExistsEx {
    for (Person thePerson : personList) {
        if (thePerson.getId() == id) {
            throw new PersonExistsEx();
        }
    }
    Person person = new Person(id, name, age);
    try {
        Thread.sleep( millis: 5000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    personList.add(person);
    return person;
}
1 usage
public boolean deletePerson(int id) throws PersonNotFoundEx {
    for (Person person : personList) {
        if (person.getId() == id) {
            personList.remove(person);
            return true;
        }
    }
    throw new PersonNotFoundEx();
}
```

```
    public Person updatePerson(int id, String name, int age)
            throws PersonNotFoundEx {
        for (Person person : personList) {
            if (person.getId() == id) {
                person.setFirstName(name);
                person.setAge(age);
                return person;
            }
        }
        throw new PersonNotFoundEx();
    }
    1 usage
    public int countPersons() {
        return personList.size();
    }
}
```

Web Service

Został zdefiniowany interfejs usługi sieciowej **PersonService**. Który został adnotowany jako @WebService, a metody jako @WebMethod.

```
import jakarta.jws.WebMethod;
import jakarta.jws.WebService;
import java.util.List;
1 usage  1 implementation
@WebService
public interface PersonService {
    no usages  1 implementation
    @WebMethod
    Person getPerson(int id) throws PersonNotFoundEx;
    no usages  1 implementation
    @WebMethod
    Person addPerson(int id, String name, int age) throws PersonExistsEx;
    no usages  1 implementation
    @WebMethod
    boolean deletePerson(int id) throws PersonNotFoundEx;
    no usages  1 implementation
    @WebMethod
    Person updatePerson(int id, String name, int age) throws PersonExistsEx, PersonNotFoundEx;
    no usages  1 implementation
    @WebMethod
    List<Person> getAllPersons();
    no usages  1 implementation
    @WebMethod
    int countPersons();
}
```

Została stworzona klasa **PersonServiceImpl** implementująca interfejs **PersonService**. W adnotacji @WebService zostały dodane atrybuty serviceName oraz endpoIntInterface. W klasie zostało zainicjalizowane repozytorium danych, a w metodach wywołanie metod z repozytorium.

```java
import jakarta.jws.WebMethod;
import jakarta.jws.WebService;

import java.util.List;

2 usages
@WebService(serviceName = "PersonService",
        endpointInterface = "org.example.jaxws.server.PersonService")
public class PersonServiceImpl implements PersonService {
    6 usages
    private final PersonRepository dataRepository = new PersonRepositoryImpl();
    no usages
    @WebMethod
    public Person getPerson(int id) throws PersonNotFoundEx {
        System.out.println("...called getPerson id=" + id);
        return dataRepository.getPerson(id);
    }
    no usages
    @WebMethod
    public List<Person> getAllPersons() {
        System.out.println("...called getAllPersons");
        return dataRepository.getAllPersons();
    }
```

```java
    @WebMethod
    public Person addPerson(int id, String name, int age) throws PersonExistsEx {
        return dataRepository.addPerson(id, name, age);
    }
    no usages
    @WebMethod
    public boolean deletePerson(int id) throws PersonNotFoundEx {
        return dataRepository.deletePerson(id);
    }
    no usages
    @WebMethod
    public Person updatePerson(int id, String name, int age) throws PersonExistsEx, PersonNotFoundEx {
        return dataRepository.updatePerson(id, name, age);
    }
    no usages
    @WebMethod
    public int countPersons() {
        return dataRepository.countPersons();
    }
}
```

Zdefiniowaliśmy klasę **ServiceHost**, która hostuje i uruchamia usługę sieci Web. W pierszej linijce metody main jest wywołana metoda MyDate.info, która została zaimportowana z pliku jar.

