

Projet Webservices

Etudiants : Abibou Wade & Coumba Sabaly

Nom du projet : SunuBiblio

Type : Application Web Services (REST + SOAP)

Technologies : Java 17, Spring Boot 3, Spring Data JPA, Spring WS, H2 Database, SoapUI, Swagger

Objectif :

Créer un service de gestion de livres pour une bibliothèque sénégalaise, accessible par API REST et par Web Service SOAP.

- **REST API** pour les étudiants/professeurs : consulter, réserver livres.
- **SOAP API** pour les bibliothécaires : ajouter, modifier, supprimer, prêter et retourner livres.

Technos:

- Java, Spring Boot (avec Spring WS pour SOAP)
- Base de données **H2**
- Test avec **Postman** (REST) + **SoapUI** (SOAP) + **Swagger** pour documentation REST.

Dossiers du projet :

-**entity**/ → pour les classes Java du modèle (ex: Livre, Reservation)

-**dto**/ → pour transporter les données entre client et serveur

-**repository**/ → pour interagir avec la BDD

-**service**/ → pour la logique métier

-**restcontroller**/ → pour les API REST

-**soap**/ → pour les WebServices SOAP

-**soap/dto** → pour les DTO spécifiques aux WebServices SOAP

The image shows the Spring Initializr web interface. On the left, under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', '3.4.5' is selected. The 'Project Metadata' section includes fields for Group (sn.unchk), Artifact (sunubiblio), Name (sunubiblio), Description (Demo project for Spring Boot), and Package name (sn.unchk.sunubiblio). Under 'Packaging', 'Jar' is selected, and under 'Java', '17' is selected. On the right, the 'Dependencies' section shows selected dependencies: 'Spring Web Services' (WEB), 'H2 Database' (SQL), 'Spring Data JPA' (SQL), and 'Spring Web' (WEB). At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and a menu icon.

Figure 1 : Initialisation avec <https://start.spring.io/> avec les dépendances (Spring web, H2 database, Spring Data JPA, Spring Web Services)

The image shows a code editor with the file 'application.properties' open. The content of the file is as follows:

```
src > main > resources > application.properties
1 # Configuration H2
2 spring.datasource.url=jdbc:h2:file:./data/sunubibliodb
3 spring.datasource.driverClassName=org.h2.Driver
4 spring.datasource.username=sa
5 spring.datasource.password=
6
7 # JPA Configuration
8 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
9 spring.jpa.show-sql=true
10 spring.jpa.hibernate.ddl-auto=update
11
12 # H2 Console (important pour voir la base visuellement)
13 spring.h2.console.enabled=true
14 spring.h2.console.path=/h2-console
15
16 # Port de l'application
17 server.port=8080
18
19 |
```

Figure 2 : Configuration de **H2 database** accessible sur <http://localhost:8080/h2-console>.

I.REST API

On va commencer avec la partie **REST API**

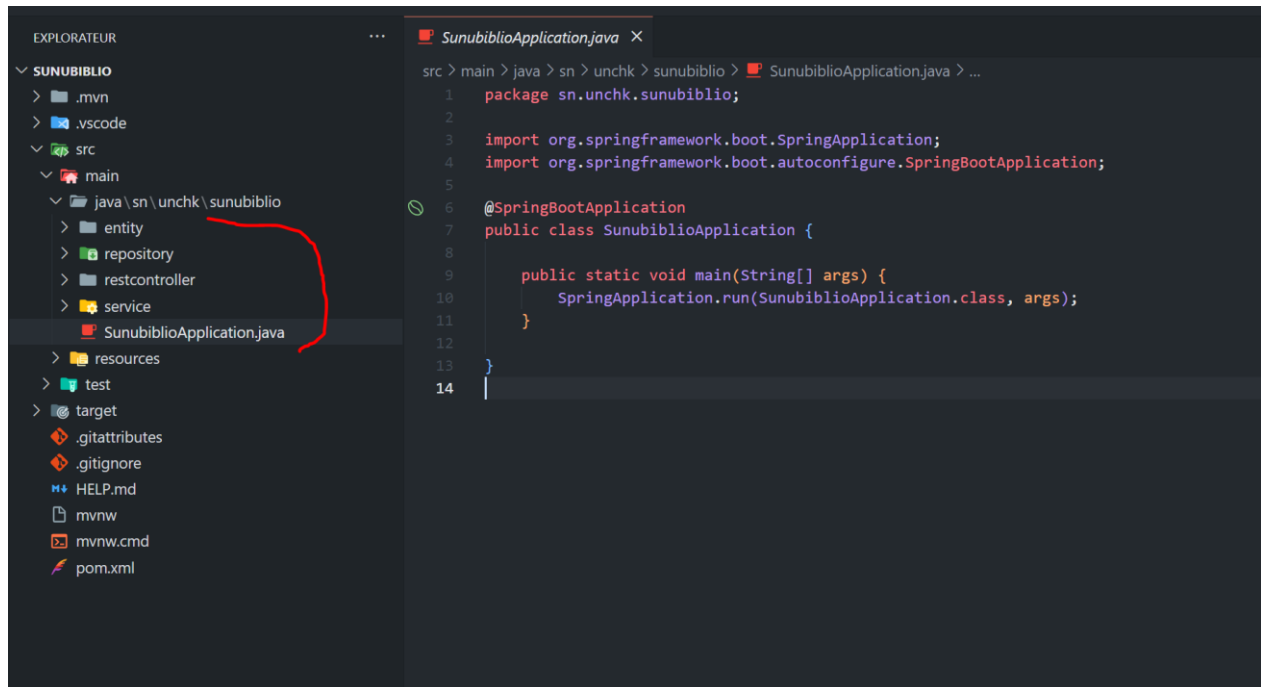


Figure 3 : Création des dossiers pour **REST API** (entity, repository, restcontroller, service) et notre fichier **SunubiblioApplication.java** qui est notre fichier base pour lancer notre application avec la commande :

`./mvnw spring-boot:run` sur un terminal vscode.

