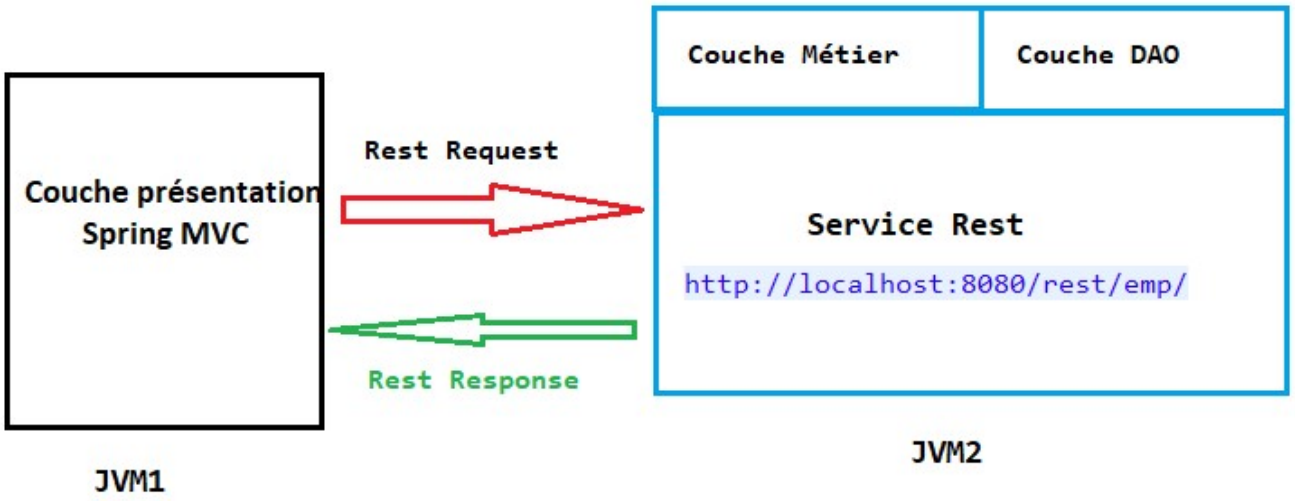
TP N°5 : Développement d'une application web avec Spring Boot et Spring MVC et utilisation de RestTemplate pour l'invocation d'un SW distribué

SOMMAIRE

I- Objectifs :

- ✓ Développer un projet web avec Spring MVC et invocation du service web http://localhost:8080/rest/emp/ réalisé en TP n°4.
- ✓ Apprendre à utiliser la classe RestTemplate de Spring.
- ✓ Utiliser Spring Boot pour créer et configurer facilement l'application.



II- Outils utilisés :

Dans ce TP, nous allons utiliser les outils suivants :

