TP : Utiliser la classe RestTemplate de Spring

Architecture des composants d’entreprise

Table des matières

I. Objectif du TP

- Développer la couche back-end avec JDK 17, Spring Boot 3.*, Spring Rest, JPA et H2.
- Développer la couche front-end avec JDK 17, Spring Boot 3.*, Spring Rest et JSP.
- Utiliser les services fournis par la classe RestTemplate de Spring pour consommer le service web fourni par la couche back-end.

II. Prérequis

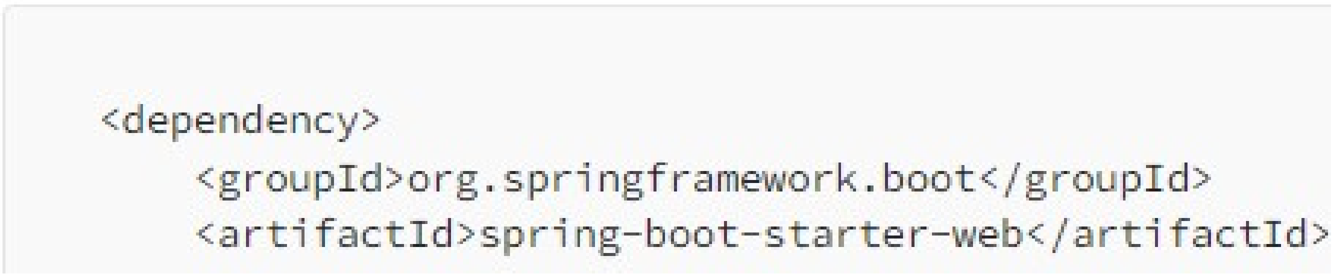
- IntelliJ IDEA ;
- JDK version 17 ;
- Une connexion Internet pour permettre à Maven de télécharger les librairies.

NB : *Ce TP a été réalisé avec IntelliJ IDEA 2023.2.3 (Ultimate Edition).*

III. La classe RestTemplate

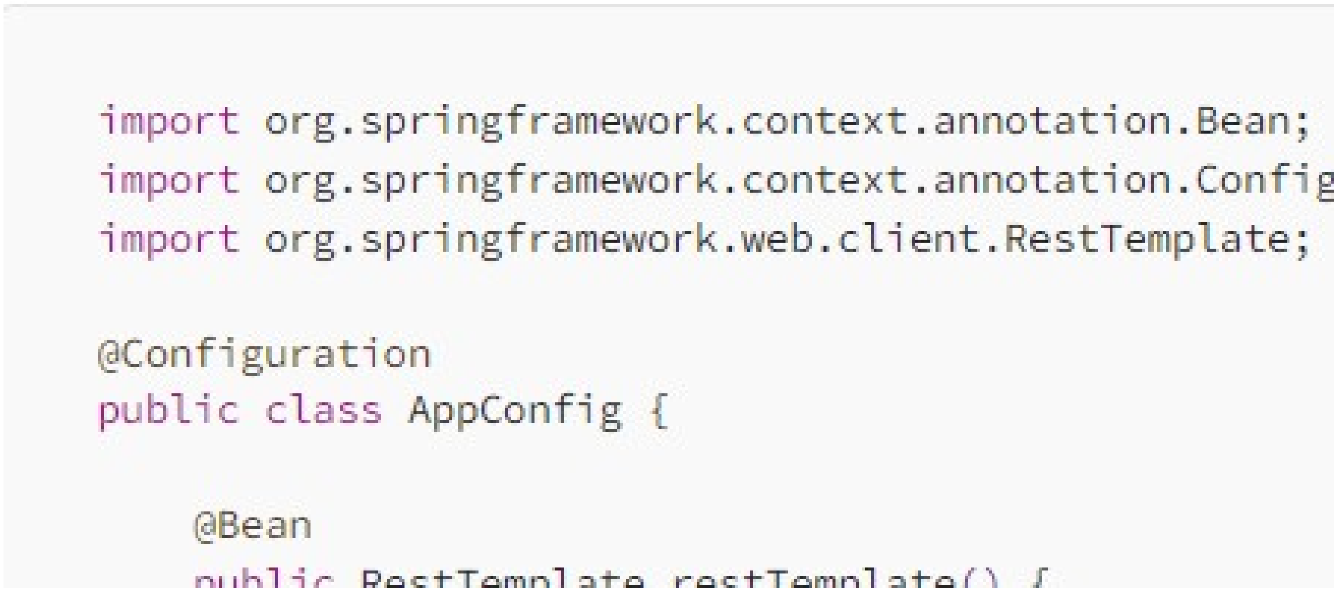
❖ ***RestTemplate*** est une classe fournie par le Framework Spring qui simplifie le processus d'exécution des requêtes HTTP et de gestion des réponses. Il fait abstraction d'une grande partie du code passe-partout généralement associé aux appels HTTP, ce qui facilite l'interaction avec les services Web RESTful.