Création des fichiers :

Les Entity :

Entity/Livre.java

```
package sn.unchk.sunubiblio.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Livre {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
private String titre;
private String auteur;
private boolean disponible = true; // Par défaut un livre est disponible

// Constructeurs
public Livre() {}

public Livre(String titre, String auteur) {
    this.titre = titre;
    this.auteur = auteur;
    this.disponible = true;
}

// Getters et Setters
public Long getId() {
    return id;
}

public String getTitre() {
    return titre;
}

public void setTitre(String titre) {
    this.titre = titre;
}

public String getAuteur() {
    return auteur;
}

public void setAuteur(String auteur) {
    this.auteur = auteur;
}

public boolean isDisponible() {
    return disponible;
}

public void setDisponible(boolean disponible) {
    this.disponible = disponible;
}
}
```

Entity/Reservation.java

```
package sn.unchk.sunubiblio.entity;

import jakarta.persistence.*;
import java.time.LocalDate;

@Entity
public class Reservation {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nomClient; // étudiant ou professeur
    private LocalDate dateReservation;

    @ManyToOne
    @JoinColumn(name = "livre_id")
    private Livre livre;

    public Reservation() {}

    public Reservation(String nomClient, LocalDate dateReservation, Livre livre)
    {
        this.nomClient = nomClient;
        this.dateReservation = dateReservation;
        this.livre = livre;
    }

    // Getters et Setters
    public Long getId() {
        return id;
    }

    public String getNomClient() {
        return nomClient;
    }

    public void setNomClient(String nomClient) {
        this.nomClient = nomClient;
    }

    public LocalDate getDateReservation() {
        return dateReservation;
    }
}
```

```

    public void setDateReservation(LocalDate dateReservation) {
        this.dateReservation = dateReservation;
    }

    public Livre getLivre() {
        return livre;
    }

    public void setLivre(Livre livre) {
        this.livre = livre;
    }
}

```

Les repository :

Repository/LivreRepository.java

```

package sn.unchk.sunubiblio.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import sn.unchk.sunubiblio.entity.Livre;

@Repository
public interface LivreRepository extends JpaRepository<Livre, Long> {

    // Méthode pour trouver tous les livres disponibles
    List<Livre> findByDisponibleTrue();
}

```

Repository/ReservationRepository.java

```

package sn.unchk.sunubiblio.repository;

import sn.unchk.sunubiblio.entity.Reservation;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository

```

```
public interface ReservationRepository extends JpaRepository<Reservation, Long> {  
  
}
```

Les services :

Service/LivreService.java

```
package sn.unchk.sunubiblio.service;  
  
import java.util.List;  
import java.util.Optional;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import sn.unchk.sunubiblio.entity.Livre;  
import sn.unchk.sunubiblio.repository.LivreRepository;  
  
@Service  
public class LivreService {  
  
    @Autowired  
    private LivreRepository livreRepository;  
  
    // Récupérer tous les livres  
    public List<Livre> getAllLivres() {  
        return livreRepository.findAll();  
    }  
  
    // Récupérer un livre par son ID  
    public Optional<Livre> getLivreById(Long id) {  
        return livreRepository.findById(id);  
    }  
  
    // Récupérer tous les livres disponibles  
    public List<Livre> getLivresDisponibles() {  
        return livreRepository.findByDisponibleTrue();  
    }  
  
    // Ajouter ou mettre à jour un livre  
    public Livre saveLivre(Livre livre) {  
        return livreRepository.save(livre);  
    }  
}
```

```

    // Supprimer un livre
    public void deleteLivre(Long id) {
        livreRepository.deleteById(id);
    }
}

```

Service/ReservationService.java

```

package sn.unchk.sunubiblio.service;

import sn.unchk.sunubiblio.entity.Livre;
import sn.unchk.sunubiblio.entity.Reservation;
import sn.unchk.sunubiblio.repository.LivreRepository;
import sn.unchk.sunubiblio.repository.ReservationRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.Optional;

@Service
public class ReservationService {

    @Autowired
    private ReservationRepository reservationRepository;

    @Autowired
    private LivreRepository livreRepository;

    // Créer une réservation
    public Reservation reserverLivre(Long livreId, String nomClient) throws
Exception {
        Optional<Livre> livreOpt = livreRepository.findById(livreId);

        if (livreOpt.isEmpty()) {
            throw new Exception("Livre non trouvé");
        }

        Livre livre = livreOpt.get();

        if (!livre.isDisponible()) {
            throw new Exception("Livre non disponible");
        }
    }
}

```



```

        // Marquer le livre comme non disponible
        livre.setDisponible(false);
        livreRepository.save(livre);

        // Créer la réservation
        Reservation reservation = new Reservation();
        reservation.setNomClient(nomClient);
        reservation.setDateReservation(LocalDate.now());
        reservation.setLivre(livre);

        return reservationRepository.save(reservation);
    }

    // Suivre une réservation par ID
    public Optional<Reservation> getReservationById(Long id) {
        return reservationRepository.findById(id);
    }
}

```

Les Controller:

restcontroller/LivreRestController.java

```

package sn.unchk.sunubiblio.restcontroller;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import sn.unchk.sunubiblio.entity.Livre;
import sn.unchk.sunubiblio.service.LivreService;

@RestController
@RequestMapping("/api/livres")
public class LivreRestController {

    @Autowired
    private LivreService livreService;
}

```

```

// Récupérer tous les livres
@GetMapping
public List<Livre> getAllLivres() {
    return livreService.getAllLivres();
}

// Récupérer un livre par ID
@GetMapping("/{id}")
public Optional<Livre> getLivreById(@PathVariable Long id) {
    return livreService.getLivreById(id);
}

// Récupérer tous les livres disponibles
@GetMapping("/disponibles")
public List<Livre> getLivresDisponibles() {
    return livreService.getLivresDisponibles();
}
}

```

restcontroller/ReservationRestController.java

```

package sn.unchk.sunubiblio.restcontroller;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import sn.unchk.sunubiblio.entity.Reservation;
import sn.unchk.sunubiblio.service.ReservationService;

@RestController
@RequestMapping("/api/reservations")
public class ReservationRestController {

    @Autowired
    private ReservationService reservationService;
}

```

```
// Créer une réservation
@PostMapping
public Reservation reserverLivre(@RequestParam Long livreId, @RequestParam
String nomClient) throws Exception {
    return reservationService.reserverLivre(livreId, nomClient);
}

// Suivre une réservation par ID
@GetMapping("/{id}")
public Optional<Reservation> getReservationById(@PathVariable Long id) {
    return reservationService.getReservationById(id);
}
}
```

