TP N°6 : Développement d’une application web avec Spring Boot, Spring Security, Thymeleaf, Spring Data JPA et MySQL.

SOMMAIRE

I- Objectifs :

- ✓ Sécuriser une application WEB avec Spring Security :
  - /login et /welcome : accessible par tous les utilisateurs qui ont un compte.
  - /admin : accessible uniquement par les utilisateurs ayant le rôle ADMIN.
  - /client : accessible uniquement pour les utilisateurs ayant le rôle CLIENT.
- ✓ Les utilisateurs sont stockés dans une base de données MySQL.
- ✓ Les vues sont développées avec le moteur de Template : ThymeLeaf.
- ✓ Utilisation de Lombok pour ne pas écrire les gettes, les setters, les constructeurs, etc.

II- Outils utilisés :

Dans cet atelier, nous allons utiliser les outils suivants :

- ✓ Eclipse Neon 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, …).
- ✓ La base de données MySQL 8.

III- Développement de l'application

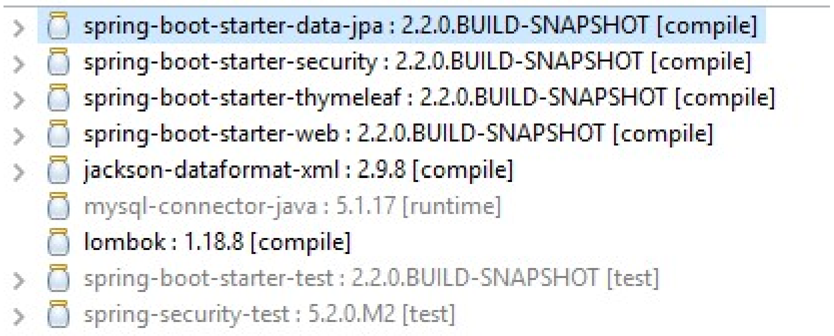
1. pom.xml

Créer un projet Maven (soit en utilisant Spring Initializr ou bien Eclipse). Le contenu du fichier pom.xml est :

```xml
<?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>springsecurity</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>springsecurity</name>
      <description>Demo project for Spring Boot</description>

      <properties>
            <java.version>1.8</java.version>
      </properties>

      <dependencies>
            <dependency>
```

```xml
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-security</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-thymeleaf</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</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>

		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<scope>runtime</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>
			<exclusions>
				<exclusion>
					<groupId>org.junit.vintage</groupId>
					<artifactId>junit-vintage-engine</artifactId>
				</exclusion>
				<exclusion>
					<groupId>junit</groupId>
					<artifactId>junit</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-test</artifactId>
			<scope>test</scope>
		</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>
```

*Remarquer les dépendances suivantes :



```
spring-boot-starter-data-jpa : 2.2.0.BUILD-SNAPSHOT [compile]
spring-boot-starter-security : 2.2.0.BUILD-SNAPSHOT [compile]
spring-boot-starter-thymeleaf : 2.2.0.BUILD-SNAPSHOT [compile]
spring-boot-starter-web : 2.2.0.BUILD-SNAPSHOT [compile]
jackson-dataformat-xml : 2.9.8 [compile]
mysql-connector-java : 5.1.17 [runtime]
lombok : 1.18.8 [compile]
spring-boot-starter-test : 2.2.0.BUILD-SNAPSHOT [test]
spring-security-test : 5.2.0.M2 [test]
```

2. application.properties

```
spring.datasource.url = jdbc:mysql://localhost:3306/tp6?createDatabaseIfNotExist=true&autoReconnect=true&useSSL=true
&useUnicode=yes&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username = root
spring.datasource.password = root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
```

3. Les modèles

```
package ma.cigma.springsecurity.service.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@Table(name = "role")
@NoArgsConstructor
public class Role {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	@Column(name = "role_id")
	private int id;
	@Column(name = "role")
	private String role;

	public Role(String role) {
		this.role = role;
	}
}
```

```java
package ma.cigma.springsecurity.service.model;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.validation.constraints.NotEmpty;

import org.hibernate.validator.constraints.Length;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@Table(name = "user")
@NoArgsConstructor
public class User {
	@Id
	@GeneratedValue
	private Long id;

	@Length(min = 5, message = "*Your username must have at least 5 characters")
	@NotEmpty(message = "*Please provide an user name")
	private String username;

	@Length(min = 5, message = "*Your password must have at least 5 characters")
	@NotEmpty(message = "*Please provide your password")
	private String password;

	@ManyToMany(cascade = CascadeType.ALL)
	@JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"), 
inverseJoinColumns = @JoinColumn(name = "role_id"))
	private List<Role> roles = new ArrayList<Role>();
}
```

