Journée « Retours d'expériences »

Présentation du framework Spring (avec Spring Boot)

David Darras

Le contexte

L’environnement Spring

Mise en pratique

Université de Lille

2012-2015

- Développements web basés sur le framework esup-commons (basé sur Spring)
⇒ Adaptations nécessaires
⇒ Abandon progressif de la communauté sur ce framework
⇒ Impossibilité de profiter des nouvelles versions de Spring

esup portail
Communauté d'établissements français d'enseignement supérieur pour l'innovation numérique

Depuis 2016

- Passage vers Spring Boot

L’environnement Spring

Université de Lille

Créé en Juin 2003
Opensource (Licence Apache)
Développé par Pivotal software (filiale de VMWare)
http://projects.spring.io/spring-framework/

⇒ Framework applicatif
⇒ Gestion de modules
⇒ Conteneur IOC (Inversion Of Control)
⇒ Gestion de l’AOP (Aspect Oriented Programming)

Université de Lille

IOC « Inversion Of Control »

Framework prend en charge l'exécution principale de l'application
⇒ Recherche et Injection des dépendances
⇒ Découplage des liens de dépendances entre objets

AOP « Aspect Oriented Programming »

⇒ Modularisation de composants techniques réutilisables
⇒ Définition des points d'activation de ces composants

Exemples d'utilisation :
- Génération de logs avant chaque action
- Gestion des transactions
…

Projets Spring

Projets Spring

SPRING DATA

Fournit une approche consistente des accès aux données

SPRING SECURITY

Protection applicative par authentification/autorisation extensible

SPRING LDAP

Développement d'applications utilisant LDAP

Projets Spring

SPRING BATCH

SPRING SOCIAL

SPRING MOBILE

SPRING FOR ANDROID

SPRING WEB FLOW

SPRING SHELL

Projets Spring

Aide à la création, packaging et déploiement d’applications Spring
Auto configuration
⇒ Tomcat, Jetty
⇒ BDDs
⇒ ...
Création d’applications « standalone »

Université de Lille

Autres fonctionnalités

Tests (unitaires, intégration, ...)
Profils de déploiement
Externalisation de configuration
Internationalisation
Cache
Spring Devtools
Switch d'utilisateurs
Connecteur CAS