```
import com.example.MyDate;
import jakarta.xml.ws.Endpoint;
import java.io.IOException;

import static java.lang.System.exit;

no usages
public class ServiceHost {
    no usages
    public static void main(String[] args) {
        MyDate.info();
        System.out.println("Web Service org.example.jaxws.server.PersonService is running ...");
        PersonServiceImpl psi = new PersonServiceImpl();
        Endpoint.publish( address: "http://localhost:8081/personservice", psi);
        System.out.println("Press ENTER to STOP org.example.jaxws.server.PersonService ...");
        try {
            System.in.read();
        } catch (IOException e) {
            e.printStackTrace();
        }
        exit( status: 0);
    }
}
```

Dla tworzenia klienta użyliśmy metodę Top-down. Odpowienio przygotowaliśmy plik **personservice.wsdl** oraz dodaliśmy plagin com.sun.xml.ws/**jaxws-maven-plugin** z **wsimport** do pliku pom. Zostały wygenerowane klasy na podstawie pliku **personservice.wsdl** do pakietu org.example.jaxws.server_topdown.

Plik personservice.wsdl wygląda następująco:

```
<definitions
targetNamespace="http://server.jaxws.example.org/"
             xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:tns="http://server.jaxws.example.org/"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             name="PersonService">
   <types>
                                    <xs:schema     version="1.0"
targetNamespace="http://server.jaxws.example.org/">
                               <xs:element   name="PersonExistsEx"
type="tns:PersonExistsEx"/>
                             <xs:element   name="PersonNotFoundEx"
type="tns:PersonNotFoundEx"/>
           <xs:element name="addPerson" type="tns:addPerson"/>
```

```
                        <xs:element  name="addPersonResponse"
type="tns:addPersonResponse"/>
                            <xs:element  name="countPersons"
type="tns:countPersons"/>
                        <xs:element  name="countPersonsResponse"
type="tns:countPersonsResponse"/>
                            <xs:element  name="deletePerson"
type="tns:deletePerson"/>
                        <xs:element  name="deletePersonResponse"
type="tns:deletePersonResponse"/>
                            <xs:element  name="getAllPersons"
type="tns:getAllPersons"/>
                        <xs:element  name="getAllPersonsResponse"
type="tns:getAllPersonsResponse"/>
            <xs:element name="getPerson" type="tns:getPerson"/>
                            <xs:element  name="getPersonResponse"
type="tns:getPersonResponse"/>
                            <xs:element  name="updatePerson"
type="tns:updatePerson"/>
                        <xs:element  name="updatePersonResponse"
type="tns:updatePersonResponse"/>
            <xs:complexType name="deletePerson">
                <xs:sequence>
                    <xs:element name="arg0" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="deletePersonResponse">
                <xs:sequence>
                                        <xs:element  name="return"
type="xs:boolean"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="PersonNotFoundEx">
                <xs:sequence>
                     <xs:element name="message" type="xs:string"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="getPerson">
                <xs:sequence>
                    <xs:element name="arg0" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="getPersonResponse">
```

```
            <xs:sequence>
                <xs:element name="return" type="tns:person"
minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="person">
            <xs:sequence>
                <xs:element name="age" type="xs:int"/>
                                <xs:element  name="firstName"
type="xs:string" minOccurs="0"/>
                <xs:element name="id" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="countPersons">
            <xs:sequence/>
        </xs:complexType>
        <xs:complexType name="countPersonsResponse">
            <xs:sequence>
                <xs:element name="return" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="getAllPersons">
            <xs:sequence/>
        </xs:complexType>
        <xs:complexType name="getAllPersonsResponse">
            <xs:sequence>
                <xs:element name="return" type="tns:person"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="addPerson">
            <xs:sequence>
                <xs:element name="arg0" type="xs:int"/>
                   <xs:element name="arg1" type="xs:string"
minOccurs="0"/>
                <xs:element name="arg2" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="addPersonResponse">
            <xs:sequence>
                <xs:element name="return" type="tns:person"
minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
```