Ajout **Swagger** pour la documentation

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.5.0</version>
</dependency>
```

On va ajouter cette dépendance dans le fichier pom.xml de notre projet. Puis relancer le server et va sur le lien : <http://localhost:8080/swagger-ui/index.html>

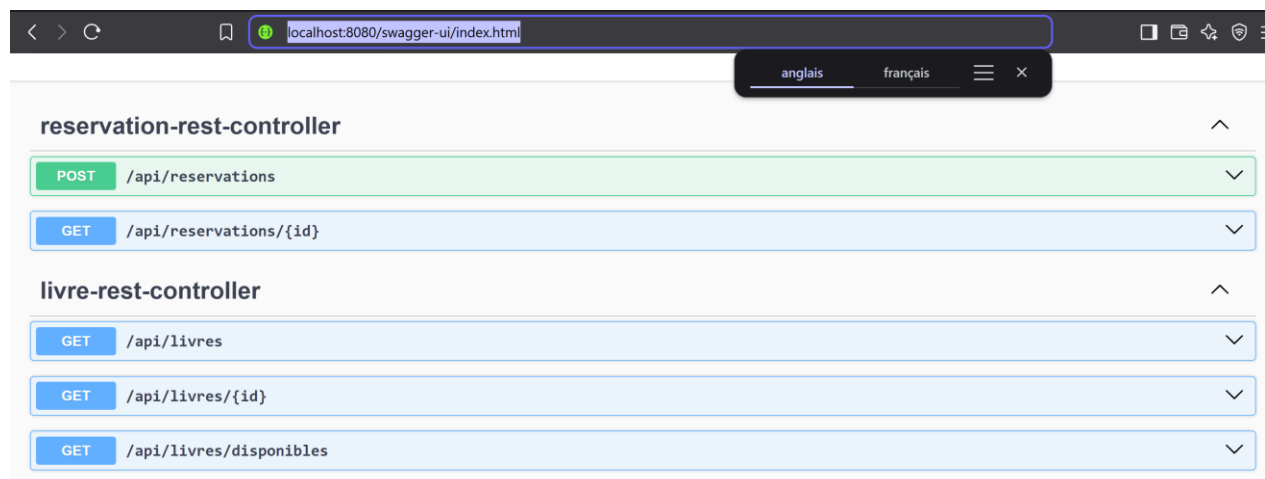


Figure 4 : Notre REST API pour le moment.

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▾

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:file:./data/sunubibliodb

User Name: sa

Password:

Connect Test Connection

Figure 5 : On va accéder à <http://localhost:8080/h2-console> pour insertion données avec H2 database avec `./data/sunubibliodb` comme JDBC Url et `sa` comme username comme on l'avait choisi dans le fichier `application.properties` .

localhost:8080/h2-console/login.do?jsessionId=00669887515e7019ad9cb5f7d1965c12

Auto commit Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:sunubibliodb

- LIVRE
- RESERVATION
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

```
INSERT INTO LIVRE (AUTEUR, DISPONIBLE, TITRE)
VALUES ('Victor Hugo', 'true', 'Les Misérables');
```

INSERT INTO LIVRE (AUTEUR, DISPONIBLE, TITRE)
VALUES ('Victor Hugo', 'true', 'Les Misérables');
Update count: 1
(7 ms)

Figure 5 : Création d'un livre avec ses valeurs (Auteur, Disponible, Titre)

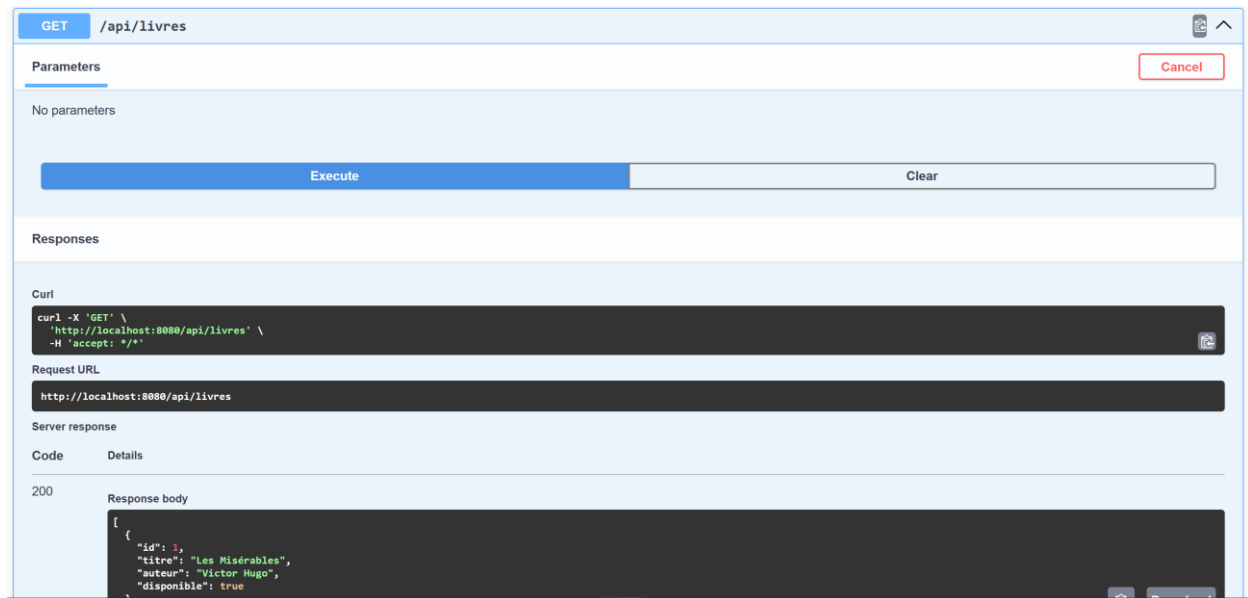


Figure 6 : Test avec **GET localhost:8080/api/livres** et les autres requêtes Get.

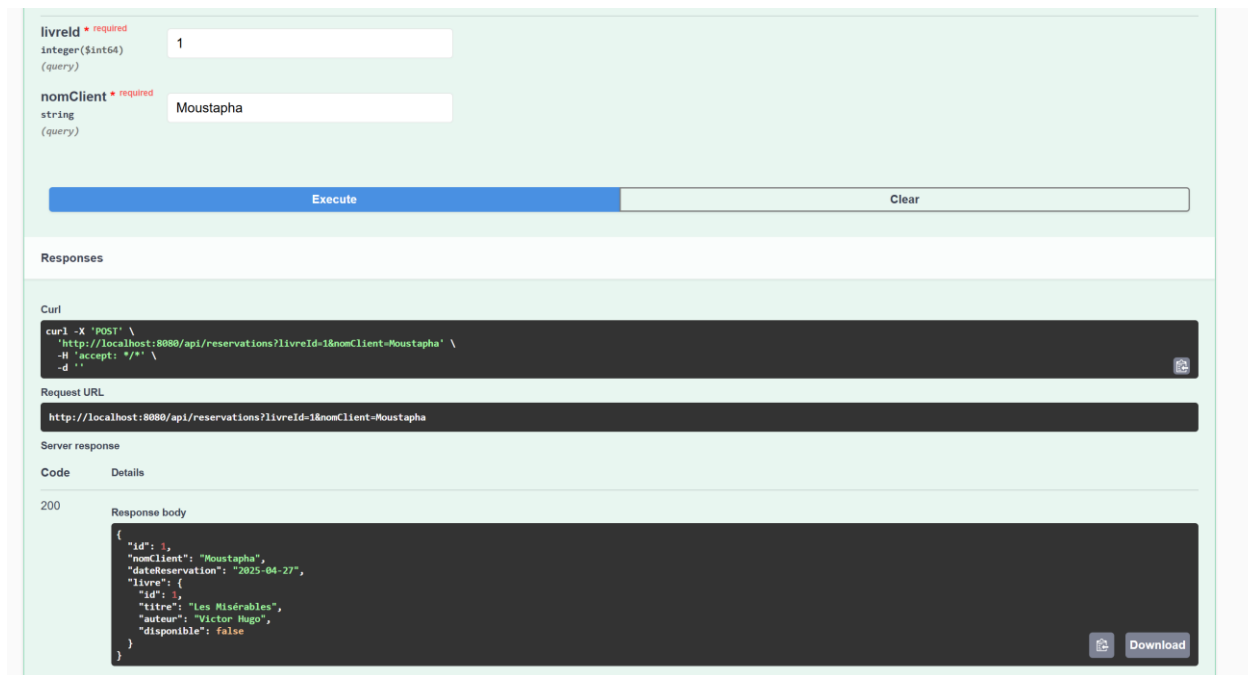


Figure 7 : Test avec **POST localhost:8080/api/reservations** avec l'id du Premier livre qu'on avait créé et un **user** du nom de « Mousatapha » et ça met aussi sa disponibilité **false** vu quelqu'un la réserver.



Figure 8 : On a notre réservation sur notre base de données.

II.SOAP

Maintenant on a terminé avec la partie **REST** on va passer par la partie **SOAP**

On va créer un package **soap** dans notre projet.

Puis un fichier : **soap/LivreSoapEndpoint.java**

