

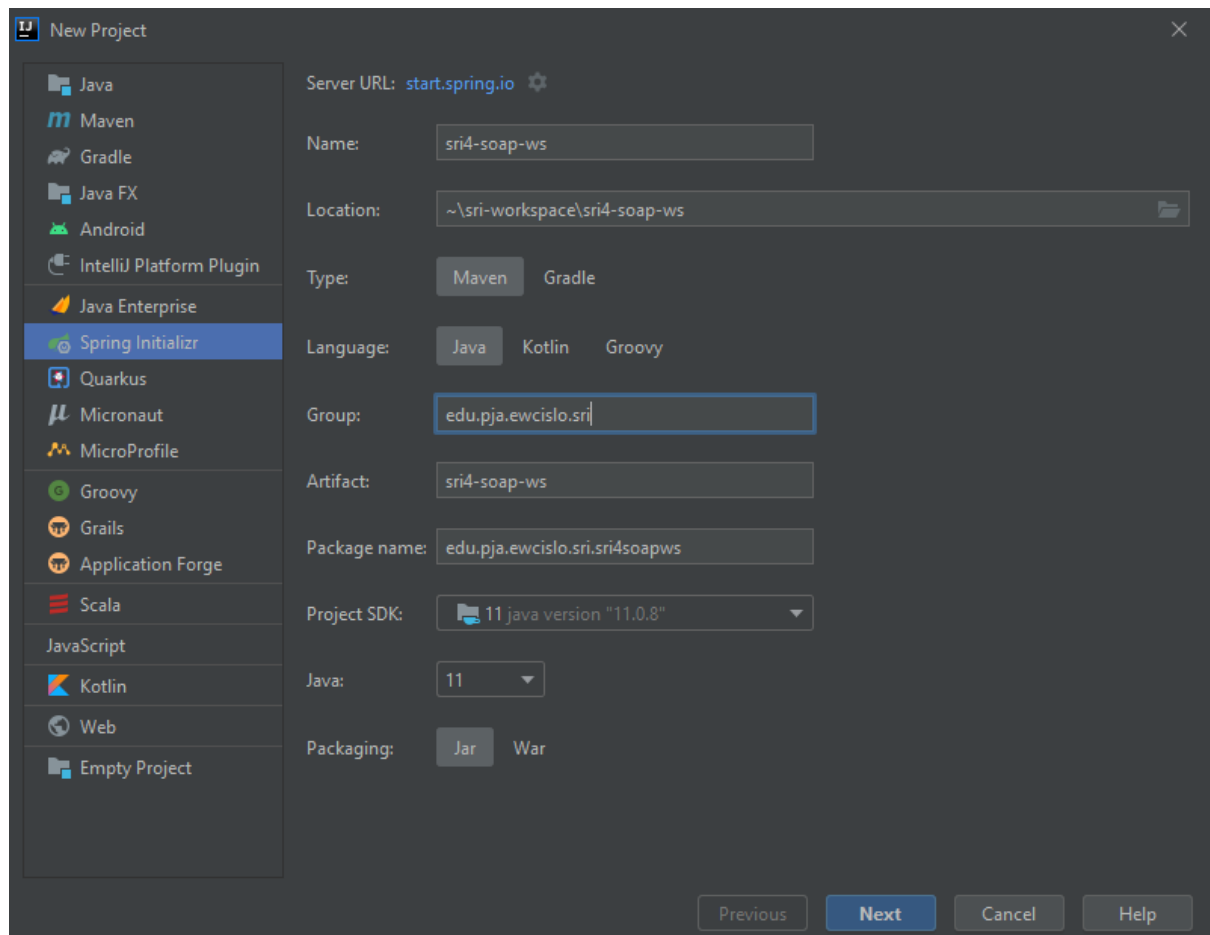
Systemy rozproszone i integracja usług - ćwiczenia nr 4: implementacja usługi typu SOAP

Celem ćwiczenia jest nabycie umiejętności implementacji usługi opartej na XML typu SOAP