❖ Pour utiliser, vous devez inclure la dépendance Spring Web dans votre fichier si vous utilisez Maven :



```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
```

❖ Pour créer un bean RestTemplate, vous pouvez définir un bean dans votre classe de configuration Spring afin qu'il puisse être injecté partout où cela est nécessaire :



```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Config
import org.springframework.web.client.RestTemplate;

@Configuration
public class AppConfig {

    @Bean
    public RestTemplate restTemplate() {
```

IV. Développement de la couche Back-end

a. Création du projet Maven

- Au niveau d’Intellig ou autre IDE, créer un nouveau projet Maven (resttemplate-back).

b. Le fichier pom.xml

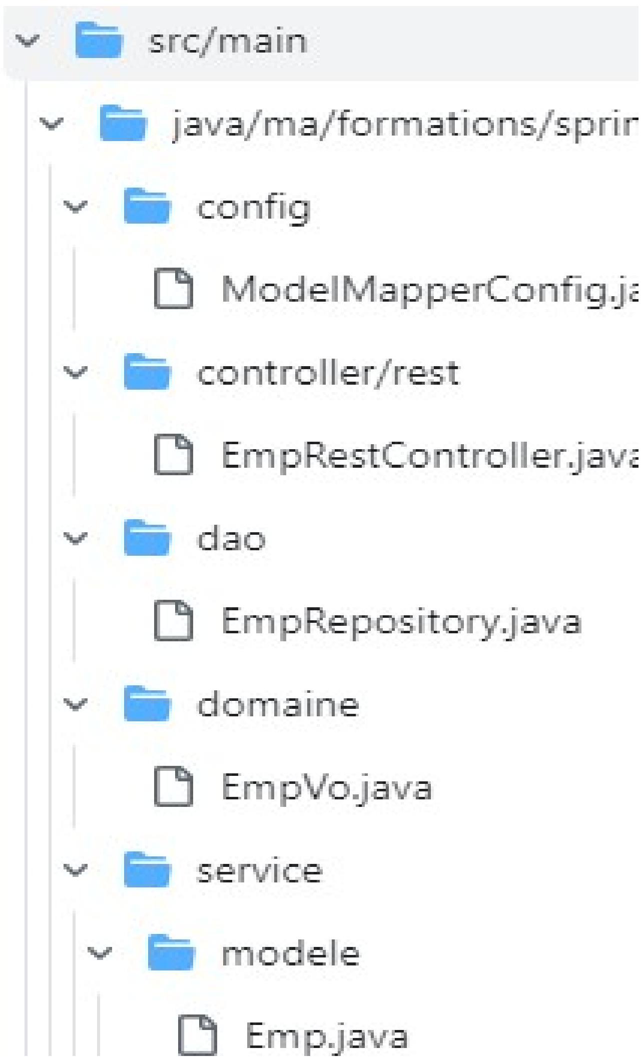
- Le contenu du fichier pom.xml est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://maven.apache.org/POM/4.0.0"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.6</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>ma.formations.spring</groupId>
  <artifactId>resttemplate-back</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <!-- <packaging>war</packaging> -->
  <name>resttemplate-back</name>
  <description>Rest Template Back end example</description>

  <properties>
    <java.version>17</java.version>
    <class>ma.formations.spring.rest.MainApplication</class>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
```

```
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
         <groupId>com.h2database</groupId>
         <artifactId>h2</artifactId>
         <scope>runtime</scope>
    </dependency>

    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-xml</artifactId>
    </dependency>

    <dependency>
        <groupId>org.modelmapper</groupId>
        <artifactId>modelmapper</artifactId>
        <version>3.2.1</version>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

- Effecteur un « ***clean install*** » pour que Maven puisse télécharger les dépendances.