```
package sn.unchk.sunubiblio.soap;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;

import sn.unchk.sunubiblio.entity.Livre;
import sn.unchk.sunubiblio.entity.Reservation;
import sn.unchk.sunubiblio.service.LivreService;
import sn.unchk.sunubiblio.service.ReservationService;
import sn.unchk.sunubiblio.soap.dto.AjouterLivreRequest;
import sn.unchk.sunubiblio.soap.dto.AjouterLivreResponse;
import sn.unchk.sunubiblio.soap.dto.ModifierLivreRequest;
import sn.unchk.sunubiblio.soap.dto.ModifierLivreResponse;
import sn.unchk.sunubiblio.soap.dto.PreterLivreRequest;
```

```

import sn.unchk.sunubiblio.soap.dto.PreterLivreResponse;
import sn.unchk.sunubiblio.soap.dto.RetournerLivreRequest;
import sn.unchk.sunubiblio.soap.dto.RetournerLivreResponse;
import sn.unchk.sunubiblio.soap.dto.SupprimerLivreRequest;
import sn.unchk.sunubiblio.soap.dto.SupprimerLivreResponse;

@Endpoint
public class LivreSoapEndpoint {

    private static final String NAMESPACE_URI = "http://sunubiblio.sn/soap";

    @Autowired
    private LivreService livreService;

    @Autowired
    private ReservationService reservationService;

    // Ajouter un livre
    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "AjouterLivreRequest")
    @ResponsePayload
    public AjouterLivreResponse ajouterLivre(@RequestPayload AjouterLivreRequest
request) {
        Livre livre = new Livre();
        livre.setTitre(request.getTitre());
        livre.setAuteur(request.getAuteur());
        livre.setDisponible(true);

        Livre savedLivre = livreService.saveLivre(livre);

        AjouterLivreResponse response = new AjouterLivreResponse();
        response.setId(savedLivre.getId());
        response.setTitre(savedLivre.getTitre());
        response.setAuteur(savedLivre.getAuteur());
        response.setDisponible(savedLivre.isDisponible());

        return response;
    }

    // Modifier un livre
    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "ModifierLivreRequest")
    @ResponsePayload
    public ModifierLivreResponse modifierLivre(@RequestPayload
ModifierLivreRequest request) {
        ModifierLivreResponse response = new ModifierLivreResponse();

```

```

        Optional<Livre> optionalLivre =
livreService.getLivreById(request.getId());
        if (optionalLivre.isPresent()) {
            Livre livre = optionalLivre.get();
            livre.setTitre(request.getTitre());
            livre.setAuteur(request.getAuteur());
            livreService.savelivre(livre);
            response.setMessage("Livre modifié avec succès !");
        } else {
            response.setMessage("Livre non trouvé !");
        }
        return response;
    }

    // Supprimer un livre
    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "SupprimerLivresRequest")
    @ResponsePayload
    public SupprimerLivresResponse supprimerLivres(@RequestPayload
SupprimerLivresRequest request) {
        SupprimerLivresResponse response = new SupprimerLivresResponse();

        Optional<Livre> optionalLivre =
livreService.getLivreById(request.getId());
        if (optionalLivre.isPresent()) {
            livreService.deleteLivres(request.getId());
            response.setMessage("Livres supprimés avec succès !");
        } else {
            response.setMessage("Livres non trouvés !");
        }
        return response;
    }

    // Prêter un livre
    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "PreterLivresRequest")
    @ResponsePayload
    public PreterLivresResponse preterLivres(@RequestPayload PreterLivresRequest
request) {
        PreterLivresResponse response = new PreterLivresResponse();

        try {
            Reservation reservation =
reservationService.reserverLivres(request.getLivresId(), request.getNomClient());
            response.setMessage("Livres prêtés avec succès à " +
reservation.getNomClient());
        } catch (Exception e) {

```



```

        response.setMessage("Erreur lors du prêt : " + e.getMessage());
    }

    return response;
}

// Retourner un livre
@PayloadRoot(namespace = NAMESPACE_URI, localPart = "RetournerLivreRequest")
@ResponsePayload
public RetournerLivreResponse retournerLivre(@RequestPayload
RetournerLivreRequest request) {
    RetournerLivreResponse response = new RetournerLivreResponse();

    Optional<Livre> optionalLivre =
livreService.getLivreById(request.getLivreId());
    if (optionalLivre.isPresent()) {
        Livre livre = optionalLivre.get();
        livre.setDisponible(true);
        livreService.saveLivre(livre);
        response.setMessage("Livre retourné avec succès !");
    } else {
        response.setMessage("Livre non trouvé !");
    }
    return response;
}
}

```

Un autre fichier : **soap/WebServiceConfig.java**

```

package sn.unchk.sunubiblio.soap;

import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.oxm.jaxb.Jaxb2Marshaller;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@EnableWs
@Configuration
public class WebServiceConfig {

    @Bean

```

```

    public ServletRegistrationBean<MessageDispatcherServlet>
messageDispatcherServlet(ApplicationContext applicationContext) {
    MessageDispatcherServlet servlet = new MessageDispatcherServlet();
    servlet.setApplicationContext(applicationContext);
    servlet.setTransformWsdlLocations(true);
    return new ServletRegistrationBean<>(servlet, "/ws/*");
}