4. Les Values objects (ou DTO : Data Transfer Object)

```
package ma.cigma.springsecurity.domaine;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
public class RoleVo {
	private int id;
	private String role;

	public RoleVo(String role) {
		this.role = role;
	}
}
```

```
package ma.cigma.springsecurity.domaine;

import java.util.ArrayList;
import java.util.List;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
public class UserVo {
	private Long id;
	private String username;
	private String password;
	private List<RoleVo> roles = new ArrayList<RoleVo>();

	public UserVo(String username, String password,List<RoleVo> roles) {
		this.username = username;
		this.password = password;
		this.roles=roles;
	}
}
```

```
package ma.cigma.springsecurity.domaine;

import java.util.ArrayList;
import java.util.List;

import ma.cigma.springsecurity.service.model.Role;

public class RoleConverter {

	public static RoleVo toVo(Role bo) {
```

```java
		if (bo == null)
			return null;
		RoleVo vo = new RoleVo();
		vo.setId(bo.getId());
		vo.setRole(bo.getRole());
		return vo;
	}

	public static Role toBo(RoleVo vo) {
		if (vo == null)
			return null;
		Role bo = new Role();
		bo.setId(vo.getId());
		bo.setRole(vo.getRole());
		return bo;
	}

	public static List<RoleVo> toVoList(List<Role> boList) {
		if (boList == null || boList.isEmpty())
			return null;
		List<RoleVo> voList = new ArrayList<>();
		for (Role role : boList) {
			voList.add(toVo(role));
		}
		return voList;
	}

	public static List<Role> toBoList(List<RoleVo> voList) {
		if (voList == null || voList.isEmpty())
			return null;
		List<Role> boList = new ArrayList<>();
		for (RoleVo roleVo : voList) {
			boList.add(toBo(roleVo));
		}
		return boList;
	}
}
```

```java
package ma.cigma.springsecurity.domaine;

import java.util.ArrayList;
import java.util.List;

import ma.cigma.springsecurity.service.model.User;

public class UserConverter {
	public static UserVo toVo(User bo) {
		if (bo == null)
			return null;
		UserVo vo = new UserVo();
		vo.setId(bo.getId());
		vo.setUsername(bo.getUsername());
		vo.setPassword(vo.getPassword());
		vo.setRoles(RoleConverter.toVoList(bo.getRoles()));
		return vo;
	}

	public static User toBo(UserVo vo) {
		if (vo == null)
			return null;
		User bo = new User();
		if (vo.getId() != null)
			bo.setId(vo.getId());
		bo.setUsername(vo.getUsername());
		bo.setPassword(vo.getPassword());
		bo.setRoles(RoleConverter.toBoList(vo.getRoles()));
		return bo;
	}

	public static List<UserVo> toVoList(List<User> boList) {
		if (boList == null || boList.isEmpty())
			return null;
		List<UserVo> voList = new ArrayList<>();
		for (User user : boList) {
			voList.add(toVo(user));
		}
		return voList;
	}

	public static List<User> toBoList(List<UserVo> voList) {
		if (voList == null || voList.isEmpty())
			return null;
		List<User> boList = new ArrayList<>();
		for (UserVo userVo : voList) {
			boList.add(toBo(userVo));
		}
		return boList;
	}
}
```

5. La couche DAO

```java
package ma.cigma.springsecurity.dao;

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

import ma.cigma.springsecurity.service.model.User;

public interface UserRepository extends JpaRepository<User, Long> {
	User findByUsername(String userName);
}
```

```java
package ma.cigma.springsecurity.dao;

import java.util.List;

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

import ma.cigma.springsecurity.service.model.Role;

```

```java
public interface RoleRepository extends JpaRepository<Role, Integer> {
	List<Role> findByRole(String role);
	List<Role> findAll();
}
```