```
            <xs:complexType name="PersonExistsEx">
                <xs:sequence>
                    <xs:element name="message" type="xs:string"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="updatePerson">
                <xs:sequence>
                    <xs:element name="arg0" type="xs:int"/>
                       <xs:element name="arg1" type="xs:string"
minOccurs="0"/>
                    <xs:element name="arg2" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="updatePersonResponse">
                <xs:sequence>
                    <xs:element name="return" type="tns:person"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </types>
    <message name="getPerson">
        <part name="parameters" element="tns:getPerson"/>
    </message>
    <message name="getPersonResponse">
                                    <part     name="parameters"
element="tns:getPersonResponse"/>
    </message>
    <message name="PersonNotFoundEx">
        <part name="fault" element="tns:PersonNotFoundEx"/>
    </message>
    <message name="getAllPersons">
        <part name="parameters" element="tns:getAllPersons"/>
    </message>
    <message name="getAllPersonsResponse">
                                    <part     name="parameters"
element="tns:getAllPersonsResponse"/>
    </message>
    <message name="addPerson">
        <part name="parameters" element="tns:addPerson"/>
    </message>
    <message name="addPersonResponse">
```

```
                                    <part     name="parameters"
element="tns:addPersonResponse"/>
    </message>
    <message name="PersonExistsEx">
        <part name="fault" element="tns:PersonExistsEx"/>
    </message>
    <message name="deletePerson">
        <part name="parameters" element="tns:deletePerson"/>
    </message>
    <message name="deletePersonResponse">
                                    <part     name="parameters"
element="tns:deletePersonResponse"/>
    </message>
    <message name="updatePerson">
        <part name="parameters" element="tns:updatePerson"/>
    </message>
    <message name="updatePersonResponse">
                                    <part     name="parameters"
element="tns:updatePersonResponse"/>
    </message>
    <message name="countPersons">
        <part name="parameters" element="tns:countPersons"/>
    </message>
    <message name="countPersonsResponse">
                                    <part     name="parameters"
element="tns:countPersonsResponse"/>
    </message>
    <portType name="PersonService">
        <operation name="getPerson">
            <input message="tns:getPerson"/>
            <output message="tns:getPersonResponse"/>
                        <fault  message="tns:PersonNotFoundEx"
name="PersonNotFoundEx"/>
        </operation>
        <operation name="getAllPersons">
            <input message="tns:getAllPersons"/>
            <output message="tns:getAllPersonsResponse"/>
        </operation>
        <operation name="addPerson">
            <input message="tns:addPerson"/>
            <output message="tns:addPersonResponse"/>
                            <fault   message="tns:PersonExistsEx"
name="PersonExistsEx"/>
        </operation>
```

```
        <operation name="deletePerson">
            <input message="tns:deletePerson"/>
            <output message="tns:deletePersonResponse"/>
                        <fault  message="tns:PersonNotFoundEx"
name="PersonNotFoundEx"/>
        </operation>
        <operation name="updatePerson">
            <input message="tns:updatePerson"/>
            <output message="tns:updatePersonResponse"/>
                          <fault  message="tns:PersonExistsEx"
name="PersonExistsEx"/>
                        <fault  message="tns:PersonNotFoundEx"
name="PersonNotFoundEx"/>
        </operation>
        <operation name="countPersons">
            <input message="tns:countPersons"/>
            <output message="tns:countPersonsResponse"/>
        </operation>
    </portType>
                <binding     name="PersonServiceImplPortBinding"
type="tns:PersonService">
                                                    <soap:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
        <operation name="getPerson">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="PersonNotFoundEx">
                          <soap:fault  name="PersonNotFoundEx"
use="literal"/>
            </fault>
        </operation>
        <operation name="getAllPersons">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
```