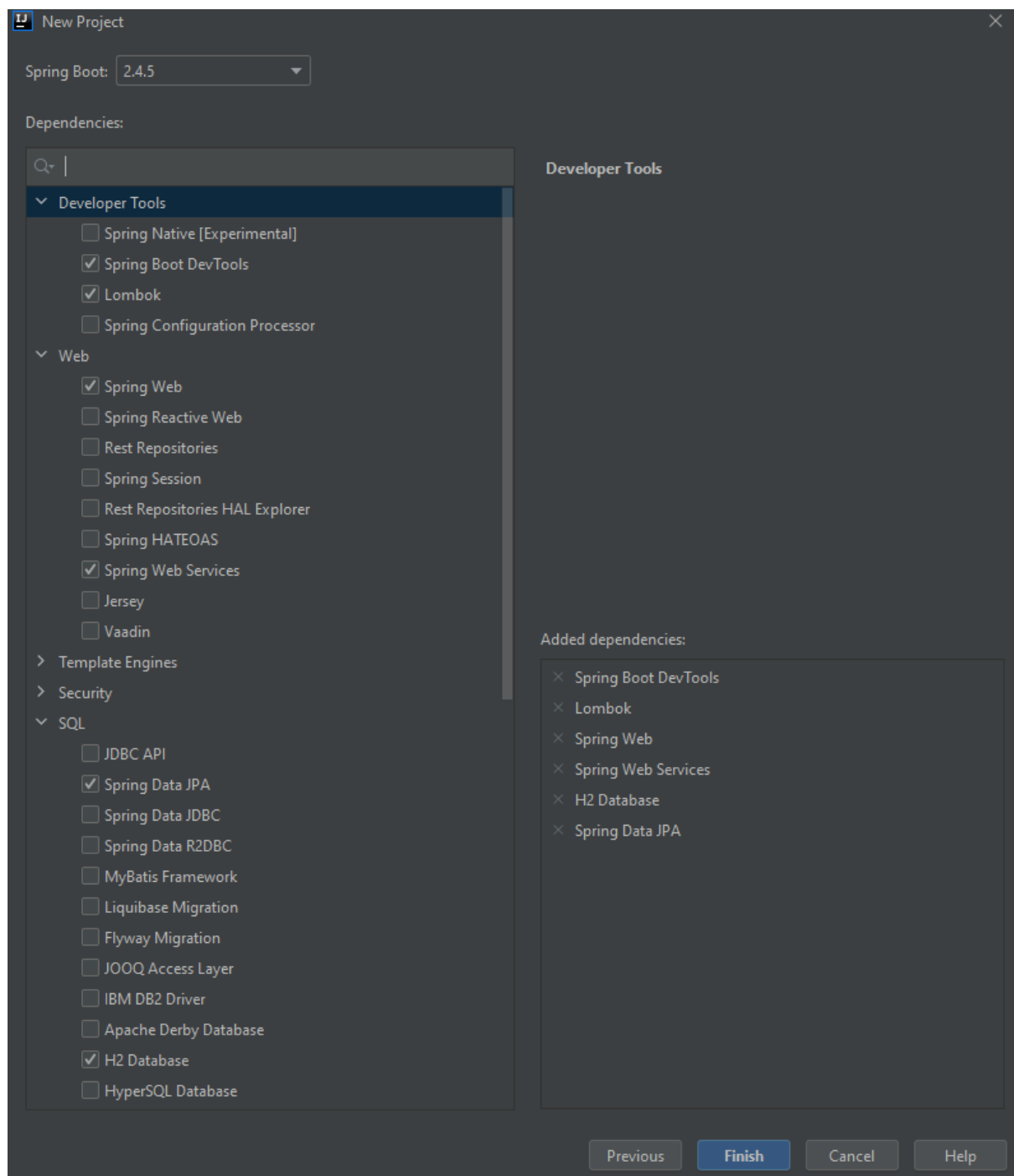
Część 1: zadanie wprowadzające

W ramach tego ćwiczenia zaimplementujemy przykładową usługę SOAP w technologii Java Spring. Zakres funkcjonalny będzie podobny do zadania 2 - będziemy udostępniać kilka podstawowych operacji na danych opartych na modelu z ćwiczenia 2.

1. Niezbędne oprogramowanie.
Do wykonania tego ćwiczenia niezbędne będzie środowisko Java (rekomendowane JDK Oracle 11 LTS), oraz środowisko developerskie (rekomendowane IntelliJ Idea Ultimate). Instrukcje instalacji znajdują się w opisie zadania 2.
2. Tworzenie szkieletu aplikacji
Podobnie jak w poprzednich ćwiczeniach wykorzystamy Spring Boot Initializer do tworzenia szkieletu aplikacji.
- 2.1. Tworzenie nowego projektu
W programie IntelliJ Idea Ultimate należy wybrać opcję New -> Project... z menu File Następnie należy wybrać 'Spring Initializr' z menu po lewej stronie okna dialogowego i uzupełnić podstawowe dane projektu:



Po wybraniu 'Next' należy zdefiniować zależności projektu, jak na poniższym zrzucie ekranu:



2.2. Uzupełnienie zależności w pom.xml:

W sekcji 'dependencies' należy dodać następujący fragment:

```
<dependency>  
  <groupId>wsdl4j</groupId>  
  <artifactId>wsdl4j</artifactId>  
</dependency>
```

A w sekcji 'plugins' należy dodać poniższy fragment:

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>jaxb2-maven-plugin</artifactId>
  <version>2.5.0</version>
  <executions>
    <execution>
      <id>xjc</id>
      <goals>
        <goal>xjc</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <sources>
      <source>${project.basedir}/src/main/resources/employees.xsd</source>
    </sources>
  </configuration>
</plugin>
```

Po zapisaniu zaktualizowanego pliku pom.xml należy kliknąć prawym klawiszem myszy na głównej gałęzi projektu w oknie po lewej stronie i z menu kontekstowego wybrać Maven -> Reload project. Nowe biblioteki powinny zostać pobrane i zainstalowane automatycznie.

3. Konfiguracja projektu

Należy upewnić się, że JDK, procesor adnotacji i plugin Lombok zostały poprawnie skonfigurowane. Szczegółowy opis znajduje się w opisie poprzedniego zadania 2 w punktach 3.1 - 3.3

4. Konfiguracja w pliku application.properties

Podobnie jak w zadaniu 2 wykorzystamy bazę danych H2 do przechowywania danych, na których będą wykonywane operacje wywoływane przez usługę sieciową.