6. La couche Service

```java
package ma.cigma.springsecurity.service;

import java.util.List;

import org.springframework.security.core.userdetails.UserDetailsService;

import ma.cigma.springsecurity.domaine.RoleVo;
import ma.cigma.springsecurity.domaine.UserVo;

public interface IUserService extends UserDetailsService{
	void save(UserVo user);
	void save(RoleVo role);
	List<UserVo> getAllUsers();
	List<RoleVo> getAllRoles();
	RoleVo getRoleByName(String role);
	void cleanDataBase();
}
```

```java
package ma.cigma.springsecurity.service;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import ma.cigma.springsecurity.dao.RoleRepository;
import ma.cigma.springsecurity.dao.UserRepository;
import ma.cigma.springsecurity.domaine.RoleConverter;
import ma.cigma.springsecurity.domaine.RoleVo;
import ma.cigma.springsecurity.domaine.UserConverter;
import ma.cigma.springsecurity.domaine.UserVo;
import ma.cigma.springsecurity.service.model.Role;
import ma.cigma.springsecurity.service.model.User;

@Service("userService")
@Transactional
public class UserServiceImpl implements IUserService {
	@Autowired
	private UserRepository userRepository;
	@Autowired
	private RoleRepository roleRepository;
	@Autowired
	private BCryptPasswordEncoder bCryptPasswordEncoder;

	public UserServiceImpl(UserRepository userRepository, RoleRepository
roleRepository,
			BCryptPasswordEncoder bCryptPasswordEncoder) {
		this.userRepository = userRepository;
		this.roleRepository = roleRepository;
		this.bCryptPasswordEncoder = bCryptPasswordEncoder;
	}

	@Override
	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
		User user = userRepository.findByUsername(username);
		boolean enabled = true;
		boolean accountNonExpired = true;
		boolean credentialsNonExpired = true;
		boolean accountNonLocked = true;
		return new
org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(),
enabled,
				accountNonExpired, credentialsNonExpired, accountNonLocked,
getAuthorities(user.getRoles()));
	}

	private Collection<? extends GrantedAuthority> getAuthorities(List<Role> roles) {
		List<GrantedAuthority> springSecurityAuthorities = new ArrayList<>();
		for (Role r : roles) {
			springSecurityAuthorities.add(new
SimpleGrantedAuthority(r.getRole()));
		}
		return springSecurityAuthorities;
	}

	@Override
	public void save(UserVo userVo) {
		User user=UserConverter.toBo(userVo);
		user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
		List<Role> rolesPersist = new ArrayList<>();
		for (Role role : user.getRoles()) {
			Role userRole = roleRepository.findByRole(role.getRole()).get(0);
			rolesPersist.add(userRole);
		}
		user.setRoles(rolesPersist);
		userRepository.save(user);
	}

	@Override
	public void save(RoleVo roleVo) {
		roleRepository.save(RoleConverter.toBo(roleVo));
	}

	@Override
	public List<UserVo> getAllUsers() {
		return UserConverter.toVoList(userRepository.findAll());
	}

	@Override
	public List<RoleVo> getAllRoles() {
		return RoleConverter.toVoList(roleRepository.findAll());
	}

	@Override
	public RoleVo getRoleByName(String role) {
		return RoleConverter.toVo(roleRepository.findByRole(role).get(0));
	}

	@Override
	public void cleanDataBase() {
		userRepository.deleteAll();
		roleRepository.deleteAll();
	}
}
```