Cas concret

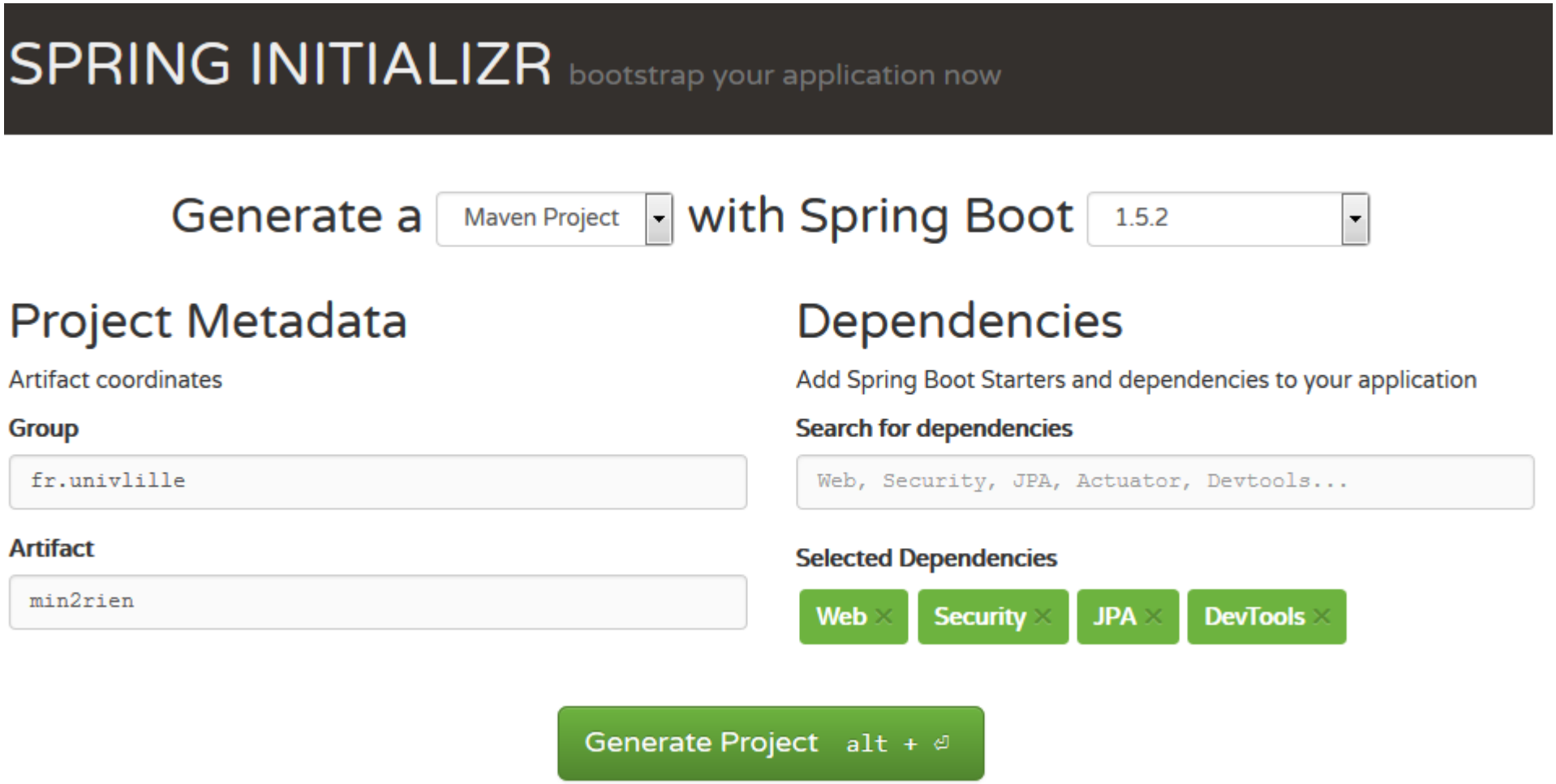
Application Spring Boot avec Spring Data (JPA)/Spring Security
⇒ Accès aux données
⇒ Mise en place de l'authentification
⇒ Gestion de l'inscription