```
spring.datasource.url=jdbc:h2:mem:sri-hr
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

5. Tworzenie modelu i repozytorium

Wykorzystamy klasy stworzone w ramach zadania 2:

```

Employee.java x
1  package edu.pja.sri.ewcislo.sri02.model;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  import javax.persistence.Entity;
8  import javax.persistence.GeneratedValue;
9  import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import java.time.LocalDate;
12
13 @Entity
14 @Data
15 @NoArgsConstructor
16 @AllArgsConstructor
17 public class Employee {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long id;
22
23     private String firstName;
24     private String lastName;
25     private LocalDate birthDate;
26     private String job;
27 }
28

```

```

EmployeeRepository.java x
1  package edu.pja.sri.ewcislo.sri02.repo;
2
3  import edu.pja.sri.ewcislo.sri02.model.Employee;
4  import org.springframework.data.repository.CrudRepository;
5
6  import java.util.List;
7
8  public interface EmployeeRepository extends CrudRepository<Employee, Long> {
9      List<Employee> findAll();
10 }
11

```

6. Tworzenie schematu XML komunikatów.
Należy utworzyć plik employees.xsd w katalogu src/main/resources. Będzie zawierać on definicję typów danych używanych w interfejsie usługi sieciowej, oraz definicję komunikatów:

```
employees.xsd
1  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
2      xmlns:tns="http://sri4soapws.ewcislo.sri.pja.edu/employees"
3      targetNamespace="http://sri4soapws.ewcislo.sri.pja.edu/employees"
4      elementFormDefault="qualified">
5
6      <xs:complexType name="employeeDto">
7          <xs:sequence>
8              <xs:element name="id" type="xs:decimal" minOccurs="0"/>
9              <xs:element name="firstName" type="xs:string"/>
10             <xs:element name="lastName" type="xs:string"/>
11             <xs:element name="birthDate" type="xs:date"/>
12             <xs:element name="job" type="xs:string"/>
13         </xs:sequence>
14     </xs:complexType>
15
16     <xs:element name="getEmployeesRequest">
17         <xs:complexType>
18         </xs:complexType>
19     </xs:element>
20
21     <xs:element name="getEmployeesResponse">
22         <xs:complexType>
23             <xs:sequence>
24                 <xs:element name="employees" type="tns:employeeDto" maxOccurs="unbounded"/>
25             </xs:sequence>
26         </xs:complexType>
27     </xs:element>
28
29     <xs:element name="getEmployeeByIdRequest">
30         <xs:complexType>
31             <xs:sequence>
32                 <xs:element name="employeeId" type="xs:decimal"/>
33             </xs:sequence>
34         </xs:complexType>
35     </xs:element>
```

```

36
37 <xs:element name="getEmployeeByIdResponse">
38   <xs:complexType>
39     <xs:sequence>
40       <xs:element name="employee" type="tns:employeeDto" minOccurs="0" maxOccurs="1"/>
41     </xs:sequence>
42   </xs:complexType>
43 </xs:element>
44
45 <xs:element name="addEmployeeRequest">
46   <xs:complexType>
47     <xs:sequence>
48       <xs:element name="employee" type="tns:employeeDto"/>
49     </xs:sequence>
50   </xs:complexType>
51 </xs:element>
52
53 <xs:element name="addEmployeeResponse">
54   <xs:complexType>
55     <xs:sequence>
56       <xs:element name="employeeId" type="xs:decimal" />
57     </xs:sequence>
58   </xs:complexType>
59 </xs:element>
60 </xs:schema>

```

W naszym projekcie będziemy tworzyć usługę z trzema metodami: `getEmployees()`, `getEmployeeById(id)` i `addEmployee(employee)`. Dla każdej z tych metod trzeba zdefiniować komunikat wejściowy i wyjściowy. Należy opisać również wszystkie własne typy danych używane w komunikacji. W naszym przykładzie opisana została klasa `Employee`. Więcej na temat definiowania typów danych w standardzie XML Schema można znaleźć tutaj: https://www.w3schools.com/xml/schema_intro.asp

7. Konfiguracja usługi SOAP

Należy utworzyć klasę `SoapWSConfig` zawierającą następującą treść:

```

1 package edu.pja.ewcislo.sri.sri4soapws.config;
2
3 import org.springframework.boot.web.servlet.ServletRegistrationBean;
4 import org.springframework.context.ApplicationContext;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.core.io.ClassPathResource;
8 import org.springframework.ws.config.annotation.EnableWs;
9 import org.springframework.ws.config.annotation.WsConfigurerAdapter;
10 import org.springframework.ws.transport.http.MessageDispatcherServlet;
11 import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
12 import org.springframework.xml.xsd.SimpleXsdSchema;
13 import org.springframework.xml.xsd.XsdSchema;
14
15 @EnableWs
16 @Configuration
17 public class SoapWSConfig extends WsConfigurerAdapter {
18
19     public static final String EMPLOYEE_NAMESPACE = "http://sri4soapws.ewcislo.sri.pja.edu/employees";
20
21     @Bean
22     public ServletRegistrationBean<MessageDispatcherServlet> messageDispatcherServlet(ApplicationContext applicationContext) {
23         MessageDispatcherServlet servlet = new MessageDispatcherServlet();
24         servlet.setApplicationContext(applicationContext);
25         servlet.setTransformWsdlLocations(true);
26         return new ServletRegistrationBean<>(servlet, "...urlMappings: \"/ws/*\"");
27     }
28
29     @Bean(name = "employees")
30     public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema employeesSchema) {
31         DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
32         wsdl11Definition.setPortTypeName("EmployeesPort");
33         wsdl11Definition.setLocationUri("/ws/employees");
34         wsdl11Definition.setTargetNamespace(EMPLOYEE_NAMESPACE);
35         wsdl11Definition.setSchema(employeesSchema);
36         return wsdl11Definition;
37     }
38
39     @Bean
40     public XsdSchema countriesSchema() { return new SimpleXsdSchema(new ClassPathResource("employees.xsd")); }
41
42 }

```

Zawiera ona podstawową konfigurację usługi i połączenie jej do stworzonego wcześniej schematu danych w pliku xsd.