c. L'arborescence du projet



d. Le fichier application.properties

```
server.port=9090
# pour consulet H2 via le le Console sur le navigateur:
# http://localhost:9090/h2-console
# Enabling H2 Console
spring.h2.console.enabled=true
# ===============================
# DB
# ===============================
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# ===============================
# JPA / HIBERNATE
# ===============================
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
```

e. La classe Emp

```
package ma.formations.spring.rest.service.modele;

import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;
import java.util.Date;

@Entity @NoArgsConstructor @Data
public class Emp implements Serializable {
   @GeneratedValue @Id
   private Long id;

   @Column(name = "NAME", unique = true, length = 30)
   private String firstName;

   private Double salaire;

   private String fonction;

   @Transient
   private Date dateAnniversaire;

   public Emp(String name, Double salary, String fonction) {
      this.firstName = name;
      this.salaire = salary;
      this.fonction = fonction;
   }

}
```

f. La classe EmpVo

```
package ma.formations.spring.rest.domaine;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;
import java.util.Date;

//Value Object (VO) <==> DTO : Data Transfer Object
//POJO : Plain Old Java Object
@NoArgsConstructor
@AllArgsConstructor
```

```
@Data
@Builder
public class EmpVo implements Serializable {
   private Long id;
   private String firstName;
   private Double salaire;
   private String fonction;
   private Date dateAnniversaire;
}
```

g. La classe ModelMapperConfig

```
package ma.formations.spring.rest.config;

import org.modelmapper.ModelMapper;
import org.modelmapper.convention.MatchingStrategies;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ModelMapperConfig {
   @Bean
   public ModelMapper modelMapper() {
      ModelMapper modelMapper = new ModelMapper();
      modelMapper.getConfiguration().setMatchingStrategy(MatchingStrategies.LOOSE);
      return modelMapper;
   }
}
```

h. L'interface EmpRepository

```
package ma.formations.spring.rest.dao;

import ma.formations.spring.rest.service.modele.Emp;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.List;

public interface EmpRepository extends JpaRepository<Emp, Long> {
   List<Emp> findBySalaire(Double salary);
   List<Emp> findByFonction(String designation);
   List<Emp> findBySalaireAndFonction(Double salary, String fonction);
   List<Emp> findByFirstName(String name);
   @Query("SELECT e From Emp e where e.salaire=(select MAX(ee.salaire) as salaire FROM Emp ee)")
   Emp getEmpHavaingMaxSalary();
}
```

i. L’interface IService

```
package ma.formations.spring.rest.service;

import java.util.List;
import ma.formations.spring.rest.domaine.EmpVo;

public interface IService {
            List<EmpVo> getEmployees();
            void save(EmpVo emp);
            EmpVo getEmpById(Long id);
            void delete(Long id);
            List<EmpVo> findBySalary(Double salary);
            List<EmpVo> findEmployeesByName(String name);
            List<EmpVo> findByFonction(String designation);
            List<EmpVo> findBySalaryAndFonction(Double salary, String fonction);
            EmpVo getEmpHavaingMaxSalary();
            // Pour la pagination
            List<EmpVo> findAll(int pageId, int size);
            // pour le tri
            List<EmpVo> sortBy(String... fieldName);
            void deleteAll();
}
```