Installation

- Spring samples
- Archétypes maven
- Spring initializr

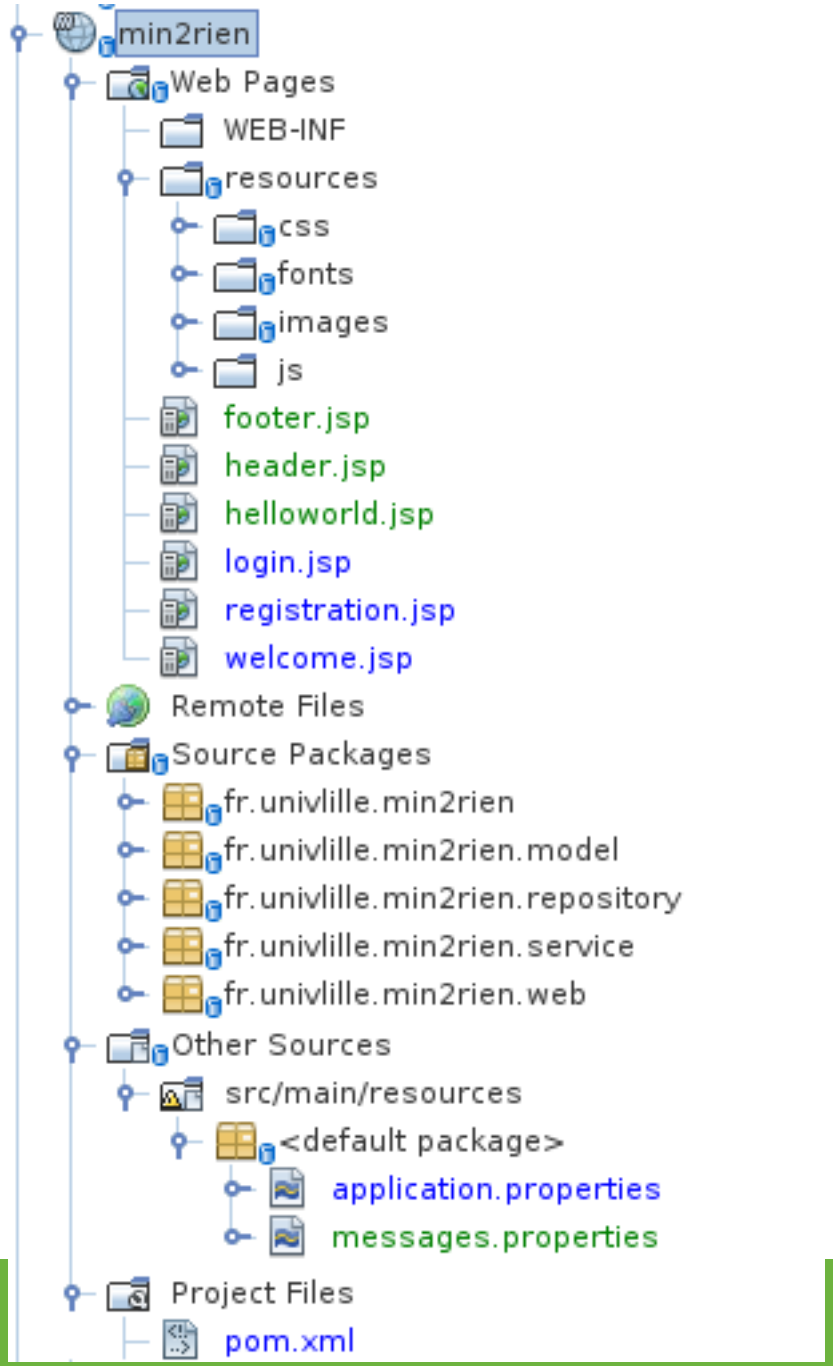
Université de Lille

https://start.spring.io/



Université de Lille

Arborescence



Exemple configuration

```
spring.mvc.view.prefix=/
spring.mvc.view.suffix=.jsp

spring.jpa.hibernate.ddl-auto=create
```

Dépendances et déploiement (Maven)