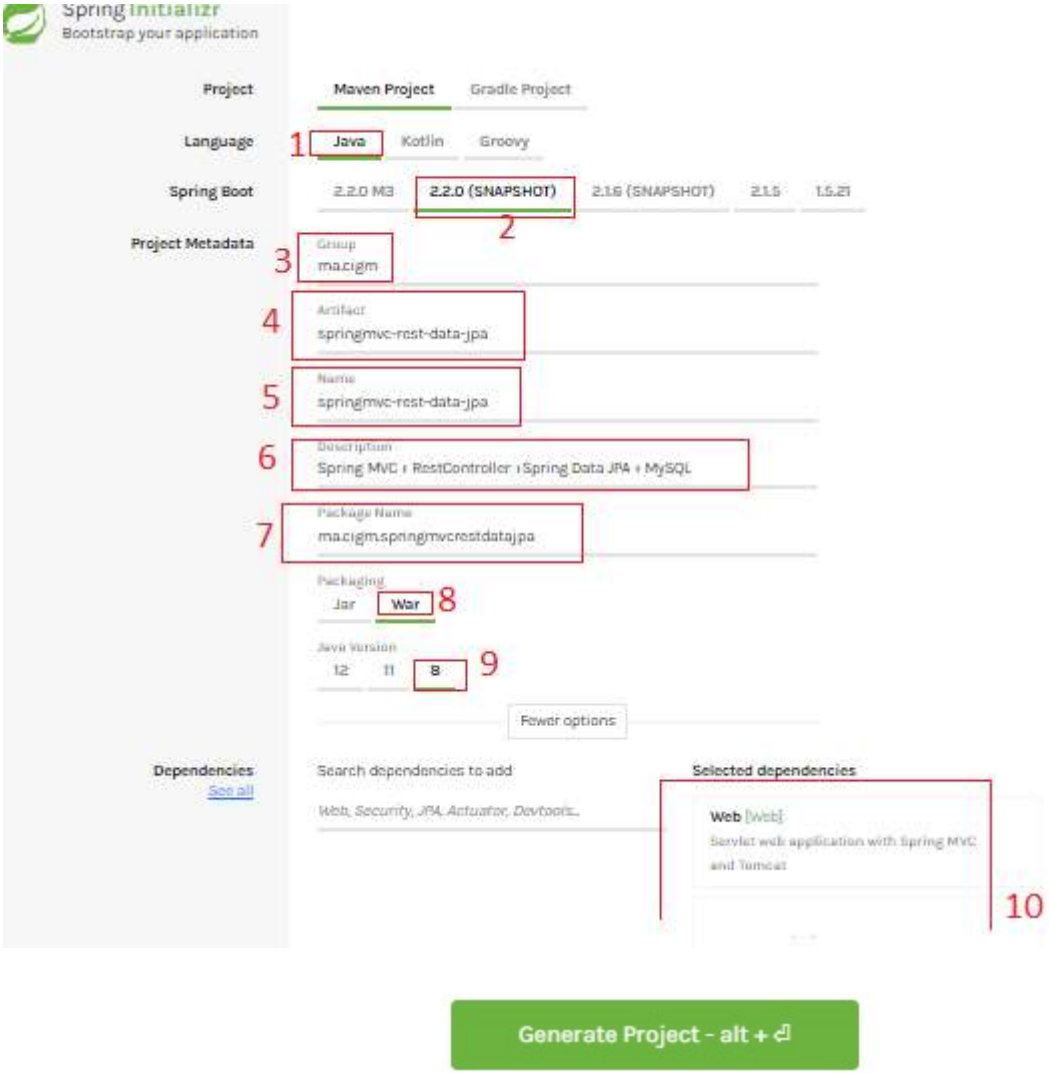
- ✓ Eclipse Mars (ou autre) avec le plugin Maven 3.x ;
- ✓ JDK 1.8 ;
- ✓ Connection à Internet pour permettre à Maven de télécharger les dépendances nécessaires (Spring Boot 2.2.0, …).

Le TP n°4 est un prérequis pour ce présent atelier.