j. La classe ServiceImpl

```
package ma.formations.spring.rest.service;

import lombok.AllArgsConstructor;
import ma.formations.spring.rest.dao.EmpRepository;
import ma.formations.spring.rest.domaine.EmpVo;
import ma.formations.spring.rest.service.modele.Emp;
import org.modelmapper.ModelMapper;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
@AllArgsConstructor
public class ServiceImpl implements IService {
    private EmpRepository empRepository;
    private ModelMapper modelMapper;

    @Override
    public List<EmpVo> findEmployeesByName(String name) {
        List<Emp> list = empRepository.findByFirstName(name);
        return list.stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
```

```
}

@Override
public List<EmpVo> getEmployees() {
   List<Emp> list = empRepository.findAll();
   return list.stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
}

@Transactional
@Override
public void save(EmpVo vo) {
   empRepository.save(modelMapper.map(vo, Emp.class));
}

@Override
public EmpVo getEmpById(Long id) {
   boolean trouve = empRepository.existsById(id);
   if (!trouve)
      return null;
   return modelMapper.map(empRepository.getOne(id), EmpVo.class);
}

@Transactional
@Override
public void delete(Long id) {
   empRepository.deleteById(id);
}

@Override
public List<EmpVo> findBySalary(Double salaty) {
   List<Emp> list = empRepository.findBySalaire(salaty);
   return list.stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
}

@Override
public List<EmpVo> findByFonction(String fonction) {
   List<Emp> list = empRepository.findByFonction(fonction);
   return list.stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
}

@Override
public List<EmpVo> findBySalaryAndFonction(Double salary, String fonction) {
   List<Emp> list = empRepository.findBySalaireAndFonction(salary, fonction);
   return list.stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
}

@Override
public EmpVo getEmpHavaingMaxSalary() {

   return modelMapper.map(empRepository.getEmpHavaingMaxSalary(), EmpVo.class);
}

@Override
public List<EmpVo> findAll(int pageId, int size) {
   //Page<Emp> result = empRepository.findAll(PageRequest.of(pageId, size));

   //Page<Emp> result = empRepository.findAll(PageRequest.of(pageId, size, Direction.DESC, "salaire","firstName"));
```

```
        //Page<Emp> result = empRepository.findAll(PageRequest.of(pageId, size,Direction.ASC,"salaire","fonction","id"));
        Page<Emp> result = empRepository.findAll(PageRequest.of(pageId, size, Sort.by("salaire")));
        return result.getContent().stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
    }

    @Override
    public List<EmpVo> sortBy(String... fieldNames) {
        return Sort.by(fieldNames).stream().map(bo -> modelMapper.map(bo, EmpVo.class)).toList();
    }

    @Transactional
    @Override
    public void deleteAll() {
        empRepository.deleteAll();
    }

}
```

k. La classe EmpRestController

```
package ma.formations.spring.rest.controller.rest;

import jakarta.validation.Valid;
import lombok.AllArgsConstructor;
import ma.formations.spring.rest.domaine.EmpVo;
import ma.formations.spring.rest.service.IService;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin(origins = "*", maxAge = 3600)
@AllArgsConstructor
public class EmpRestController {

    private IService service;

    @GetMapping(value = "/rest/emp", produces = {MediaType.APPLICATION_XML_VALUE,
MediaType.APPLICATION_JSON_VALUE})
    public List<EmpVo> getAll() {
        return service.getEmployees();
    }

    @GetMapping(value = "/rest/emp/id/{id}")
    public ResponseEntity<Object> getEmpById(@PathVariable(value = "id") Long empVoId) {
        EmpVo empVoFound = service.getEmpById(empVoId);
        if (empVoFound == null)
            return new ResponseEntity<>("employee doen't exist", HttpStatus.OK);
        return new ResponseEntity<>(empVoFound, HttpStatus.OK);
    }
    @GetMapping(value = "/rest/emp/name/{name}", produces = {MediaType.APPLICATION_XML_VALUE,
MediaType.APPLICATION_JSON_VALUE})
    public List<EmpVo> getAll(@PathVariable(value = "name") String name) {
        return service.findEmployeesByName(name);
    }

    @PostMapping(value = "/rest/emp")
    public ResponseEntity<Object> createEmp(@Valid @RequestBody EmpVo empVo) {
        service.save(empVo);
        return new ResponseEntity<>("employee is created successfully", HttpStatus.CREATED);
    }

    @PutMapping(value = "/rest/emp/{id}")
    public ResponseEntity<Object> updateEmp(@PathVariable(name = "id") Long empVoId, @RequestBody EmpVo
empVo) {
        EmpVo empVoFound = service.getEmpById(empVoId);
        if (empVoFound == null)
            return new ResponseEntity<>("employee doen't exist", HttpStatus.OK);
        empVo.setId(empVoId);
        service.save(empVo);
        return new ResponseEntity<>("Employee is updated successsfully", HttpStatus.OK);
    }

    @DeleteMapping(value = "/rest/emp/{id}")
    public ResponseEntity<Object> deleteEmp(@PathVariable(name = "id") Long empVoId) {
        EmpVo empVoFound = service.getEmpById(empVoId);
        if (empVoFound == null)
            return new ResponseEntity<>("employee doen't exist", HttpStatus.OK);
        service.delete(empVoId);
        return new ResponseEntity<>("Employee is deleted successsfully", HttpStatus.OK);
    }

    @DeleteMapping(value = "/rest/emp")
    public ResponseEntity<Object> deleteAll() {
        service.deleteAll();
        return new ResponseEntity<>("All employees are deleted successsfully", HttpStatus.OK);
    }

    @GetMapping(value = "/rest/sort/{fieldName}", produces = {MediaType.APPLICATION_XML_VALUE,
MediaType.APPLICATION_JSON_VALUE})
    public List<EmpVo> sortBy(@PathVariable String fieldName) {
        return service.sortBy(fieldName);
    }

    @GetMapping(value = "/rest/sort/{fieldName1}/{fieldName2}", produces = {MediaType.APPLICATION_XML_VALUE,
MediaType.APPLICATION_JSON_VALUE})
    public List<EmpVo> sortByBis(@PathVariable String fieldName1, @PathVariable String fieldName2) {
        return service.sortBy(fieldName1, fieldName2);
    }

    @GetMapping("/rest/pagination/{pageid}/{size}")
    public List<EmpVo> pagination(@PathVariable int pageid, @PathVariable int size, Model m) {
        return service.findAll(pageid, size);
    }
}
```