7. La couche présentation

```
package ma.cigma.springsecurity.presentation;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class LoginController {

	@RequestMapping(value = { "/", "/login" }, method = RequestMethod.GET)
	public ModelAndView login() {
		ModelAndView modelAndView = new ModelAndView();
		modelAndView.setViewName("login");
		return modelAndView;
	}

	@RequestMapping(value = "/welcome", method = RequestMethod.GET)
	public ModelAndView welcome() {
		ModelAndView modelAndView = new ModelAndView();
		Authentication auth = SecurityContextHolder.getContext().getAuthentication();
		modelAndView.addObject("userLogIn", auth.getName());
		modelAndView.setViewName("welcome");
		return modelAndView;
	}

	@RequestMapping(value = "/admin", method = RequestMethod.GET)
	public ModelAndView methodForAdmin() {
		ModelAndView modelAndView = new ModelAndView();
		Authentication auth = SecurityContextHolder.getContext().getAuthentication();
		modelAndView.addObject("userName", "Welcome " + auth.getName());
		modelAndView.addObject("adminMessage", "Content Available Only for Admins
with ADMIN Role");
```

```
		modelAndView.setViewName("/admin/admin");
		return modelAndView;
	}

	@RequestMapping(value = "/client", method = RequestMethod.GET)
	public ModelAndView methodForClient() {
		ModelAndView modelAndView = new ModelAndView();
		Authentication auth = SecurityContextHolder.getContext().getAuthentication();
		modelAndView.addObject("userName", "Welcome " + auth.getName());
		modelAndView.addObject("clientMessage", "Content Available Only for Clients 
with CLIENT Role");
		modelAndView.setViewName("client/client");
		return modelAndView;
	}

	@RequestMapping(value = "/access-denied", method = RequestMethod.GET)
	public ModelAndView accessdenied() {
		ModelAndView modelAndView = new ModelAndView();
		modelAndView.setViewName("access-denied");
		return modelAndView;
	}
}
```

8. Les vues

Il faut créer les pages html au niveau du dossier /resoures/templates

La page login.html (/resoures/templates/login.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
	xmlns:th="http://www.thymeleaf.org">

<head>
	<title>Formation Spring Boot : Services Web</title>
	<link rel="stylesheet" type="text/css" th:href="@{/css/login.css}" />
	<link rel="stylesheet" 
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
	<script 
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
	<script 
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>

<body>
	<div class="container">
		<img th:src="@{/images/login.jpg}" class="img-responsive center-block" 
width="300" height="300" alt="Logo" />
		<form th:action="@{/login}" method="POST" class="form-signin">
			<h3 class="form-signin-heading" th:text="Welcome"></h3>
			<br/>

			<input type="text" id="username" name="username" 
th:placeholder="Username" class="form-control" /> <br/>
			<input type="password" id="password" name="password" 
th:placeholder="Password" class="form-control" /> <br />

			<div align="center" th:if="${param.error}">
```

```
                            <p style="font-size: 20; color: #FF1C19;">Username ou Mot de
passe incorrect</p>
                     </div>
                     <button class="btn btn-lg btn-primary btn-block" name="Submit"
value="Login" type="Submit" th:text="Login"></button>
              </form>
       </div>
</body>
</html>
```

La page welcome.html (/resources/templates/welcome.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:th="http://www.thymeleaf.org">

<head>
<title>Formation Spring Boot : Services Web</title>
<link rel="stylesheet" type="text/css" th:href="@{/css/home.css}" />
<link rel="stylesheet"
       href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script
       src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script
       src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>

<body>
       <div class="container">
              <span th:utext="${userName}"></span>
              <br>
              <form th:action="@{/client}" method="get">
                     <button class="btn btn-md btn-danger btn-block"    name="Services
Métier" type="Submit">Pour les clients</button>
              </form>
              <br>
              <form th:action="@{/admin}" method="get">
                     <button class="btn btn-md btn-danger btn-block"    name="Créer un
nouvel utilisateur" type="Submit">Pour les admin</button>
              </form>
       </div>
</body>
</html>
```

La page access-denied.html (/resources/templates/access-denied.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:th="http://www.thymeleaf.org">

<head>
       <title>Spring Security Tutorial</title>
       <link rel="stylesheet" type="text/css" th:href="@{/css/login.css}" />
       <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
       <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
       <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
```

```
<body>
      <form th:action="@{/login}" method="get">
            <button class="btn btn-md btn-warning btn-block" type="Submit">Login</button>
      </form>
      <h2>Vous n'avez pas le droit d'accéder à cette page. Merci de vous
s'authentifier</h2>
</body>
</html>
```

La page admin.html (/resources/templates/admin/admin.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Formation Spring Boot : Services Web</title>
    <link rel="stylesheet" type="text/css" th:href="@{/css/home.css}"/>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>

<body>
<div class="container">

    <form th:action="@{/logout}" method="get">
        <button class="btn btn-md btn-danger btn-block" name="registration"
                type="Submit">Logout
        </button>
    </form>

    <div class="panel-group" style="margin-top:40px">
        <div class="panel panel-primary">
            <div class="panel-heading">
                <span th:utext="${userName}"></span>
            </div>
            <div class="panel-body">
                <img th:src="@{/images/data.jpg}" class="img-responsive center-block"
width="400" height="400"
                     alt="Beer"/>
            </div>
            <p class="admin-message-text text-center" th:utext="${adminMessage}"></p>
        </div>
    </div>

</div>
</body>
</html>
```