8. Tworzenie danych początkowych
Opcjonalnie można stworzyć klasę DataInitializer, która będzie odpowiedzialna za załadowanie bazy danych przy starcie aplikacji:

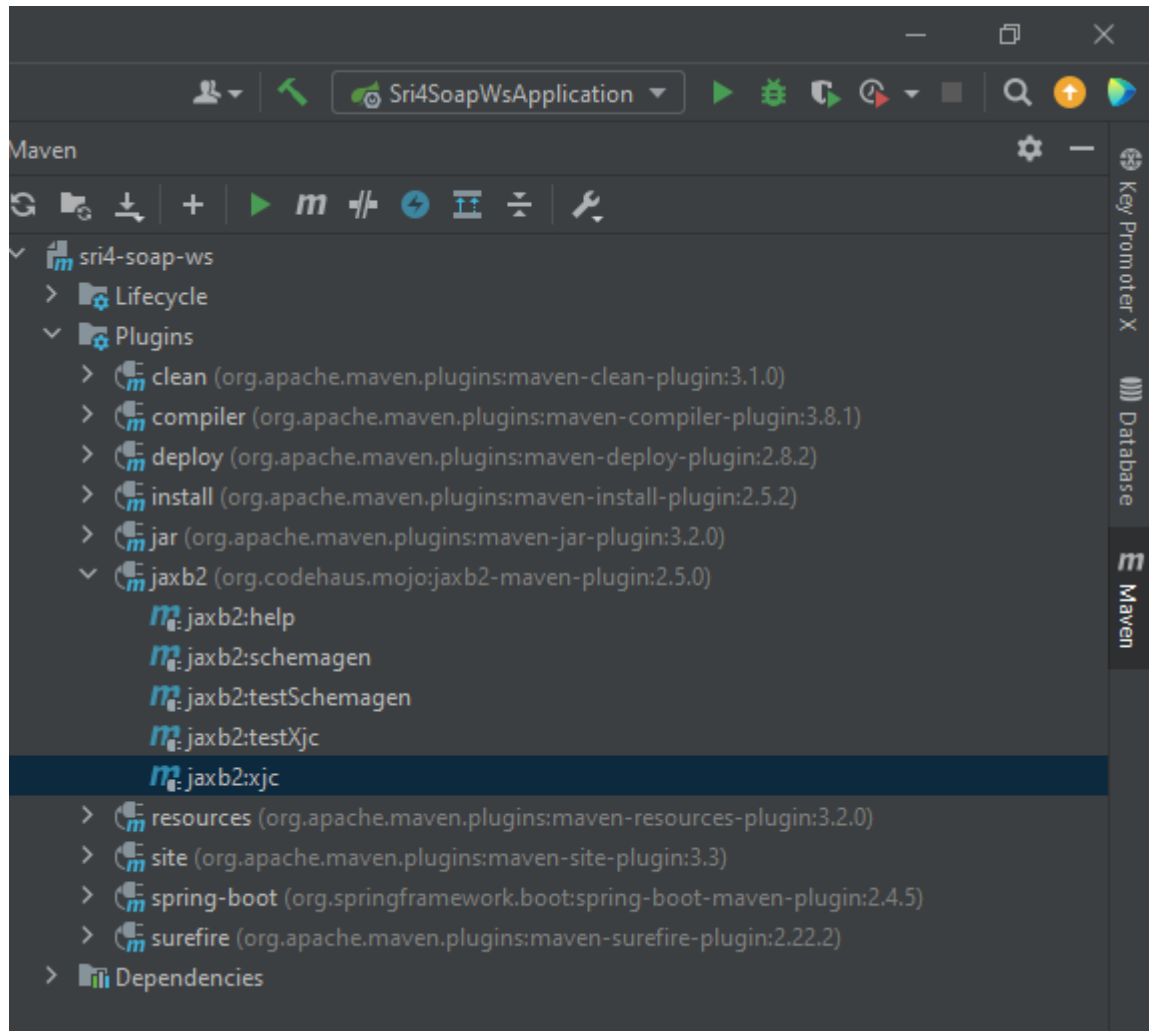

```

1  package edu.pja.ewcislo.sri.sri4soapws;
2
3  import edu.pja.ewcislo.sri.sri4soapws.model.Employee;
4  import edu.pja.ewcislo.sri.sri4soapws.repo.EmployeeRepository;
5  import lombok.RequiredArgsConstructor;
6  import org.slf4j.Logger;
7  import org.slf4j.LoggerFactory;
8  import org.springframework.context.ApplicationListener;
9  import org.springframework.context.event.ContextRefreshedEvent;
10 import org.springframework.stereotype.Component;
11
12 import java.time.LocalDate;
13 import java.util.Arrays;
14
15 @Component
16 @RequiredArgsConstructor
17 public class DataInitializer implements ApplicationListener<ContextRefreshedEvent> {
18
19     private static final Logger LOG = LoggerFactory.getLogger(DataInitializer.class);
20
21     private final EmployeeRepository employeeRepository;
22
23     public void initData() {