III- Développement de l'application

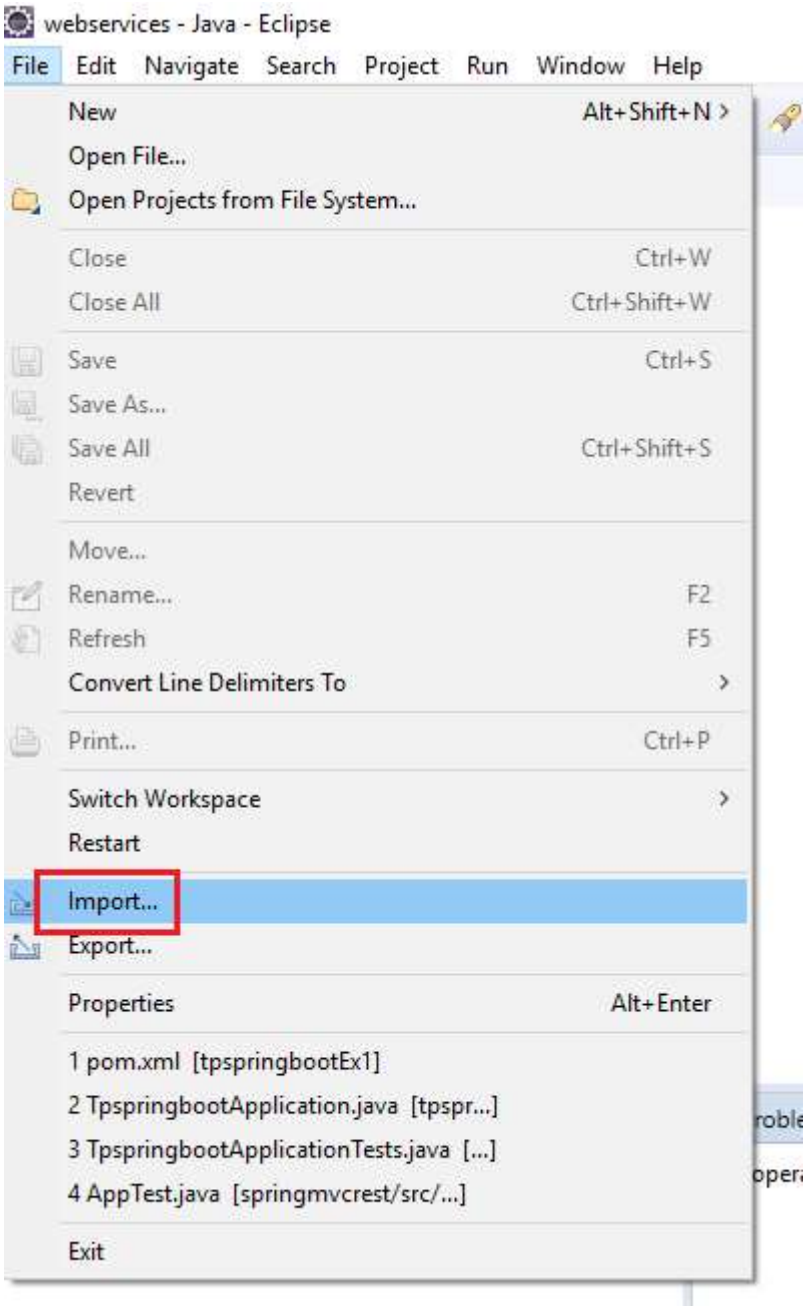
1. Génération du squelette de l'application moyennant Spring Initializr

- Aller au site : https://start.spring.io/ :