La page client.html (/resources/templates/client/client.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">

<head>
 <title>Formation Spring Boot : Services Web</title>
    <link rel="stylesheet" type="text/css" th:href="@{/css/home.css}"/>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></sc
ript>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
</head>

<body>
<div class="container">

    <form th:action="@{/logout}" method="get">
        <button class="btn btn-md btn-danger btn-block" name="registration"
                type="Submit">Logout
        </button>
    </form>

    <div class="panel-group" style="margin-top:40px">
        <div class="panel panel-primary">
            <div class="panel-heading">
                <span th:utext="${userName}"></span>
            </div>
            <div class="panel-body">
                <img th:src="@{/images/client.jpg}" class="img-responsive
center-block" width="400" height="400" alt="Data"/>
            </div>
            <p class="admin-message-text text-center"
th:utext="${clientMessage}"></p>
        </div>
    </div>
</div>
</body>
</html>
```

9. Les feuilles de style (*.css)

Les feuilles de style doivent être dans le dossier /resources/static.

home.css (/resources/static/css/home.css)

```css
.admin-message-text {
    font-style: normal;
    font-size: 22px;
    color: #004080;
}
```

login.css (/resources/static/css/login.css)

```css
.wrapper {
    margin-top: 80px;
    margin-bottom: 20px;
}

.form-signin {
    max-width: 420px;
    padding: 30px 38px 66px;
    margin: 0 auto;
    background-color: #eee;
    border: 3px dotted rgba(0,0,0,0.1);
}

.form-signin-heading {
    text-align:center;
    margin-bottom: 30px;
}

.form-control {
    position: relative;
    font-size: 16px;
    height: auto;
    padding: 10px;
}

input[type="text"] {
    margin-bottom: 0px;
    border-bottom-left-radius: 0;
    border-bottom-right-radius: 0;
}

input[type="password"] {
    margin-bottom: 20px;
    border-top-left-radius: 0;
    border-top-right-radius: 0;
}
```

10. Les images

Copier les images client.jsp, data.jsp et admin.jsp dans le dossier /resources/static/images.

11.La classe de démarrage de Spring Boot

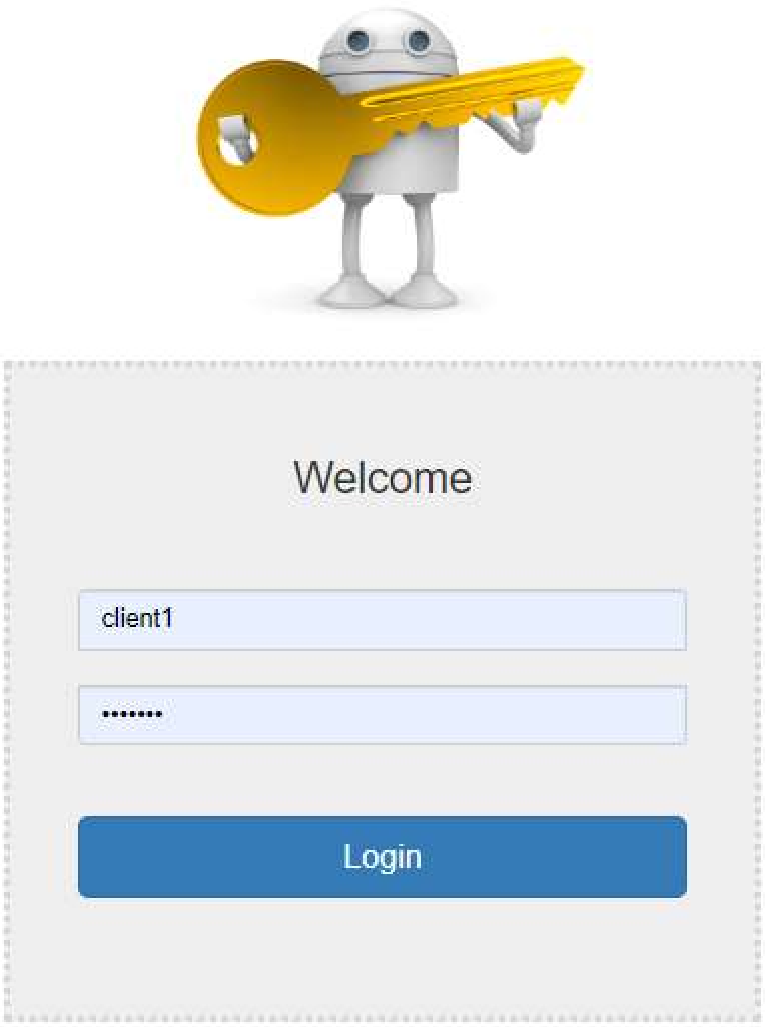
```java
package ma.cigma.springsecurity;