I. La classe MainApplication

```
package ma.formations.spring.rest;

import ma.formations.spring.rest.domaine.EmpVo;
import ma.formations.spring.rest.service.IService;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class MainApplication extends SpringBootServletInitializer {

  public static void main(String[] args) {
    SpringApplication.run(MainApplication.class, args);
    System.out.println("Application démarrée");
  }

  @Override
  protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
    return application.sources(MainApplication.class);
  }

  @Bean
  public CommandLineRunner initDatabase(IService service) {
    return args -> {
      service.deleteAll();
      service.save(EmpVo.builder().firstName("ALAMI").salaire(10000D).fonction("INGENIEUR").build());
      service.save(EmpVo.builder().firstName("amrani").salaire(200000D).fonction("DIRECTEUR").build());
      service.save(EmpVo.builder().firstName("Jamali").salaire(3000D).fonction("Technicien").build());
      service.save(EmpVo.builder().firstName("KAOUTAR").salaire(20000D).fonction("INGENIEUR").build());
    };
  }

}
```

V. Développement de la couche Front-end

a. Création du projet Maven

- Au niveau d'Intellig ou autre IDE, créer un nouveau projet Maven (resttemplate-front).

b. Le fichier pom.xml

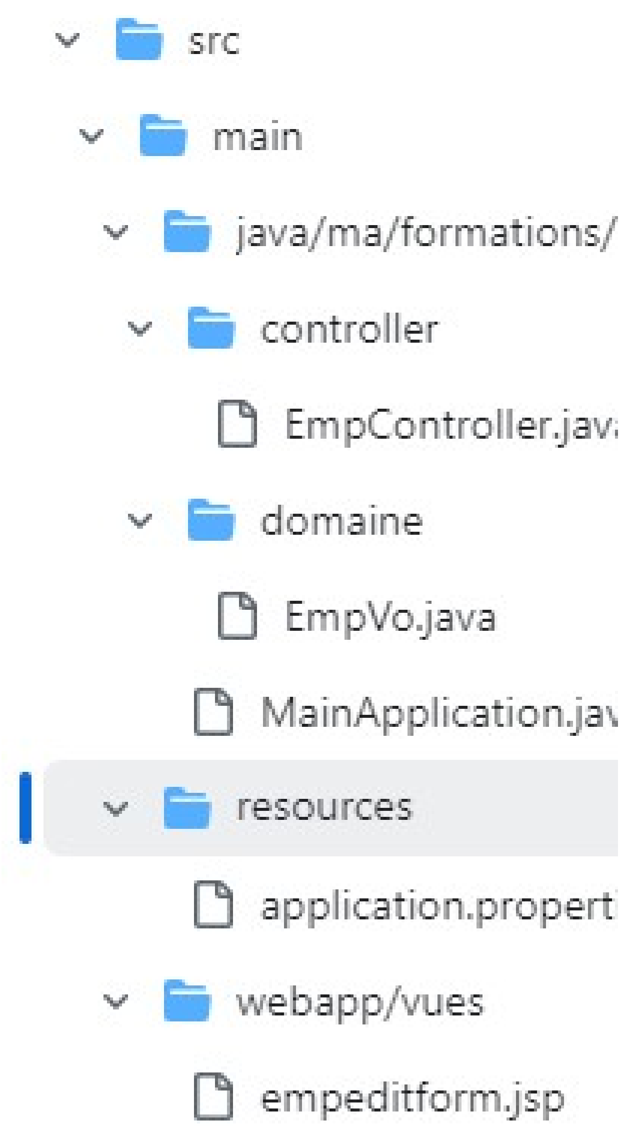
- Le contenu du fichier pom.xml est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>3.3.6</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>ma.formations.spring</groupId>
        <artifactId>resttemplate-front</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <name>resttemplate-front</name>
        <description>Using RestTemplate class</description>
        <properties>
                <java.version>17</java.version>
                <class>ma.formations.spring.rest.MainApplication</class>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-tomcat</artifactId>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <optional>true</optional>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
                <!-- Pour pouvoir utiliser JSP,les dépendances suivantes sont nécessaires -->