@Bean(name = "livres")
public DefaultWsd11Definition defaultWsd11Definition(XsdSchema
livresSchema) {
    DefaultWsd11Definition wsdl11Definition = new DefaultWsd11Definition();
    wsdl11Definition.setPortTypeName("LivresPort");
    wsdl11Definition.setLocationUri("/ws");
    wsdl11Definition.setTargetNamespace("http://sunubiblio.sn/soap");
    wsdl11Definition.setSchema(livresSchema);
    return wsdl11Definition;
}

@Bean
public XsdSchema livresSchema() {
    return new SimpleXsdSchema(new
org.springframework.core.io.ClassPathResource("xsd/livre.xsd"));
}

@Bean
public Jaxb2Marshaller marshaller() {
    Jaxb2Marshaller marshaller = new Jaxb2Marshaller();
    marshaller.setClassesToBeBound(
        sn.unchk.sunubiblio.soap.dto.AjouterLivreRequest.class,
        sn.unchk.sunubiblio.soap.dto.AjouterLivreResponse.class,
        sn.unchk.sunubiblio.soap.dto.ModifierLivreRequest.class,
        sn.unchk.sunubiblio.soap.dto.ModifierLivreResponse.class,
        sn.unchk.sunubiblio.soap.dto.SupprimerLivreRequest.class,
        sn.unchk.sunubiblio.soap.dto.SupprimerLivreResponse.class,
        sn.unchk.sunubiblio.soap.dto.PreterLivreRequest.class,
        sn.unchk.sunubiblio.soap.dto.PreterLivreResponse.class,
        sn.unchk.sunubiblio.soap.dto.RetournerLivreRequest.class,
        sn.unchk.sunubiblio.soap.dto.RetournerLivreResponse.class
    );
    return marshaller;
}
}

```

Et on va créer un fichier **livre.xsd** dans `resources/xsd`.

Ce fichier **XSD** (XML Schema) sert à définir **la structure** des requêtes et réponses SOAP (comme un modèle).

`resources/xsd/livre.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://sunubiblio.sn/soap"
  xmlns:tns="http://sunubiblio.sn/soap"
  elementFormDefault="qualified">

  <!-- Ajouter un Livre -->
  <xs:element name="AjouterLivreRequest">

    <xs:complexType>
      <xs:sequence>
        <xs:element name="titre" type="xs:string"/>
        <xs:element name="auteur" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="AjouterLivreResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:long"/>
        <xs:element name="titre" type="xs:string"/>
        <xs:element name="auteur" type="xs:string"/>
        <xs:element name="disponible" type="xs:boolean"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Modifier un Livre -->
  <xs:element name="ModifierLivreRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:long"/>
        <xs:element name="titre" type="xs:string"/>
        <xs:element name="auteur" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


```

```

<xs:element name="ModifierLivreResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="message" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Supprimer un Livre -->
<xs:element name="SupprimerLivreRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="xs:long"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SupprimerLivreResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="message" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Prêter un Livre -->
<xs:element name="PreterLivreRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="livreId" type="xs:long"/>
      <xs:element name="nomClient" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="PreterLivreResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="message" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Retourner un Livre -->
<xs:element name="RetournerLivreRequest">

```

```

        <xs:complexType>
            <xs:sequence>
                <xs:element name="livreId" type="xs:long"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="RetournerLivreResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="message" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>

```

Ce que ça définit livre.xsd :

- AjouterLivreRequest
- AjouterLivreResponse
- ModifierLivreRequest
- ModifierLivreResponse
- SupprimerLivreRequest
- SupprimerLivreResponse
- PreterLivreRequest
- PreterLivreResponse
- RetournerLivreRequest
- RetournerLivreResponse

Maintenant on va ajouter une dépendance à notre pom.xml pour la génération du WSDL.

```

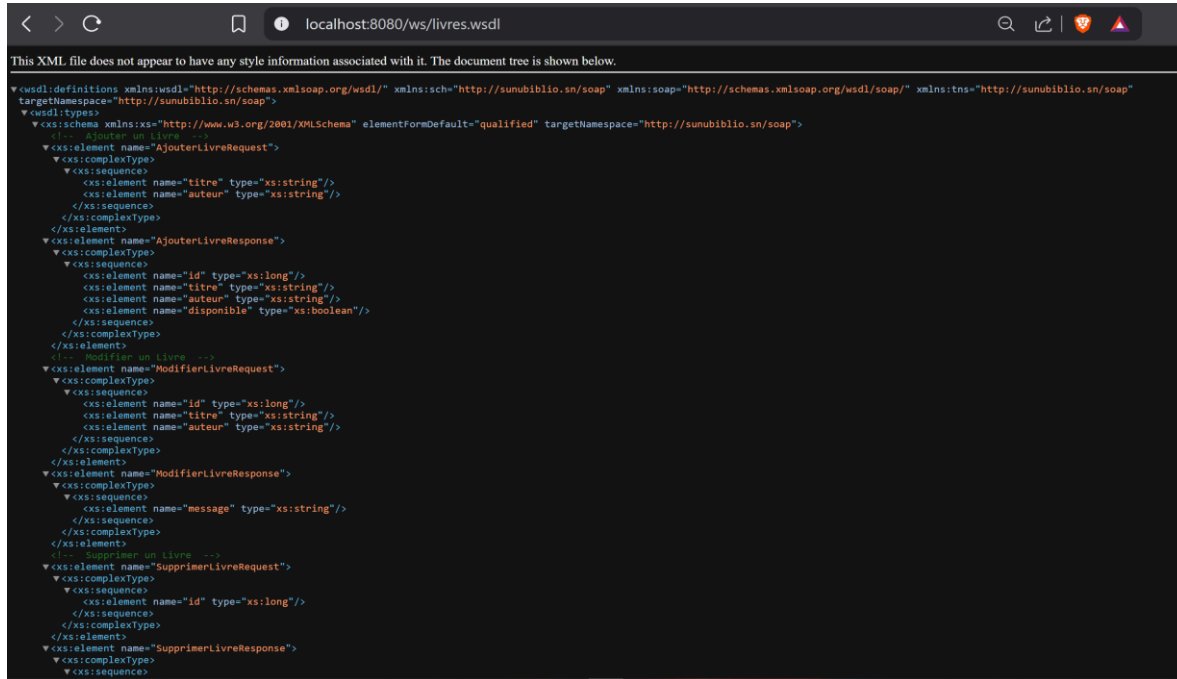
<dependency>
    <groupId>wsdl4j</groupId>
    <artifactId>wsdl4j</artifactId>
</dependency>