import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import ma.cigma.springsecurity.domaine.RoleVo;
import ma.cigma.springsecurity.domaine.UserVo;
import ma.cigma.springsecurity.service.IUserService;

@SpringBootApplication
public class LoginApplication implements CommandLineRunner {

	@Autowired
	private IUserService userService;
	public static void main(String[] args) {
		SpringApplication.run(LoginApplication.class, args);
	}

	@Bean
	public BCryptPasswordEncoder passwordEncoder() {
		BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
		return bCryptPasswordEncoder;
	}
	@Override
	public void run(String... args) throws Exception {
		userService.cleanDataBase();
		userService.save(new RoleVo("ADMIN"));
		userService.save(new RoleVo("CLIENT"));

		RoleVo roleAdmin=userService.getRoleByName("ADMIN");
		RoleVo roleClient=userService.getRoleByName("CLIENT");
		UserVo admin1=new UserVo("admin1", "admin1",Arrays.asList(roleAdmin));
		UserVo client1=new UserVo("client1","client1",Arrays.asList(roleClient));
		userService.save(admin1);
		userService.save(client1);
	}

}
```

12.Le classe de configuration (Java Config)

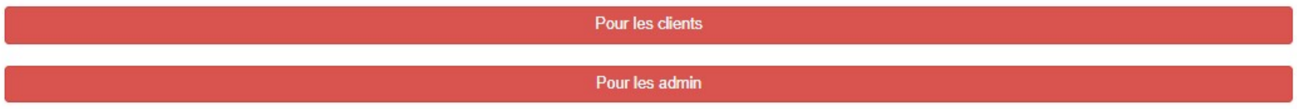
```java
package ma.cigma.springsecurity.configuration;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import ma.cigma.springsecurity.service.IUserService;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

	@Autowired
	private BCryptPasswordEncoder bCryptPasswordEncoder;

	@Autowired
	private IUserService userService;

	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws Exception {
		auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder);
	}
	@Override
	protected void configure(HttpSecurity http) throws Exception {