                        <groupId>jakarta.servlet.jsp.jstl</groupId>
                        <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
                        <version>3.0.0</version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.web</groupId>
                        <artifactId>jakarta.servlet.jsp.jstl</artifactId>
                        <version>3.0.1</version>
                </dependency>
                <dependency>
                        <groupId>org.apache.tomcat.embed</groupId>
                        <artifactId>tomcat-embed-jasper</artifactId>
                </dependency>
                </dependencies>
```

```
		<build>
			<plugins>
				<plugin>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-maven-plugin</artifactId>
				</plugin>
			</plugins>
		</build>
</project>
```

c. L'arborescence du projet



```
src
  main
    java/ma/formations/
      controller
        EmpController.jav
      domaine
        EmpVo.java
      MainApplication.jav
    resources
      application.propert
    webapp/vues
      empeditform.jsp
```

d. Le fichier application.properties

```
#Pour Spring MVC :
spring.mvc.view.prefix=/vues/
spring.mvc.view.suffix=.jsp
```

```
server.port=9191
#Le serveur Rest :
server.rest.url=http://localhost:9090/rest/emp
```

e. La classe EmpVo

```java
package ma.formations.spring.rest.domaine;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;
import java.util.Date;

@NoArgsConstructor
@AllArgsConstructor
@Data
public class EmpVo implements Serializable {
        private Long id;
        private String firstName;
        private Double salaire;
        private String fonction;
        private Date dateAnniversaire;
}
```

f. La classe MainApplication

```java
package ma.formations.spring.rest;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class MainApplication extends SpringBootServletInitializer {

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(MainApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(MainApplication.class, args);
        System.out.println("Application démarrée");
    }
}
```

g. La classe EmpController

```
package ma.formations.spring.rest.controller;

import ma.formations.spring.rest.domaine.EmpVo;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.client.RestTemplate;

import java.util.Arrays;
import java.util.List;

@Controller
public class EmpController {

  private final RestTemplate restTemplate;
  @Value("${server.rest.url}")
  private String url;
  public EmpController(RestTemplate restTemplate) {
    this.restTemplate = restTemplate;
  }
  @RequestMapping("/")
  public String showWelcomeFile(Model m) {
    return "index";
  }

  @RequestMapping("/empform")
  public String showform(Model m) {
    m.addAttribute("empVo", new EmpVo());
    return "empform";
  }

  @RequestMapping(value = "/save", method = RequestMethod.POST)
  public String save(@ModelAttribute("empVo") EmpVo emp) {

    // HttpHeaders
    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(List.of(MediaType.APPLICATION_JSON));
    // Request to return JSON format
    headers.setContentType(MediaType.APPLICATION_JSON);
    HttpEntity<EmpVo> entity = new HttpEntity<EmpVo>(emp, headers);

    ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.POST, entity,
String.class);
```

```
        HttpStatusCode statusCode = response.getStatusCode();
        System.out.println("Response Satus Code: " + statusCode);

        return "redirect:/viewemp";// will redirect to viewemp request mapping
    }

    @RequestMapping("/viewemp")
    public String viewemp(Model m) {
        // HttpHeaders
        EmpVo[] list = null;
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(List.of(MediaType.APPLICATION_JSON));
        // Request to return JSON format
        headers.setContentType(MediaType.APPLICATION_JSON);
        // HttpEntity<String>: To get result as String.
        HttpEntity<EmpVo[]> entity = new HttpEntity<EmpVo[]>(headers);

        // Send request with GET method, and Headers.
        ResponseEntity<EmpVo[]> response = restTemplate.exchange(url, HttpMethod.GET, entity,
EmpVo[].class);