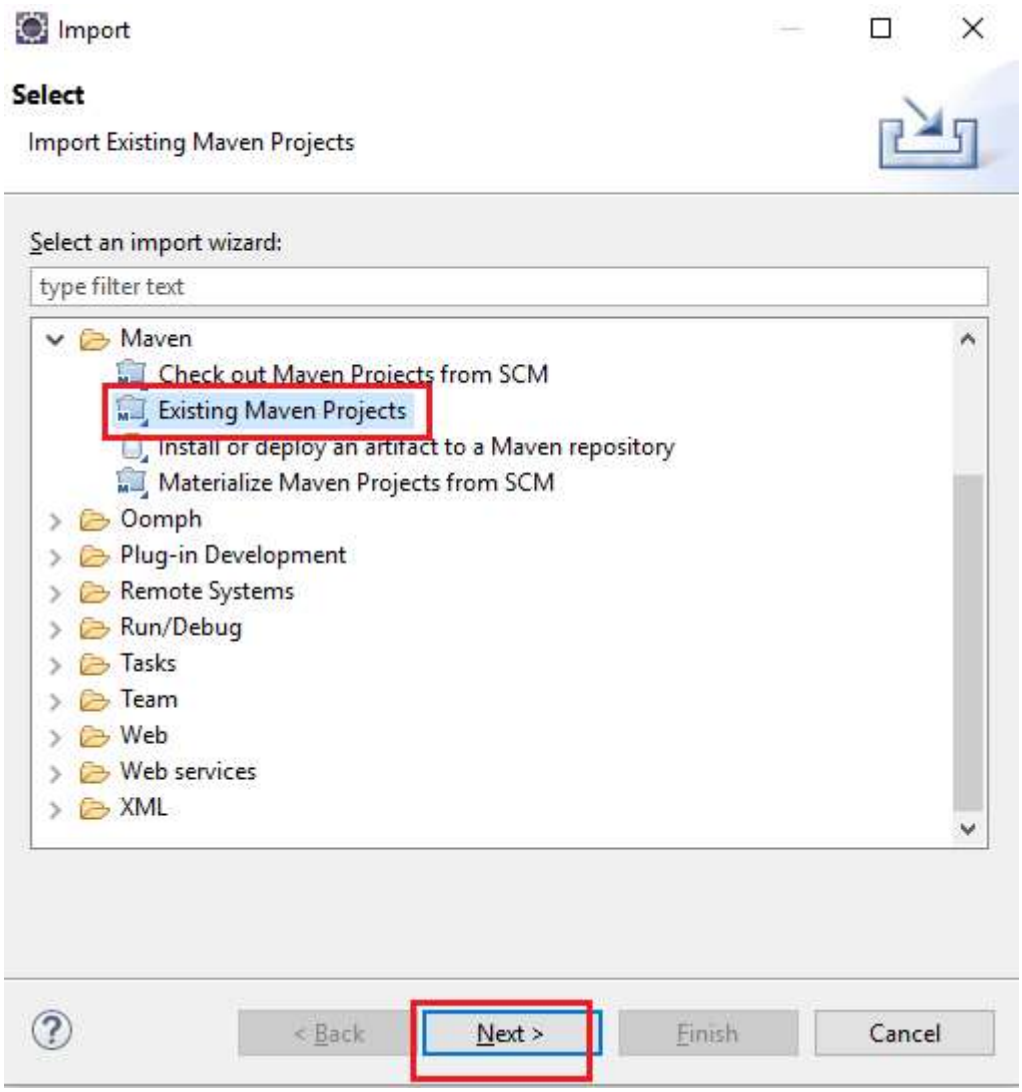


- 1 : Choisir « Maven Project » et Java dans Language.
- 2 : Choisir la version 2.2.0 de Spring Boot.
- 3 : Entrer le group (ma.cigma).
- 4 : Entrer l'artifact (springboot-client-mvc)
- 5 : Entrer le nom (springboot-client-mvc).
- 6 : Entrer la description de votre projet (Spring Boot + Spring MVC + RestTemplate).
- 7 : Entrer le nom du package racine (ma.cigma.springbootclientmvc).
- 8 : Choisir War.
- 9 : Choisir 8 dans la version de Java.
- 10 : Ajouter la dépendance : web.
- Enfin, cliquer sur le bouton « Generate Project » pour générer le fichier ZIP.

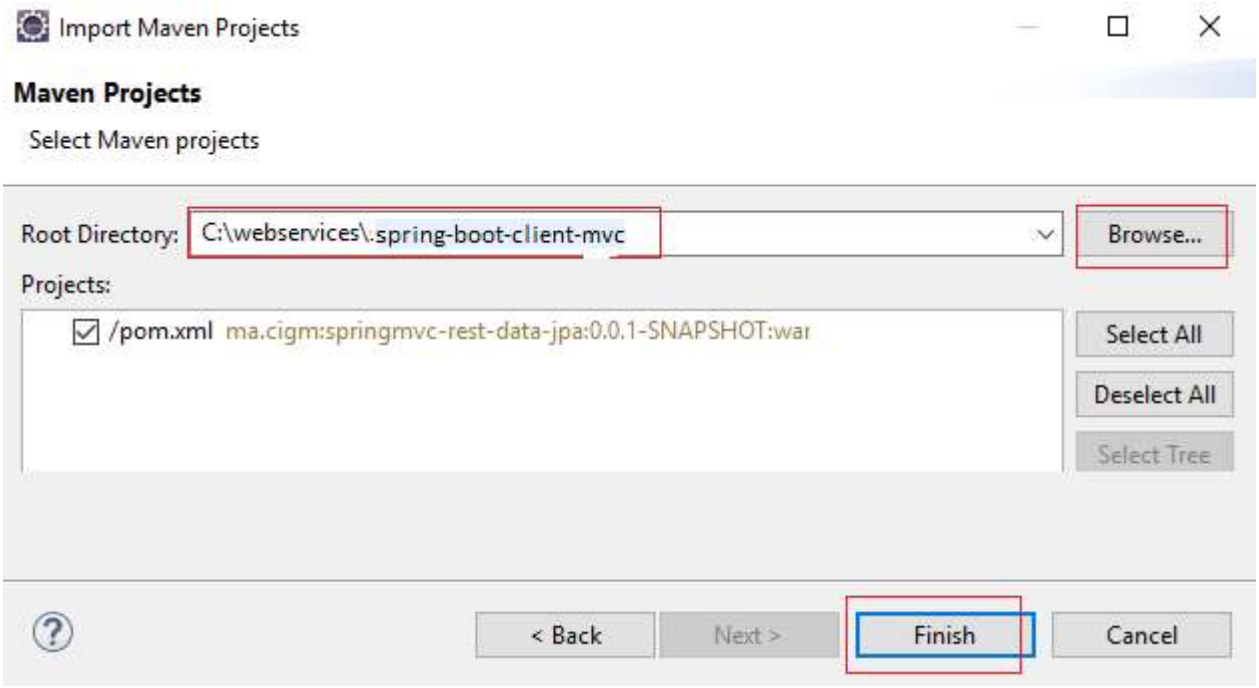
- Décompresser le fichier springboot-client-mvc.zip dans le dossier c:\webservices par exemple, ensuite importer le projet Maven au niveau d'éclipse comme illustré ci-après :