```
            </output>
        </operation>
        <operation name="addPerson">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="PersonExistsEx">
                                <soap:fault  name="PersonExistsEx"
use="literal"/>
            </fault>
        </operation>
        <operation name="deletePerson">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="PersonNotFoundEx">
                            <soap:fault  name="PersonNotFoundEx"
use="literal"/>
            </fault>
        </operation>
        <operation name="updatePerson">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="PersonExistsEx">
                                <soap:fault  name="PersonExistsEx"
use="literal"/>
            </fault>
            <fault name="PersonNotFoundEx">
                            <soap:fault  name="PersonNotFoundEx"
use="literal"/>
            </fault>
```

```
        </operation>
        <operation name="countPersons">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="PersonService">
                            <port    name="PersonServiceImplPort"
binding="tns:PersonServiceImplPortBinding">
                                                    <soap:address
location="http://localhost:8081/personservice"/>
        </port>
    </service>
</definitions>
```

Plugin odpowiedzialny za wygenerowanie klas:

```
<build>
    <plugins>
        <plugin>
            <groupId>com.sun.xml.ws</groupId>
            <artifactId>jaxws-maven-plugin</artifactId>
            <version>3.0.2</version>
            <executions>
                <execution>
                    <goals>
                        <goal>wsimport</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <wsdlFiles>
                    <wsdlFile>
                        ${project.basedir}/src/main/resources/personservice.wsdl
                    </wsdlFile>
                </wsdlFiles>
                <packageName>org.example.jaxws.server_topdown</packageName>
                <sourceDestDir>${project.basedir}/src/main/java/</sourceDestDir>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Stworzyliśmy klasę **ESClient,** zdefiniowaliśmy adres URL usługi, utworzyliśmy obiekt PersonService_Service, pobraliśmy proxy klienta z obiektu PersonService_Service oraz użyliśmy proxy klienta do wywołania operacji Web Service. W pierszej linijce metody main jest wywołana metoda MyDate.info. Dodatkowo stworzyliśmy proste menu.

```
PawelK *
public class ESClient {

    7 usages
    static Scanner scan = new Scanner(System.in);

    PawelK *
    public static void main(String[] args) {
        MyData.info();
        PersonService_Service pService = new PersonService_Service();
        PersonService pServiceProxy = pService.getPersonServiceImplPort();

        ((BindingProvider) pServiceProxy).getRequestContext().put(BindingProviderProperties.REQUEST_TIMEOUT, 1000);

        boolean isRunning = true;

        while (isRunning) {
            try {

                System.out.println("1. Pobierz wszsytkie osoby");
                System.out.println("2. Pobierz osobę po id");
                System.out.println("3. Podaj liczbę osób");
                System.out.println("4. Dodaj osobę");
                System.out.println("5. Edytuj dane osobowe");
                System.out.println("6. Usuń osobę");
                System.out.println("7. Wyjście");

                System.out.println("Wybierz operację ");
                int option = scan.nextInt();
```

```
            switch (option) {                                                    39
                case 1 -> {
                    List<Person> allPersons = pServiceProxy.getAllPersons();
                    allPersons.forEach(ESClient::printPerson);
                }
                case 2 -> {
                    System.out.println("Podaj id");
                    Person person = pServiceProxy.getPerson(scan.nextInt());
                    printPerson(person);
                }
                case 3 -> System.out.println(pServiceProxy.countPersons());
                case 4 -> {
                    Object[] personParameters = getPersonParameters();
                    pServiceProxy.addPerson((int) personParameters[0], (String) personParameters[1], (int) personParameters[2]);
                    System.out.println("Zapisano");
                }
                case 5 -> {
                    Object[] personParameters = getPersonParameters();
                    pServiceProxy.updatePerson((int) personParameters[0], (String) personParameters[1], (int) personParameters[2]);
                    System.out.println("Zaaktualizowano");
                }
                case 6 -> {
                    System.out.println("Podaj id");
                    int id3 = scan.nextInt();
                    pServiceProxy.deletePerson(id3);
                    System.out.println("Usunięto");
                }
                case 7 -> isRunning = false;
                default -> System.out.println("Wprowadzono błędną opcję");
            }
```

```
        } catch (PersonNotFoundEx_Exception e) {
            System.out.println("Nie znaleziono użytkownika");
        } catch (PersonExistsEx_Exception e) {
            System.out.println("Taki użytkownik już istnieje");
        } catch (WebServiceException e) {
            System.out.println("Czas oczekiwania został przekroczony");
        }
    }
}

2 usages  PawelK
private static void printPerson(Person person) {
    System.out.println("Person{" +
            "id=" + person.getId() +
            ", firstName='" + person.getFirstName() + '\'' +
            ", age=" + person.getAge() +
            '}');
}

2 usages  PawelK
private static Object[] getPersonParameters() {
    Object[] parameters = new Object[3];
    System.out.println("Podaj id");
    parameters[0] = scan.nextInt();
    scan.nextLine();
    System.out.println("Podaj imię");
    parameters[1] = scan.nextLine();
    System.out.println("Podaj wiek");
    parameters[2] = scan.nextInt();
    return parameters;
```