```

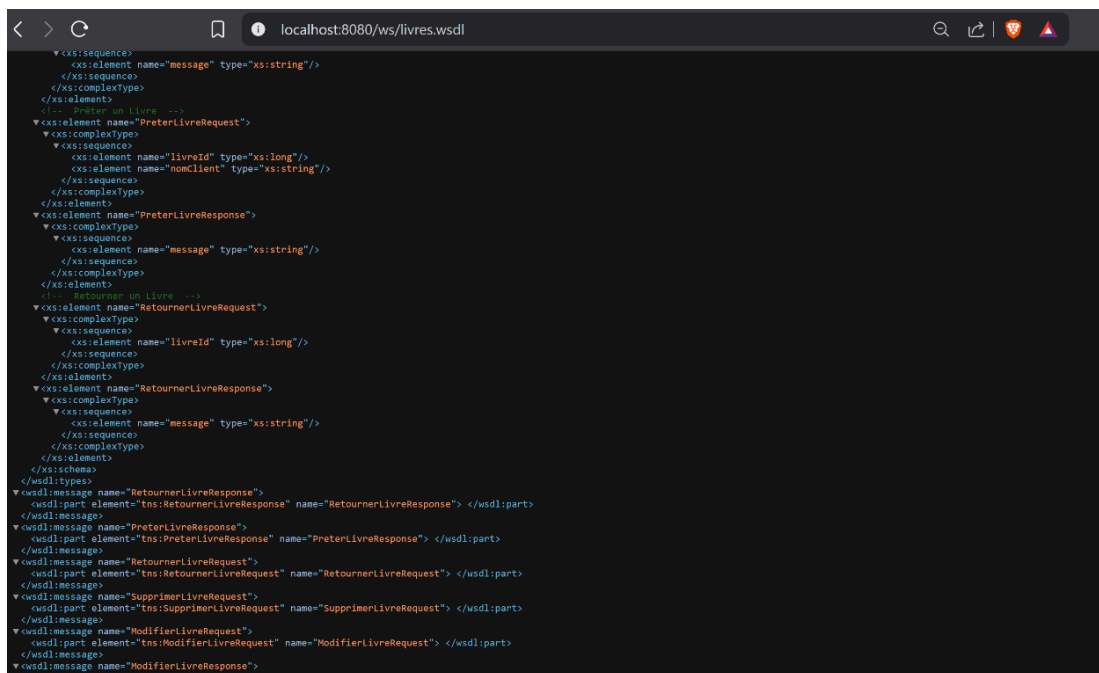
Après on va forcer à **maven** de télécharger les nouvelles dépendances avec cette commande :

./mvnw clean install

Puis on relance le server: **./mvnw spring-boot:run** et acceder à <http://localhost:8080/ws/livres.wsdl>



```
<?xml version='1.0' encoding='UTF-8'?>
<definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://sunubiblio.sn/soap" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://sunubiblio.sn/soap" targetNamespace="http://sunubiblio.sn/soap">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://sunubiblio.sn/soap">
      <!-- Ajouter un livre -->
      <xs:element name="AjouterLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="AjouterLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
            <xs:element name="disponible" type="xs:boolean"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- Modifier un livre -->
      <xs:element name="ModifierLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ModifierLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="message" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- Supprimer un livre -->
      <xs:element name="SupprimerLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SupprimerLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="message" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="AjouterLivreRequest" wsdl:part="AjouterLivreRequest" wsdl:required="true"/>
  <wsdl:message name="AjouterLivreResponse" wsdl:part="AjouterLivreResponse" wsdl:required="true"/>
  <wsdl:message name="ModifierLivreRequest" wsdl:part="ModifierLivreRequest" wsdl:required="true"/>
  <wsdl:message name="ModifierLivreResponse" wsdl:part="ModifierLivreResponse" wsdl:required="true"/>
  <wsdl:message name="SupprimerLivreRequest" wsdl:part="SupprimerLivreRequest" wsdl:required="true"/>
  <wsdl:message name="SupprimerLivreResponse" wsdl:part="SupprimerLivreResponse" wsdl:required="true"/>
  <wsdl:port name="ws" type="soap" binding="soap:binding" wsdl:required="true"/>
  <wsdl:binding name="ws" type="soap" wsdl:required="true"/>
  <wsdl:service name="ws" wsdl:required="true">
    <wsdl:operation name="AjouterLivre" wsdl:required="true" wsdl:documentation="Ajouter un livre" wsdl:input="AjouterLivreRequest" wsdl:output="AjouterLivreResponse"/>
    <wsdl:operation name="ModifierLivre" wsdl:required="true" wsdl:documentation="Modifier un livre" wsdl:input="ModifierLivreRequest" wsdl:output="ModifierLivreResponse"/>
    <wsdl:operation name="SupprimerLivre" wsdl:required="true" wsdl:documentation="Supprimer un livre" wsdl:input="SupprimerLivreRequest" wsdl:output="SupprimerLivreResponse"/>
  </wsdl:service>
</definitions>
```



```
<?xml version='1.0' encoding='UTF-8'?>
<definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://sunubiblio.sn/soap" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://sunubiblio.sn/soap" targetNamespace="http://sunubiblio.sn/soap">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://sunubiblio.sn/soap">
      <!-- Ajouter un livre -->
      <xs:element name="AjouterLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="AjouterLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
            <xs:element name="disponible" type="xs:boolean"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- Modifier un livre -->
      <xs:element name="ModifierLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
            <xs:element name="titre" type="xs:string"/>
            <xs:element name="auteur" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ModifierLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="message" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- Supprimer un livre -->
      <xs:element name="SupprimerLivreRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:long"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SupprimerLivreResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="message" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="AjouterLivreRequest" wsdl:part="AjouterLivreRequest" wsdl:required="true"/>
  <wsdl:message name="AjouterLivreResponse" wsdl:part="AjouterLivreResponse" wsdl:required="true"/>
  <wsdl:message name="ModifierLivreRequest" wsdl:part="ModifierLivreRequest" wsdl:required="true"/>
  <wsdl:message name="ModifierLivreResponse" wsdl:part="ModifierLivreResponse" wsdl:required="true"/>
  <wsdl:message name="SupprimerLivreRequest" wsdl:part="SupprimerLivreRequest" wsdl:required="true"/>
  <wsdl:message name="SupprimerLivreResponse" wsdl:part="SupprimerLivreResponse" wsdl:required="true"/>
  <wsdl:port name="ws" type="soap" binding="soap:binding" wsdl:required="true"/>
  <wsdl:binding name="ws" type="soap" wsdl:required="true"/>
  <wsdl:service name="ws" wsdl:required="true">
    <wsdl:operation name="AjouterLivre" wsdl:required="true" wsdl:documentation="Ajouter un livre" wsdl:input="AjouterLivreRequest" wsdl:output="AjouterLivreResponse"/>
    <wsdl:operation name="ModifierLivre" wsdl:required="true" wsdl:documentation="Modifier un livre" wsdl:input="ModifierLivreRequest" wsdl:output="ModifierLivreResponse"/>
    <wsdl:operation name="SupprimerLivre" wsdl:required="true" wsdl:documentation="Supprimer un livre" wsdl:input="SupprimerLivreRequest" wsdl:output="SupprimerLivreResponse"/>
  </wsdl:service>
</definitions>
```

```

</xs:schema>
</wsdl:types>
<wsdl:message name="RetournerLivreResponse">
  <wsdl:part element="tns:RetournerLivreResponse" name="RetournerLivreResponse"> </wsdl:part>
</wsdl:message>
<wsdl:message name="PreterLivreResponse">
  <wsdl:part element="tns:PreterLivreResponse" name="PreterLivreResponse"> </wsdl:part>
</wsdl:message>
<wsdl:message name="RetournerLivreRequest">
  <wsdl:part element="tns:RetournerLivreRequest" name="RetournerLivreRequest"> </wsdl:part>
</wsdl:message>
<wsdl:message name="SupprimerLivreRequest">
  <wsdl:part element="tns:SupprimerLivreRequest" name="SupprimerLivreRequest"> </wsdl:part>
</wsdl:message>
<wsdl:message name="ModifierLivreRequest">
  <wsdl:part element="tns:ModifierLivreRequest" name="ModifierLivreRequest"> </wsdl:part>
</wsdl:message>
<wsdl:message name="ModifierLivreResponse">
  <wsdl:part element="tns:ModifierLivreResponse" name="ModifierLivreResponse"> </wsdl:part>
</wsdl:message>
<wsdl:message name="SupprimerLivreResponse">
  <wsdl:part element="tns:SupprimerLivreResponse" name="SupprimerLivreResponse"> </wsdl:part>
</wsdl:message>
<wsdl:message name="AjouterLivreResponse">
  <wsdl:part element="tns:AjouterLivreResponse" name="AjouterLivreResponse"> </wsdl:part>
</wsdl:message>
<wsdl:message name="AjouterLivreRequest">
  <wsdl:part element="tns:AjouterLivreRequest" name="AjouterLivreRequest"> </wsdl:part>
</wsdl:message>
<wsdl:message name="PreterLivreRequest">
  <wsdl:part element="tns:PreterLivreRequest" name="PreterLivreRequest"> </wsdl:part>
</wsdl:message>
<wsdl:portType name="LivresPort">
  <wsdl:operation name="RetournerLivre">
    <wsdl:input message="tns:RetournerLivreRequest" name="RetournerLivreRequest"> </wsdl:input>
    <wsdl:output message="tns:RetournerLivreResponse" name="RetournerLivreResponse"> </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="PreterLivre">
    <wsdl:input message="tns:PreterLivreRequest" name="PreterLivreRequest"> </wsdl:input>
    <wsdl:output message="tns:PreterLivreResponse" name="PreterLivreResponse"> </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SupprimerLivre">
    <wsdl:input message="tns:SupprimerLivreRequest" name="SupprimerLivreRequest"> </wsdl:input>
    <wsdl:output message="tns:SupprimerLivreResponse" name="SupprimerLivreResponse"> </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ModifierLivre">
    <wsdl:input message="tns:ModifierLivreRequest" name="ModifierLivreRequest"> </wsdl:input>
    <wsdl:output message="tns:ModifierLivreResponse" name="ModifierLivreResponse"> </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="AjouterLivre">
    <wsdl:input message="tns:AjouterLivreRequest" name="AjouterLivreRequest"> </wsdl:input>
    <wsdl:output message="tns:AjouterLivreResponse" name="AjouterLivreResponse"> </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
</wsdl:service>