24         Employee e1 = Employee.builder()
25             .firstName("Jan")
26             .lastName("Kowalski")
27             .job("Clerk")
28             .birthDate(LocalDate.of(year: 1990, month: 01, dayOfMonth: 01))
29             .build();
30         Employee e2 = Employee.builder()
31             .firstName("Adam")
32             .lastName("Nowak")
33             .job("Manager")
34             .birthDate(LocalDate.of(year: 1991, month: 01, dayOfMonth: 01))
35             .build();
36         Employee e3 = Employee.builder()
37             .firstName("Anna")
38             .lastName("Iksińska")
39             .job("Assistant")
40             .birthDate(LocalDate.of(year: 1992, month: 01, dayOfMonth: 01))
41             .build();
42
43         employeeRepository.saveAll(Arrays.asList(e1, e2, e3));
44         LOG.info("Data initialized");
45     }
46
47     @Override
48     public void onApplicationEvent(ContextRefreshedEvent event) { initData(); }
49
50 }
51

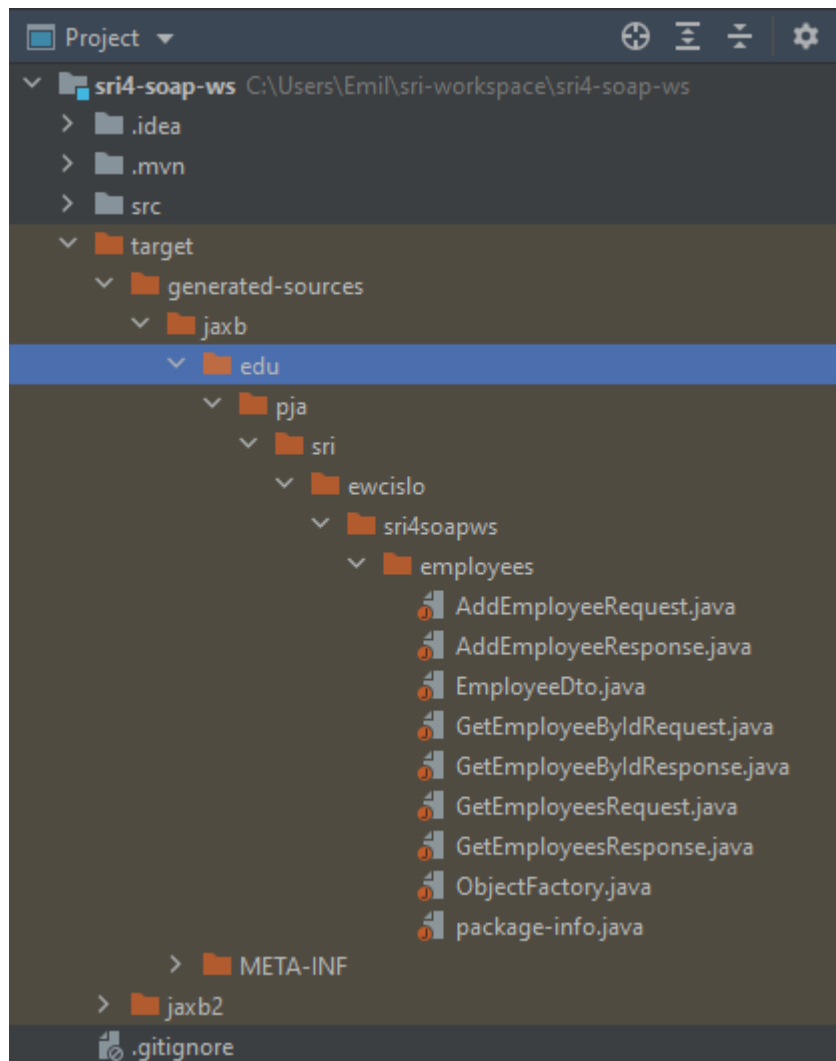
```