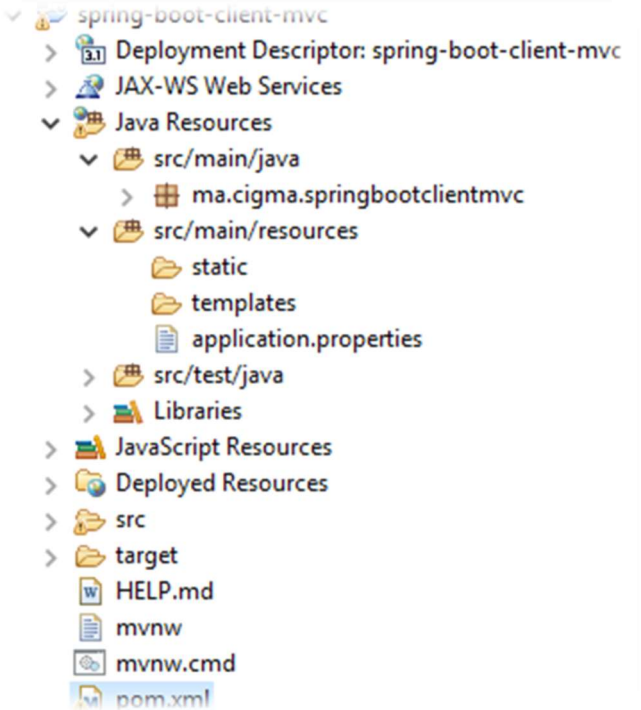
- Cliquer sur le menu « Import… » :



- Choisir « Existing Maven Projects » et cliquer sur Next> :



- Cliquer sur « Browse… » et choisir le répertoire dans lequel existe votre projet Maven (ici : c:\webservices\spring-boot-client-mvc) et cliquer sur Finish. L'arborescence du projet « spring-boot-client-mvc » est comme suit :



2. pom.xml

- Editer le fichier pom.xml et ajouter les dépendances suivantes :

```
<!-- Pour pouvoir utiliser JSP, les dépendances suivantes sont nécessaires --
>
<dependency>
   <groupId>javax.servlet.jsp.jstl</groupId>
   <artifactId>javax.servlet.jsp.jstl-api</artifactId>
   <version>1.2.1</version>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>

<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
  <version>9.0.2</version>
</dependency>

<!-- Pour que les RestController puissent produire le format XML, la
dépendance suivante est nécessaire -->
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

- Le fichier pom.xml, une fois les modifications opérées, 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>2.2.0.BUILD-SNAPSHOT</version>
            <relativePath /> <!-- lookup parent from repository -->
      </parent>
      <groupId>ma.cigma</groupId>
      <artifactId>spring-boot-client-mvc</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>war</packaging>
      <name>spring-boot-client-mvc</name>
      <description>Demo project for Spring Boot</description>

      <properties>
            <java.version>1.8</java.version>
      </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.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
			<exclusions>
				<exclusion>
					<groupId>org.junit.vintage</groupId>
					<artifactId>junit-vintage-engine</artifactId>
				</exclusion>
				<exclusion>
					<groupId>junit</groupId>
					<artifactId>junit</artifactId>
				</exclusion>
			</exclusions>
		</dependency>

		<!-- Pour pouvoir utiliser JSP,les dépendances suivantes sont nécessaires -->
		<dependency>
			<groupId>javax.servlet.jsp.jstl</groupId>
			<artifactId>javax.servlet.jsp.jstl-api</artifactId>
			<version>1.2.1</version>
		</dependency>

		<dependency>
			<groupId>taglibs</groupId>
			<artifactId>standard</artifactId>
			<version>1.1.2</version>
		</dependency>

		<dependency>
			<groupId>org.apache.tomcat.embed</groupId>
			<artifactId>tomcat-embed-jasper</artifactId>
			<version>9.0.2</version>
		</dependency>
		<!-- Pour que les RestController puissent produire le format XML, la
dépendance suivante est nécessaire -->
		<dependency>
			<groupId>com.fasterxml.jackson.dataformat</groupId>
			<artifactId>jackson-dataformat-xml</artifactId>
		</dependency>

	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
        </plugins>
    </build>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
        </pluginRepository>
    </pluginRepositories>

</project>
```