```

Figure 9 : Génération du WSDL

Après on va créer un dossier **dto** dans **soap**

Dans ce dossier on va créer les fichiers :

Soap/dto/AjouterLivreRequest.java

```

package sn.unchk.sunubiblio.soap.dto;

import jakarta.xml.bind.annotation.XmlAccessType;
import jakarta.xml.bind.annotation.XmlAccessorType;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlType;

@XmlRootElement(name = "AjouterLivreRequest")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "AjouterLivreRequest")
public class AjouterLivreRequest {

```

```

private String titre;
private String auteur;

// Getters and Setters
public String getTitre() {
    return titre;
}
public void setTitre(String titre) {
    this.titre = titre;
}
public String getAuteur() {
    return auteur;
}
public void setAuteur(String auteur) {
    this.auteur = auteur;
}
}

```

Soap/dto/AjouterLivreResponse.java

```

package sn.unchk.sunubiblio.soap.dto;

import jakarta.xml.bind.annotation.XmlAccessType;
import jakarta.xml.bind.annotation.XmlAccessorType;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlType;

@XmlRootElement(name = "AjouterLivreResponse")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "AjouterLivreResponse")
public class AjouterLivreResponse {

    private Long id;
    private String titre;
    private String auteur;
    private boolean disponible;

    // Getters and Setters
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
}

```



```
public String getTitre() {  
    return titre;  
}  
public void setTitre(String titre) {  
    this.titre = titre;  
}  
public String getAuteur() {  
    return auteur;  
}  
public void setAuteur(String auteur) {  
    this.auteur = auteur;  
}  
public boolean isDisponible() {  
    return disponible;  
}  
public void setDisponible(boolean disponible) {  
    this.disponible = disponible;  
}  
}
```

On va continuer avec les autres dto :

- ModifierLivreRequest
- ModifierLivreResponse
- SupprimerLivreRequest
- SupprimerLivreResponse
- PreterLivreRequest
- PreterLivreResponse
- RetournerLivreRequest
- RetournerLivreResponse

Une fois terminer on va relancer le server et copier : <http://localhost:8080/ws/livres.wsdl> et le tester sur **Soap UI** :