9. Generowanie klas na podstawie schematu xsd

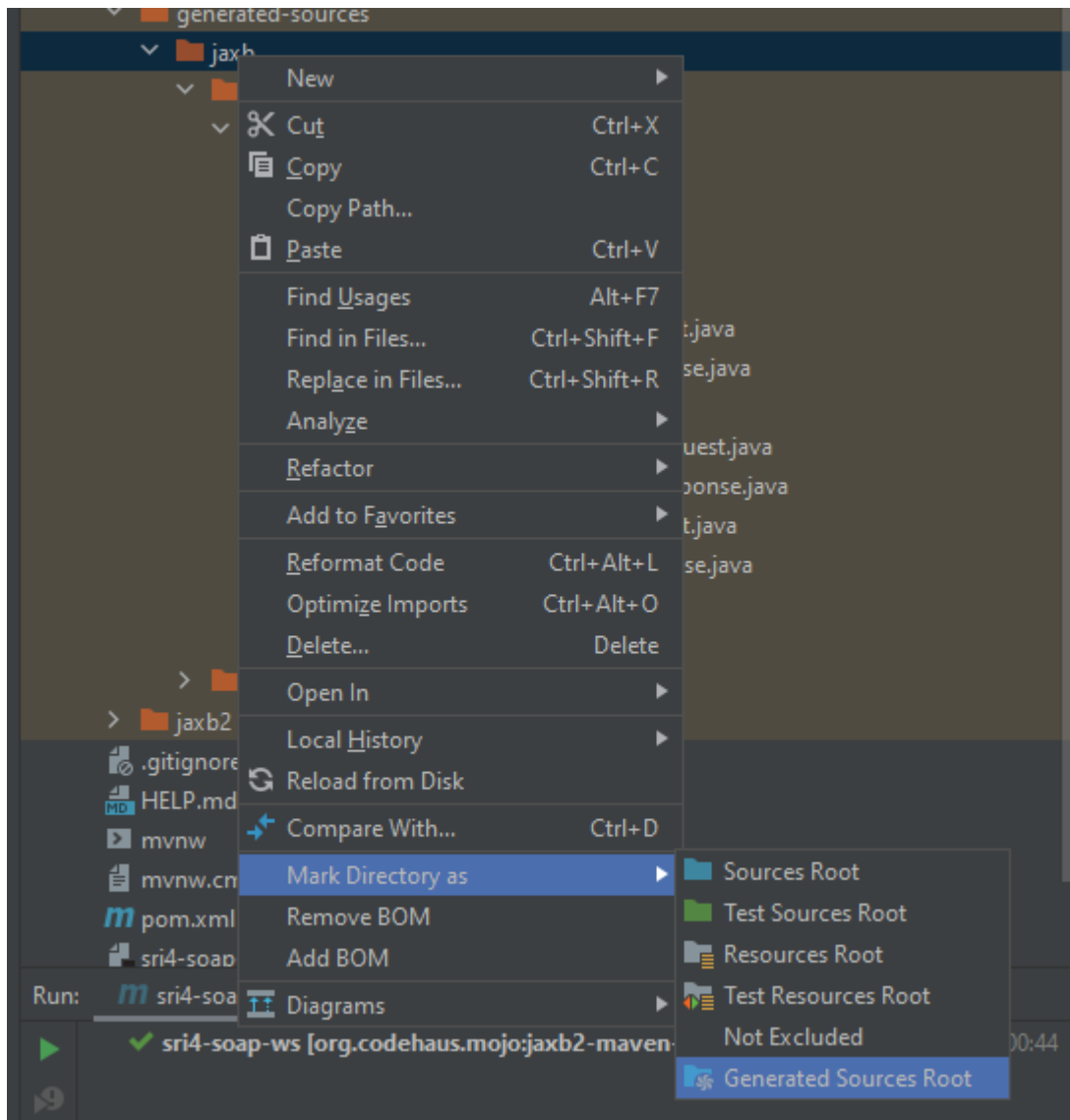
Ponieważ usługa typu SOAP korzysta z modelu danych opisanych w schemacie XML Schema, potrzebujemy klas wygenerowanych na podstawie tego schematu, aby wykorzystać je w implementacji usługi. W tym celu uruchamiamy plugin jaxb2 dodany wcześniej do pliku pom.xml. Aby to zrobić, należy otworzyć zakładkę 'Maven' w środowisku IntelliJ (domyślnie przy prawej krawędzi) i wybrać zadanie 'jaxb2:xjc' z katalogu 'Plugins':



Po wykonaniu zadania w katalogu target powinny pojawić się wygenerowane klasy:



Klas tych na razie nie możemy wykorzystać w projekcie (nie są dodane do classpath projektu). Aby to naprawić, należy kliknąć prawym klawiszem myszy na folderze jaxb i wybrać opcję 'Mark Directory as Generated Sources Root':



10. Tworzenie implementacji usługi
W tym celu należy stworzyć klasę EmployeeEndpoint o następującej treści:

```

EmployeeEndpoint.java
1  package edu.pja.ewcislo.sri.sri4soapws.endpoint;
2
3  import edu.pja.ewcislo.sri.sri4soapws.config.SoapWSConfig;
4  import edu.pja.ewcislo.sri.sri4soapws.model.Employee;
5  import edu.pja.ewcislo.sri.sri4soapws.repo.EmployeeRepository;
6  import edu.pja.sri.ewcislo.sri4soapws.employees.*;
7  import lombok.RequiredArgsConstructor;
8  import org.springframework.ws.server.endpoint.annotation.Endpoint;
9  import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
10 import org.springframework.ws.server.endpoint.annotation.RequestPayload;
11 import org.springframework.ws.server.endpoint.annotation.ResponsePayload;
12
13 import javax.xml.datatype.DatatypeConfigurationException;
14 import javax.xml.datatype.DatatypeFactory;
15 import javax.xml.datatype.XMLGregorianCalendar;
16 import java.math.BigDecimal;
17 import java.util.List;
18 import java.util.Optional;
19 import java.util.stream.Collectors;
20
21 @Endpoint
22 @RequiredArgsConstructor
23 public class EmployeeEndpoint {
24
25     private final EmployeeRepository employeeRepository;
26
27     @PayloadRoot(namespace = SoapWSConfig.EMPLOYEE_NAMESPACE, localPart = "getEmployeesRequest")
28     @ResponsePayload
29     public GetEmployeesResponse getEmployees(@RequestPayload GetEmployeesRequest req) {
30         List<Employee> employeeList = employeeRepository.findAll();
31         List<EmployeeDto> dtoList = employeeList.stream()
32             .map( this::convertToDto )
33             .collect(Collectors.toList());
34         GetEmployeesResponse res = new GetEmployeesResponse();
35         res.getEmployees().addAll(dtoList);
36         return res;
37     }

```

```

39     @PayloadRoot(namespace = SoapWSConfig.EMPLOYEE_NAMESPACE, localPart = "getEmployeeByIdRequest")
40     @ResponsePayload
41     @ public GetEmployeeByIdResponse getEmployeeById(@RequestPayload GetEmployeeByIdRequest req) {
42         Long employeeId = req.getEmployeeId().longValue();
43         Optional<Employee> emp = employeeRepository.findById(employeeId);
44         GetEmployeeByIdResponse res = new GetEmployeeByIdResponse();
45         res.setEmployee(convertToDto(emp.orElse( other: null)));
46         return res;
47     }
48
49     @PayloadRoot(namespace = SoapWSConfig.EMPLOYEE_NAMESPACE, localPart = "addEmployeeRequest")
50     @ResponsePayload
51     @ public AddEmployeeResponse addEmployee(@RequestPayload AddEmployeeRequest req) {
52         EmployeeDto empDto = req.getEmployee();
53         Employee emp = convertToEntity(empDto);
54         employeeRepository.save(emp);
55         AddEmployeeResponse response = new AddEmployeeResponse();
56         response.setEmployeeId(new BigDecimal(emp.getId()));
57         return response;
58     }