3. Le fichier application.properties

Le contenu du fichier application.properties est :

```
#Pour Spring MVC :
spring.mvc.view.prefix=/vues/
spring.mvc.view.suffix=.jsp
server.port=9999
#Le serveur Rest :
rest.url=http://localhost:8080/rest/emp/
```

4. Le Value Object

- Créer le package ma.cigma.springbootclientmvc.domaine.
- Ensuite, créer la classe ***EmpVo*** suivante :

```
package ma.cigma.springbootclientmvc.domaine;

public class EmpVo {
	private Long id;
	private String name;
	private Double salary;
	private String fonction;

	public EmpVo() {
		super();
	}

	public EmpVo(Long id, String name, Double salary, String fonction)
{
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
		this.fonction = fonction;
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public Double getSalary() {
		return salary;
	}

	public void setSalary(Double salary) {
		this.salary = salary;
	}
```

```
	public String getFonction() {
		return fonction;
	}

	public void setFonction(String fonction) {
		this.fonction = fonction;
	}
}
```

5. Le contrôleur

- Créer le package ma.cigma.springbootclientmvc.controller.
- Ensuite, créer la classe ***EmpController*** suivante :

```
package ma.cigma.springbootclientmvc.controller;

import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
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 ma.cigma.springbootclientmvc.domaine.EmpVo;

@Controller
public class EmpController {

	@Autowired
	private RestTemplate restTemplate;

	@Value("${rest.url}")
	private String url;

	/**
	 * Lorsqu'on tape le lien http://localhost:8080, la page
	 * /WEB-INF/vues/index.jsp. Aucun objet n'est passé dans le Model.
	 */
	@RequestMapping("/")
	public String showWelcomeFile(Model m) {
		return "index";
	}

	/**
```

```
	 * Permet d'afficher la page /WEB-INF/vues/empform.jsp. L'objet qui est
	 * passé dans la requête est "employe" de type la classe EmpVo. Les
	 * attributs de l'objet "employe" seront accessibles au niveau de la page
	 * moyennant les gettets et les setters.
	 */
	@RequestMapping("/empform")
	public String showform(Model m) {
		m.addAttribute("empVo", new EmpVo());
		return "empform";
	}

	/**
	 * 1°) Au niveau du formulaire "empform.jsp", lorsqu'on clique sur le bouton
	 * Submit, l'action "/save" sera exécutée. Les valeurs du formulaires seront
	 * passés dans l'objet EmpVo. Ici, il faut préciser que la méthode HTTP est
	 * bien POST car la méthode par défaut est GET.
	 *
	 * 2°) la méthode save() de l'interface IService sera lancée. 3°) Ensuite la
	 * réponse sera redirigée vers la page /WEB-INF/vues/viewemp.jsp
	 */
	@RequestMapping(value = "/save", method = RequestMethod.POST)
	public String save(@ModelAttribute("empVo") EmpVo emp) {

		// HttpHeaders
		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { 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);

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

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

	/**
	 * lorsqu'on tape le lien http://localhost:8080/viewemp, la page
	 * /WEB-INF/vues/viewemp.jsp sera affichée. La liste des employées est
	 * placée dans le Model.
	 */
	@RequestMapping("/viewemp")
	public String viewemp(Model m) {

		// HttpHeaders
		EmpVo[] list = null;
		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { 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);
```

