Opis interfejsów SOAP/REST

Laboratorium 3

Wymagania

1. Edytor plików xml/yaml (np. Visual Studio Code)
2. Dostęp do Swagger Editor
3. IntelliJ IDEA Community / Ultimate lub inne IDE umożliwiające pracę z Java + Maven.

Wstęp

Laboratorium polegać będzie na przećwiczeniu sposobów opisów interfejsów udostępnianych jako WebService oraz REST API. Przygotowane struktury odzwierciedlać mają następująco zamodelowane klasy obiektów.

- Użytkownik
    - Imię
    - Nazwisko
    - Wiek
    - Pesel
    - Obywatelstwo
    - Adres email
- Obywatelstwo
    - Przyjmuje jedną z wartości: PL, DE, UK

Zadanie 1

Przygotowanie specyfikacji WebService'u w postaci pliku WSDL przy wykorzystaniu narzędzi do generowania opisu interfejsu na podstawie utworzonego kodu.

1. Utworzyć pliku users.xsd odzwierciedlający model opisany na początku instrukcji. Plik powinien zawierać opis modelu oraz dodatkowo opis zapytań i odpowiedzi np. dla metody getUser (getUserRequest, getUserResponse).
2. Przygotować nowy projekt w frameworku Spring Boot. Projekt musi posiadać moduł Spring Web Services. Podstawowy szablon można wygenerować za pomocą Spring Initializr.
3. Do pliku pom.xml dodać następującą zależność:

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

4. Utworzony plik user.xsd wkleić do zasobów projektu (src/resource/users.xsd).

5. Do projektu dodać plugin, który podczas budowania projektu wygeneruje modele na podstawie specyfikacji opisanej w *user.xsd*. Do *pom.xml* w sekcji <*plugins* /> należy dodać następujący 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/users.xsd</source>
    </sources>
  </configuration>
</plugin>
```

Utworzone modele pojawią się w ścieżce *target/generated-sources/jaxb*

6. Utworzyć konfiguracja WebService'u. Należy stworzyć następująca klasę:

```
@EnableWs
@Configuration
public class WebServiceConfig extends WsConfigurerAdapter {
    @Bean
    public ServletRegistrationBean messageDispatcherServlet(ApplicationContext
applicationContext) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean(servlet, "/ws/*");
    }
    @Bean(name = "users")
    public DefaultWsdl11Definition defaultWsdl11Definition (XsdSchema
usersSchema) {
        DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
        wsdl11Definition.setPortTypeName("UserssPort");
        wsdl11Definition.setLocationUri("/ws");
        wsdl11Definition.setTargetNamespace("http://spring.io/guides/gs-producing-
web-service");
        wsdl11Definition.setSchema(usersSchema);
        return wsdl11Definition;
    }
    @Bean public XsdSchema usersSchema() {
        return new SimpleXsdSchema(new ClassPathResource("users.xsd"));
    }
}
```

7. Uruchomić aplikacje i zaprezentować wygenerowany opis interfejsu. Po poprawnym wykonaniu instrukcji WSDL powinien znajdować się pod adresem:

http://localhost:8080/ws/users.wsdl

Zadanie 2

Przygotowanie specyfikacji interfejsu REST API w oparciu o standard OpenAPI.

1. Specyfikacja dotyczy modelu określonego na początku instrukcji laboratoryjnej. Model ma zostać wykorzystany do opisu interfejsu CRUD.
2. Modele muszą określać odpowiednie typy oraz formaty pól. Należy skorzystać z elementów określających długość ciągu znaków, akceptowalny wzorzec oraz określić wymagalność pól.
3. Do każdego zapytania dodać informacje o dacie wysłania żądania oraz identyfikatorze zapytania. Ta sama informacja powinno zostać uwzględniona również w odpowiedzi.
4. Każde z pól powinno posiadać przykładową wartość.
5. Opis interfejsu powinien uwzględniać informacje o metodzie autoryzacji, która wymagana jest podczas dostępu do API. Proszę o zabezpieczenie dwóch endpointów za pomocą Basic Authentication oraz dwóch pozostałych za pomocą Bearer Authentication.

Przydatne linki:

1. Edytor YAML: https://editor.swagger.io/
2. Specyfikacja OpenAPI 2.0: https://swagger.io/specification/v2/
3. Przykładowy interfejs: https://petstore.swagger.io/