```

```

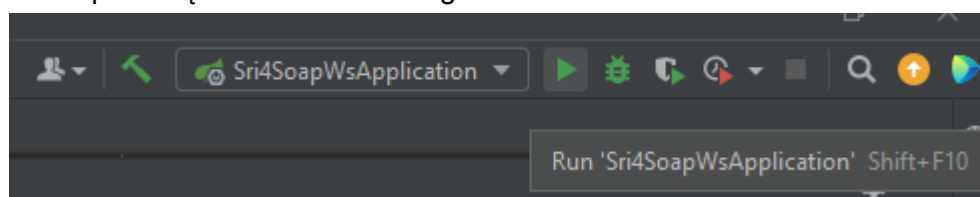
61 private EmployeeDto convertToDto(Employee e) {
62     if(e == null) {
63         return null;
64     }
65     try {
66         EmployeeDto dto = new EmployeeDto();
67         dto.setId(new BigDecimal(e.getId()));
68         dto.setFirstName(e.getFirstName());
69         dto.setLastName(e.getLastName());
70         XMLGregorianCalendar birthDate = null;
71         birthDate = DatatypeFactory.newInstance().newXMLGregorianCalendar(e.getBirthDate().toString());
72         dto.setBirthDate(birthDate);
73         dto.setJob(e.getJob());
74
75         return dto;
76     } catch (DatatypeConfigurationException datatypeConfigurationException) {
77         datatypeConfigurationException.printStackTrace();
78         return null;
79     }
80 }
81
82 @ private Employee convertToEntity(EmployeeDto dto) {
83     return Employee.builder()
84         .id(dto.getId() != null ? dto.getId().longValue() : null)
85         .firstName(dto.getFirstName())
86         .lastName(dto.getLastName())
87         .job(dto.getJob())
88         .build();
89 }
90
91 }

```

Nasza usługa zawiera trzy metody biznesowe: `getEmployees()`, `getEmployeeById()` i `addEmployee()`. Jako parametry tych metod oraz rezultat wykorzystujemy klasy wygenerowane na podstawie schematu xsd. Ponieważ konieczna jest konwersja z wygenerowanej klasy `EmployeeDto` na klasę encji `Employee` i odwrotnie, stworzymy metody konwertujące: `convertToDto()` i `convertToEntity()`.

11. Uruchomienie aplikacji

Aplikację uruchamiamy klikając na zieloną strzałkę w prawym górnym rogu ekranu, lub za pomocą skrótu klawiszowego `Shift-F10`:



Po uruchomieniu na konsoli nie powinny pojawić się błędy:

```
Spring Boot (v2.4.5)
2021-05-08 01:14:44.656 INFO 50528 --- [ restartedMain] e.p.e.s.s.Sri4SoapWsApplication : Starting Sri4SoapWsApplication using Java 11.0.8 on EMIL-DESKTOP with PID 50528 (C:\Users\Emil\workspace\sri4-soap-ws)
2021-05-08 01:14:44.658 INFO 50528 --- [ restartedMain] e.p.e.s.s.Sri4SoapWsApplication : No active profile set, falling back to default profiles: default
2021-05-08 01:14:44.695 INFO 50528 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2021-05-08 01:14:44.695 INFO 50528 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2021-05-08 01:14:45.177 INFO 50528 --- [ restartedMain] s.s.p.s.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-05-08 01:14:45.210 INFO 50528 --- [ restartedMain] s.s.p.s.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 27 ms. Found 1 JPA repository interfaces.
2021-05-08 01:14:45.353 INFO 50528 --- [ restartedMain] trationDelegate$BeanPostProcessorChecker : Bean 'soapWSConfig' of type [edu.pja.eu.cislo.sri.sri4soaps.config.SoapWSConfig$$EnhancerBySpringCGLIB$$4374ac7] is not a
2021-05-08 01:14:45.354 INFO 50528 --- [ restartedMain] a.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2021-05-08 01:14:45.608 INFO 50528 --- [ restartedMain] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-05-08 01:14:45.615 INFO 50528 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-05-08 01:14:45.615 INFO 50528 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.45]
2021-05-08 01:14:45.679 INFO 50528 --- [ restartedMain] o.s.a.c.c.Tomcat.[localhost.]/ : Initializing Spring embedded WebApplicationContext
2021-05-08 01:14:45.699 INFO 50528 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 983 ms
2021-05-08 01:14:45.714 INFO 50528 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-05-08 01:14:45.788 INFO 50528 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-05-08 01:14:45.793 INFO 50528 --- [ restartedMain] o.s.b.e.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:sri-hr'
2021-05-08 01:14:45.895 INFO 50528 --- [ restartedMain] o.hibernat.jpa.internal.util.LogHelper : HHN000204: Processing PersistenceUnitInfo [name: default]
2021-05-08 01:14:45.928 INFO 50528 --- [ restartedMain] org.hibernate.Version : HHN000412: Hibernate ORM core version 5.4.30.Final
2021-05-08 01:14:46.006 INFO 50528 --- [ restartedMain] o.hibernat.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-05-08 01:14:46.071 INFO 50528 --- [ restartedMain] org.hibernate.dialect.Dialect : HHN000400: Using dialect: org.hibernate.dialect.H2Dialect
2021-05-08 01:14:46.399 INFO 50528 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHN000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2021-05-08 01:14:46.405 INFO 50528 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-05-08 01:14:46.428 INFO 50528 --- [ restartedMain] o.s.b.e.s.OptionalLiveReloadServer : LiveReload server is running on port 35729
2021-05-08 01:14:46.456 WARN 50528 --- [ restartedMain] jpaee.config.Configuration$JpaeeConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure it to false to reduce this log level.
2021-05-08 01:14:46.778 INFO 50528 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-05-08 01:14:46.945 INFO 50528 --- [ restartedMain] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-05-08 01:14:46.995 INFO 50528 --- [ restartedMain] e.p.e.s.s.sri.sri4soaps.DataInitializer : Data initialized
2021-05-08 01:14:47.008 INFO 50528 --- [ restartedMain] e.p.e.s.s.Sri4SoapWsApplication : Started Sri4SoapWsApplication in 2.015 seconds (JVM running for 4.101)
```