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

		// Status Code: 200
		if (statusCode == HttpStatus.OK)
			list = response.getBody();
		m.addAttribute("list", Arrays.asList(list));
		return "viewemp";
	}

	/**
	 * lorsqu'on tape le lien http://localhost:8080/editemp/id, la page
	 * /WEB-INF/vues/empeditform.jsp sera affichée. L'objet EmpVo est placé dans
	 * le Model.
	 */
	@RequestMapping(value = "/editemp/{id}")
	public String edit(@PathVariable Long id, Model m) {
		// HttpHeaders
		EmpVo emp = null;
		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { 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 + id, HttpMethod.GET,
entity, EmpVo.class);

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

		// Status Code: 200
		if (statusCode == HttpStatus.OK)
			emp = response.getBody();
		m.addAttribute("empVo", emp);
		return "empeditform";
	}

	/**
	 * lorsqu'on tape le lien http://localhost:8080/editsave, l'objet EmpVo est
	 * passé dans la reqûete, ensuite on exécute la méthode save(). Ensuite, on
	 * redirige la réponse vers la page /WEB-INF/vues/viewemp.jsp. Ici, il faut
	 * préciser la méthode POST.
	 */
	@RequestMapping(value = "/editsave", method = RequestMethod.POST)
	public String editsave(@ModelAttribute("empVo") EmpVo emp) {
		// HttpHeaders
		HttpHeaders headers = new HttpHeaders();
		headers.setAccept(Arrays.asList(new MediaType[] { 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);
```

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

            return "redirect:/viewemp";
      }

      /**
       * lorsqu'on tape le lien http://localhost:8080/deleteemp/id, on récupère la
       * valeur du paramètre id, on exécute save() et après on redirige la réponse
       * vers la page /WEB-INF/vues/viewemp.jsp.
       */
      @RequestMapping(value = "/deleteemp/{id}", method = RequestMethod.GET)
      public String delete(@PathVariable Long id) {
            // HttpHeaders
            HttpHeaders headers = new HttpHeaders();
            headers.setAccept(Arrays.asList(new MediaType[] { 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);

            HttpStatus statusCode = response.getStatusCode();
            System.out.println("Response Satus Code: " + statusCode);
            return "redirect:/viewemp";
      }
}
```

6. **Les pages JSP**
   - Créer le dossier « vues » dans src/main/webapp.
   - Créer en suite les pages suivantes :

src/main/webapp/vues/***index.jsp***

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

src/main/webapp/vues/**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="name" /></td>
            </tr>
            <tr>
                  <td>Salary :</td>
                  <td><form:input path="salary" /></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>
```

src/main/webapp/vues/**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>${empVo.name}</td>
                        <td>${empVo.salary}</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>
```

src/main/webapp/vues/ **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="name"  /></td>
         </tr>
         <tr>
          <td>Salary :</td>
          <td><form:input path="salary" /></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>
```

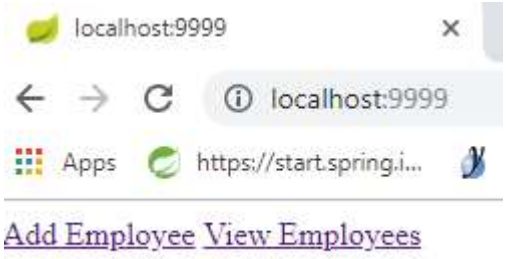
7. La classe de démarrage de Spring Boot

Modifier la classe de démarrage de Spring Boot comme suit :

```
package ma.cigma.springbootclientmvc;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class MainApplication {

	public static void main(String[] args) {
		SpringApplication.run(MainApplication.class, args);
	}

	@Bean
	public RestTemplate getRestTemplate() {
		return new RestTemplate();
	}
}
```

8. Les tests

- Lancer la méthode main ci-dessus et exécuter par la suite le lien http://localhost:9999. La page index.jsp devrait être exécutée :



- Effectuer ensuite les tests CRUD (idem que le TP n°4).