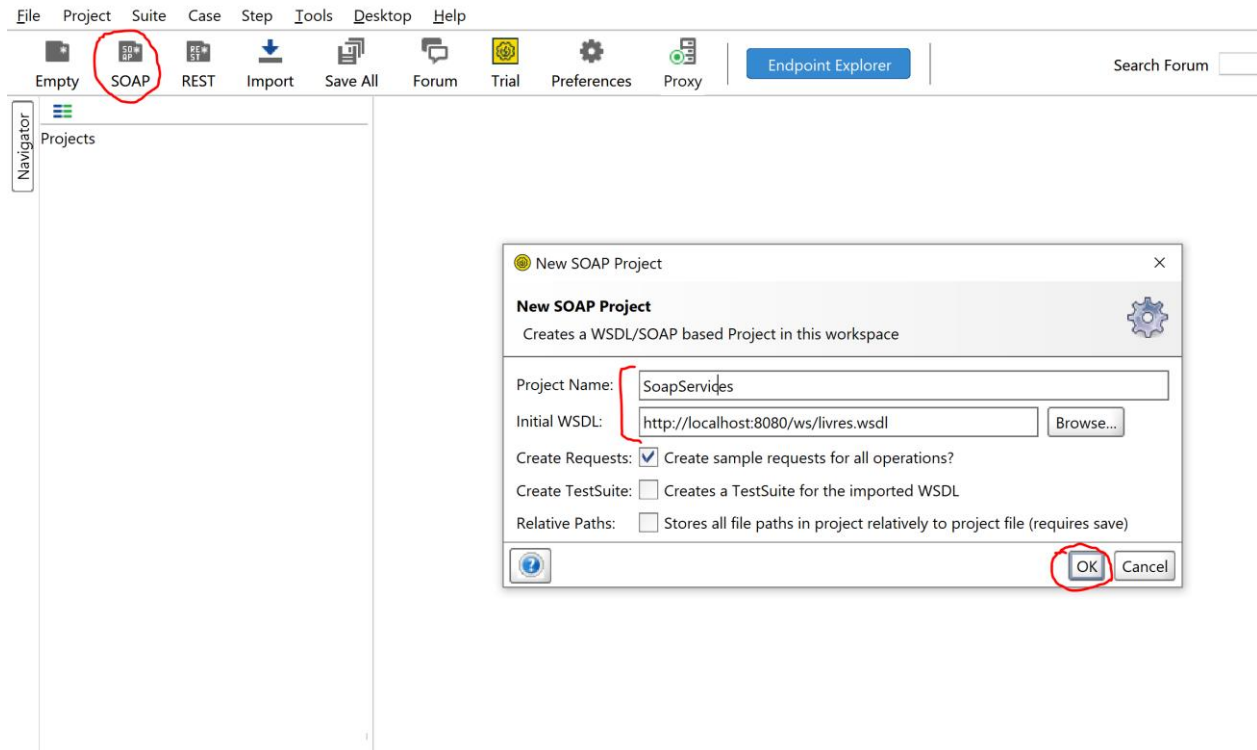


Figure 10 : Tester avec **Soap Ui** d'abord on va cliquer sur **SOAP** en haut à gauche puis coller <http://localhost:8080/ws/livres.wsdl> dans **initial WSDL** puis choisir un nom pour notre service puis cliquer sur **OK**.

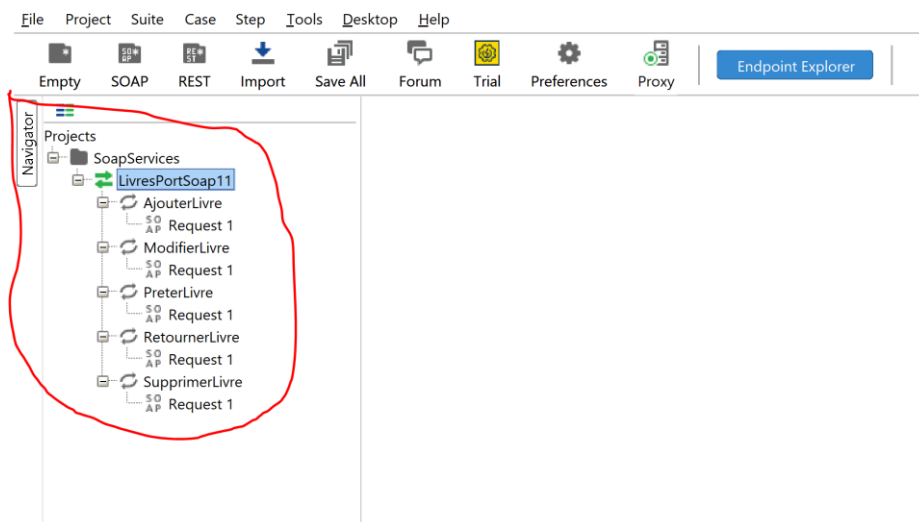


Figure 11 : Soap Ui avec nos dto

On peut tester le service maintenant !

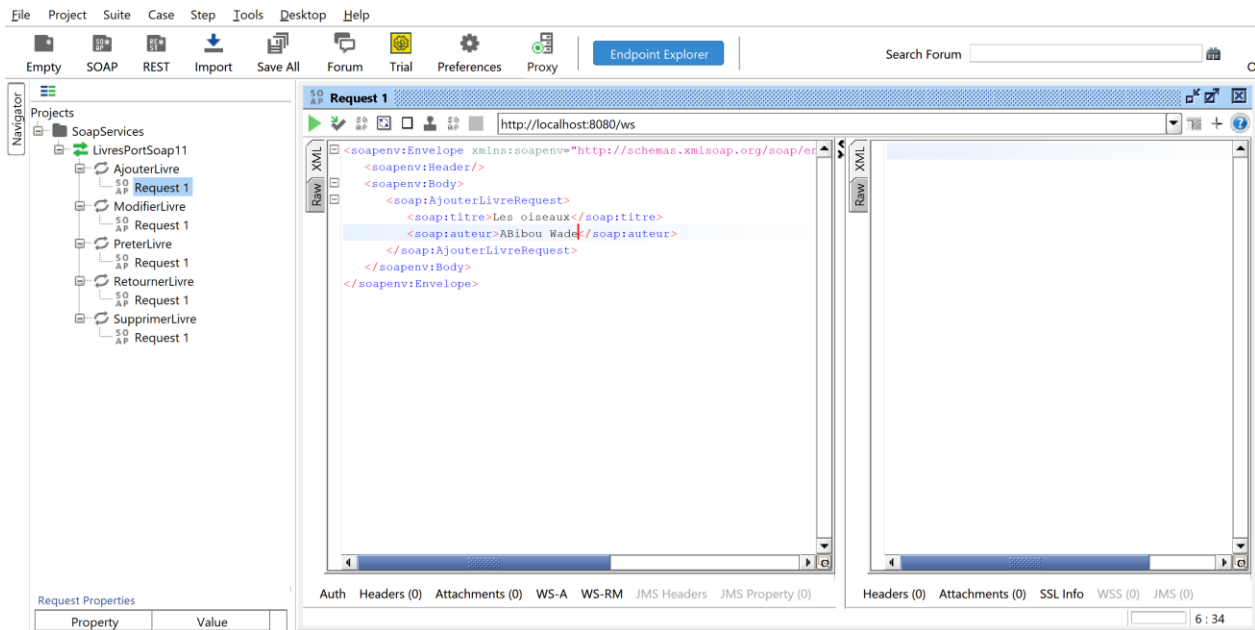


Figure 12 : Test d'ajout d'un livre avec un titre et nom d'auteur.

Conclusion

SunuBiblio est une application qui combine les deux styles de communication modernes : REST (JSON) et SOAP (XML) sur la même base de données, dans un projet Spring Boot léger. Ce projet prouve la capacité à développer des Web Services hybrides REST et SOAP à partir d'une architecture propre et modulaire.