12. Testowanie usługi SOAP za pomocą programu SoapUI

12.1. Instalacja oprogramowania

Ze strony <https://www.soapui.org/downloads/soapui/> należy pobrać SoapUI w wersji open source i zainstalować.

12.2. Testowanie usługi

Po uruchomieniu programu SoapUI należy stworzyć nowy projekt wybierając z menu File -> New SOAP Project:

New SOAP Project
Creates a WSDL/SOAP based Project in this workspace

Project Name:

Initial WSDL:

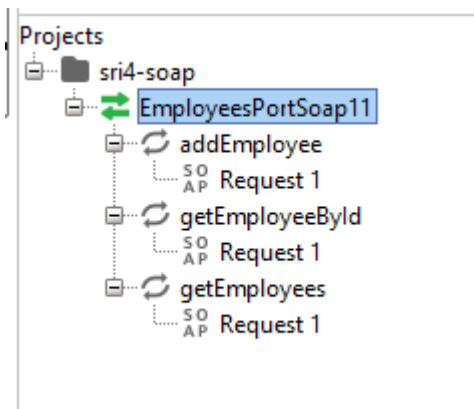
Create Requests: ☒ Create sample requests for all operations?

Create TestSuite: ☐ Creates a TestSuite for the imported WSDL

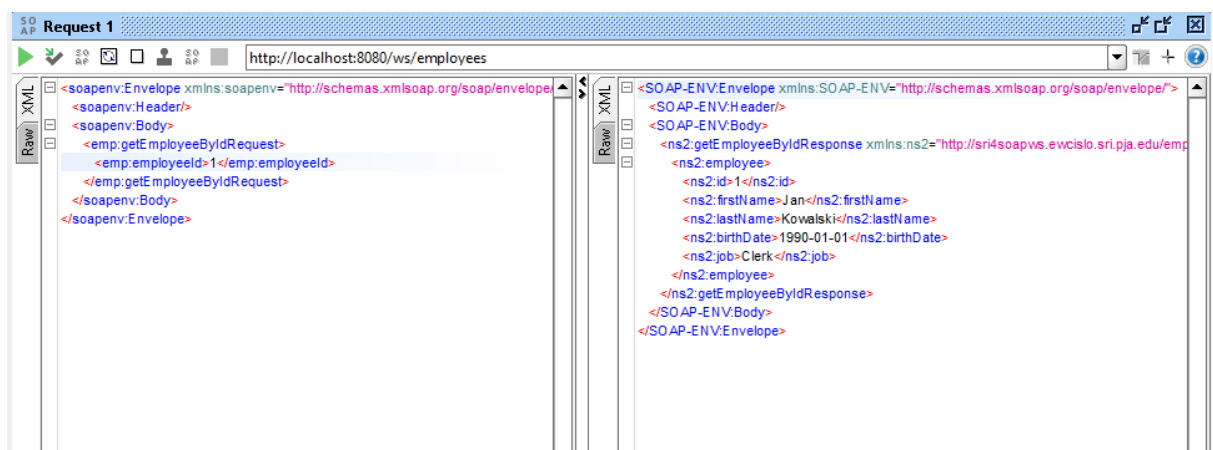
Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

Należy uzupełnić nazwę projektu oraz wskazanie na lokalizację pliku WSDL zawierający opis usługi. Opis ten zostanie wygenerowany automatycznie przez naszą aplikację po jej uruchomieniu. Publikowany jest pod adresem: <http://localhost:8080/ws/employees.wsdl>

Po utworzeniu projektu SoapUI odczyta schemat usługi i wygeneruje przykładowe żądania:



Po wybraniu żądania i uzupełnieniu danych można przetestować daną metodę klikając na zieloną strzałkę uruchamiającą żądanie:



Część 2: zadanie na ocenę

Zaprojektuj i zaimplementuj system realizujący dowolną funkcjonalność biznesową, udostępniony za pomocą usługi typu SOAP. Minimalne wymagania to:

1. Usługa musi udostępniać co najmniej 3 metody biznesowe.
2. Należy zastosować własne klasy do jako parametry metod oraz jako typy danych zwracanych przez metody.
3. Projekt musi mieć sensowną funkcjonalność biznesową.

Projekt należy przetestować za pomocą SoapUI, lub podobnego narzędzia.

Gotowy projekt i udokumentowane testy (np. w postaci zrzutów ekranu z narzędzia testującego) należy wysłać jako rozwiązanie.