Działanie systemu

System uruchomiliśmy w konfiguracji jedno maszynowej. Działanie systemu po stronie serwera prezentuje się następująco:

```
/home/pawelk/.jdks/corretto-17.0.5/bin/java ...
Paweł Kluska, 260391
Katya Zyatikava, 245891
25 kwietnia 18:17:51
17.0.5
pawelk
Linux
192.168.100.2
Web Service org.example.jaxws.server.PersonService is running ...
Press ENTER to STOP org.example.jaxws.server.PersonService ...
...called getAllPersons
...called getPerson id=1
...called updatePerson
...called deletePerson
```

Natomiast po stronie klienta zostały zwrócone wyniki metod wykonywanych przez serwer:

```
/home/pawelk/.jdks/corretto-17.0.5/bin/java ...
Paweł Kluska, 260391
Katya Zyatikava, 245891
25 kwietnia 18:17:56
17.0.5
pawelk
Linux
192.168.100.2
1. Pobierz wszsytkie osoby
2. Pobierz osobę po id
3. Podaj liczbę osób
4. Dodaj osobę
5. Edytuj dane osobowe
6. Usuń osobę
7. Wyjście
Wybierz operację
1
Person{id=1, firstName='Mariusz', age=9}
Person{id=2, firstName='Andrzej', age=10}
```

```
Wybierz operację
2
Podaj id
1
Person{id=1, firstName='Mariusz', age=9}
```

```
Wybierz operację
5
Podaj id
1
Podaj imię
Paweł
Podaj wiek
21
Zaaktualizowano
```

```
Wybierz operację
6
Podaj id
1
Usunięto
```

Dodatkowo został zaimplementowany mechanizm timeoutów. Gdy serwer wykonuje obliczenia dłużej niż sekunda, zostaje zwrócony wyjątek. Mechanizm prezentuje się następująco:

```
MyData.info();
PersonService_Service pService = new PersonService_Service();
PersonService pServiceProxy = pService.getPersonServiceImplPort();

((BindingProvider) pServiceProxy).getRequestContext().put(BindingProviderProperties.REQUEST_TIMEOUT, 1000);
```

W celu zasymulowania działania timeoutów wprowadziliśmy uśpienie na 5 sekund w metodzie addPerson()

```
1 usage  PawelK *
public Person addPerson(int id, String name, int age) throws PersonExistsEx {
    for (Person thePerson : personList) {
        if (thePerson.getId() == id) {
            throw new PersonExistsEx();
        }
    }
    Person person = new Person(id, name, age);
    try {
        Thread.sleep( millis: 5000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    personList.add(person);
    return person;
}
```

Teraz gdy wywołamy metodę addPerson zostanie rzucony wyjątek obsłużony po stronie klienta.

```
Wybierz operację
4
Podaj id
6
Podaj imię
Pawel
Podaj wiek
22
Czas oczekiwania został przekroczony
```