        HttpStatusCode statusCode = response.getStatusCode();
        System.out.println("Response Satus Code: " + statusCode);

        if (statusCode == HttpStatus.OK)
            list = response.getBody();
        m.addAttribute("list", Arrays.asList(list));
        return "viewemp";
    }

    @RequestMapping(value = "/editemp/{id}")
    public String edit(@PathVariable Long id, Model m) {
        // HttpHeaders
        EmpVo emp = null;
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(List.of(MediaType.APPLICATION_JSON));
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<EmpVo> entity = new HttpEntity<EmpVo>(headers);

        ResponseEntity<EmpVo> response = restTemplate.exchange(url + "/id/" + id, HttpMethod.GET,
entity, EmpVo.class);

        HttpStatusCode statusCode = response.getStatusCode();
        System.out.println("Response Satus Code: " + statusCode);

        if (statusCode == HttpStatus.OK)
            emp = response.getBody();
        m.addAttribute("empVo", emp);
        return "empeditform";
    }

    @RequestMapping(value = "/editsave", method = RequestMethod.POST)
    public String editsave(@ModelAttribute("empVo") EmpVo emp) {
        // HttpHeaders
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(List.of(MediaType.APPLICATION_JSON));
        // Request to return JSON format
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<EmpVo> entity = new HttpEntity<EmpVo>(emp, headers);
        ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.POST, entity,
    String.class);
        HttpStatusCode statusCode = response.getStatusCode();
        System.out.println("Response Satus Code: " + statusCode);

        return "redirect:/viewemp";
      }
      @RequestMapping(value = "/deleteemp/{id}", method = RequestMethod.GET)
      public String delete(@PathVariable Long id) {
        // HttpHeaders
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(List.of(MediaType.APPLICATION_JSON));
        // Request to return JSON format
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<EmpVo> entity = new HttpEntity<EmpVo>(headers);
        ResponseEntity<String> response = restTemplate.exchange(url + "/" + id, HttpMethod.DELETE,
    entity, String.class);
        HttpStatusCode statusCode = response.getStatusCode();
        System.out.println("Response Satus Code: " + statusCode);
        return "redirect:/viewemp";
      }
    }
```

h. La page index.jsp

```
<a href="empform">Add Employee</a>
<a href="viewemp">View Employees</a>
```

i. La page empform.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<h1>Add New Employee</h1>
<form:form method="post" action="save" modelAttribute="empVo">
    <table>
        <tr>
                <td>Name :</td>
                <td><form:input path="firstName" /></td>
        </tr>
        <tr>
                <td>Salary :</td>
                <td><form:input path="salaire" /></td>
        </tr>
        <tr>
                <td>Fonction :</td>
                <td><form:input path="fonction" /></td>
        </tr>
        <tr>
                <td></td>
                <td><input type="submit" value="Save" /></td>
        </tr>
    </table>
</form:form>
```

j. La page empeditform.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
    <h1>Edit Employee</h1>
   <form:form method="POST" action="/editsave" modelAttribute="empVo">
    <table >
    <tr>
    <td></td>
     <td><form:hidden  path="id" /></td>
     </tr>
     <tr>
      <td>Name : </td>
      <td><form:input path="firstName"  /></td>
     </tr>
     <tr>
      <td>Salary :</td>
      <td><form:input path="salaire" /></td>
     </tr>
     <tr>
      <td>Fonction :</td>
      <td><form:input path="fonction" /></td>
     </tr>
     <tr>
      <td> </td>
      <td><input type="submit" value="Edit Save" /></td>
     </tr>
    </table>
   </form:form>
```

k. La page viewemp.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<h1>Employees List</h1>
<table border="2" width="70%" cellpadding="2">
	<tr>
		<th>Id</th>
		<th>Name</th>
		<th>Salary</th>
		<th>Fonction</th>
		<th>Edit</th>
		<th>Delete</th>
	</tr>
	<c:forEach var="empVo" items="${list}">
		<tr>
			<td>${empVo.id}</td>
			<td>${empVo.firstName}</td>
			<td>${empVo.salaire}</td>
			<td>${empVo.fonction}</td>
			<td><a href="editemp/${empVo.id}">Edit</a></td>
			<td><a href="deleteemp/${empVo.id}">Delete</a></td>
		</tr>
	</c:forEach>
</table>
<br />
<a href="empform">Add New Employee</a>
```