		http.authorizeRequests().
		antMatchers("/").permitAll().
		antMatchers("/login").permitAll().
		antMatchers("/welcome").hasAnyAuthority("ADMIN","CLIENT").
		antMatchers("/admin/**").hasAuthority("ADMIN").
		antMatchers("/client/**").hasAuthority("CLIENT").
		anyRequest().authenticated().
		and().csrf().disable().
		formLogin().loginPage("/login").
		failureUrl("/login?error=true").
		defaultSuccessUrl("/welcome").
		usernameParameter("username").
		passwordParameter("password").
		and().logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout")).logoutSuccessUrl("/").
		and().exceptionHandling().accessDeniedPage("/access-denied");
	}
	@Override
	public void configure(WebSecurity web) throws Exception {
		web.ignoring().antMatchers("/resources/**", "/static/**", "/css/**", "/js/**", "/images/**");
	}
}
```

13.Les tests

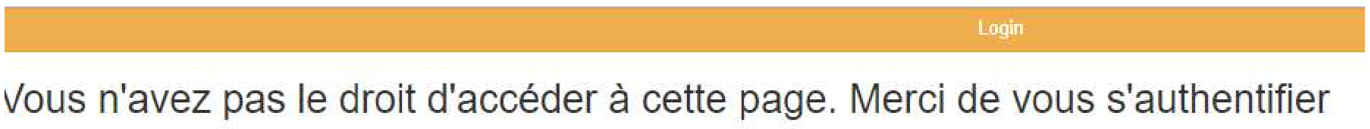
*Lancer la méthode méthode main de la classe LoginApplication et ensuite accéder au site http://localhost:8080.
Le résultat est :



- Entrer admin1/admin1 et cliquer sur Login :

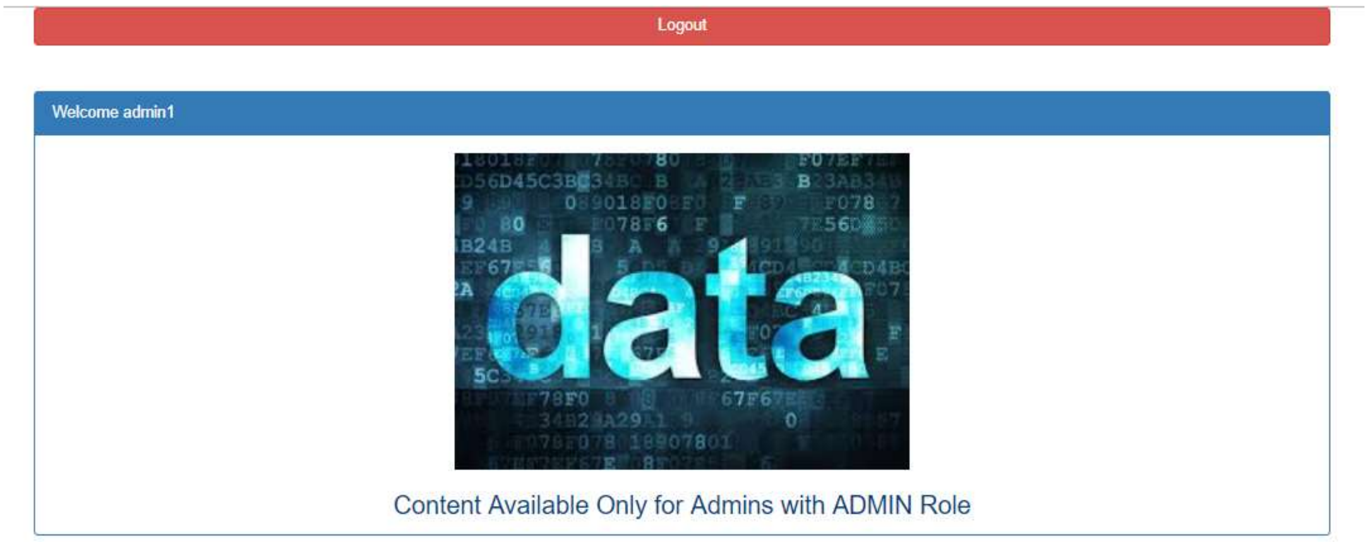


- Cliquer sur « Pour les clients » :



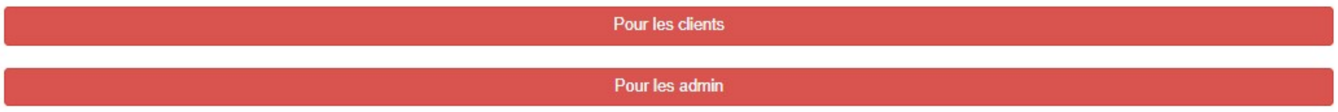
Vous n'avez pas le droit d'accéder à cette page. Merci de vous s'authentifier

- Cliquer sur « Pour les admins » :

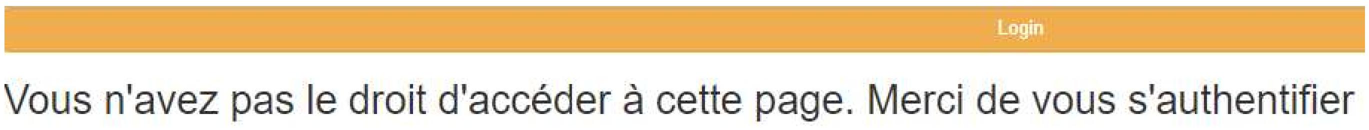


Content Available Only for Admins with ADMIN Role

- De même, entrer client1/client1 :



- Essayer de cliquer sur « Pour les admin » :



Vous n'avez pas le droit d'accéder à cette page. Merci de vous s'authentifier