Outil de gestion de dépendances

Déploiement d'applications
⇒ mvn clean install
⇒ mvn spring-boot:run

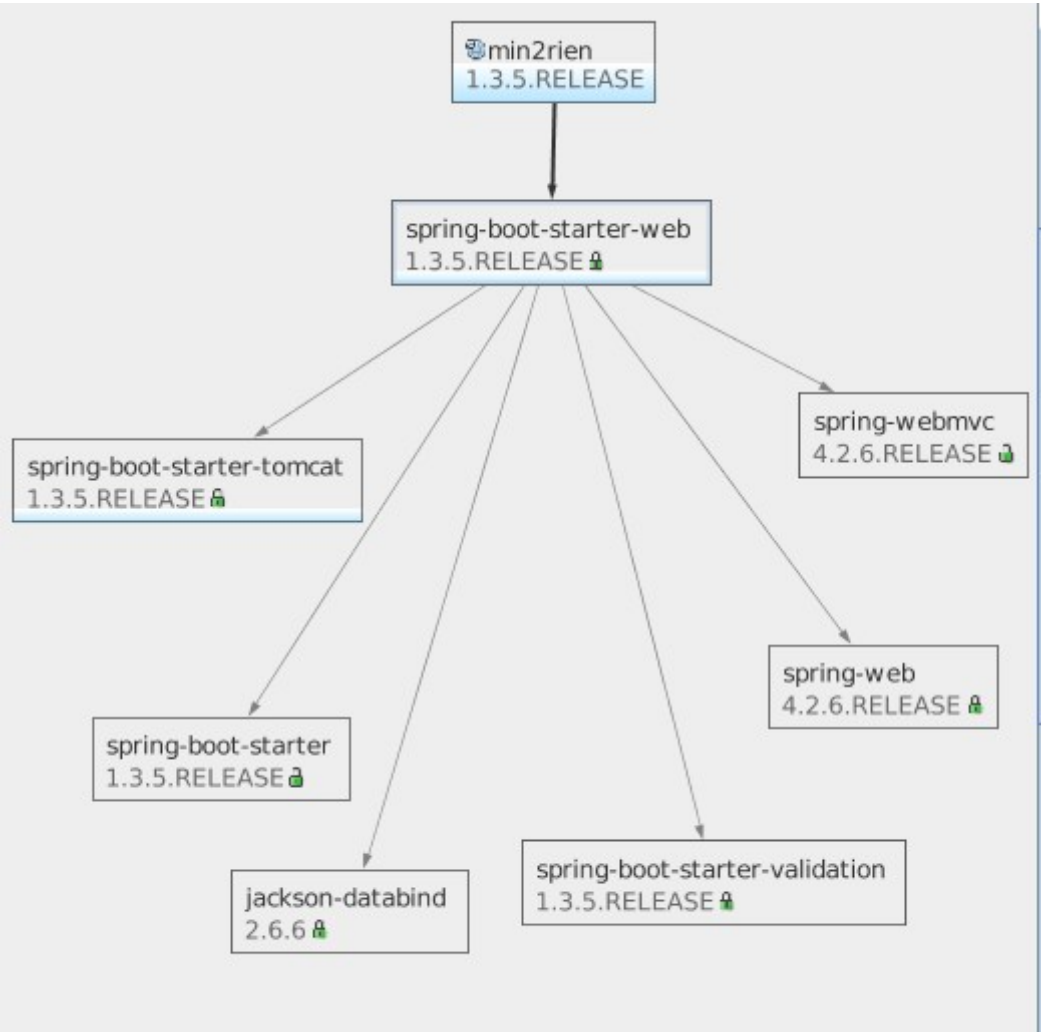
Fichier de configuration pom.xml

```
<?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>
    <artifactId>min2rien</artifactId>
    <name>min2rien</name>
    <description>auth</description>
    <packaging>war</packaging>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.3.5.RELEASE</version>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
    </properties>

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

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

Université de Lille

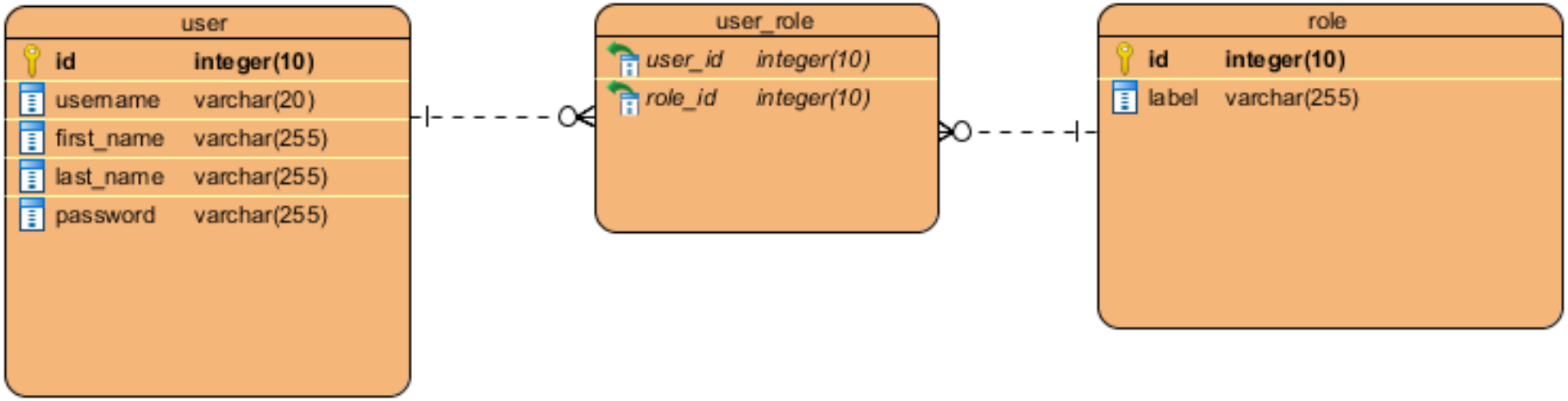
Maven

Arbre de dépendances



Université de Lille

Accès aux données (Spring data / JPA)



UserRepository.java

```java
public interface UserRepository extends JpaRepository<User, Long> {

    // par nom : de méthode. Cf doc ==> http://docs.spring.io/spring-data/jpa/docs
    User findByUsername(String username);

    // par query jpa
    @Query(value = "select u from User u where u.firstname = ?1")
    List<User> findUsersJPA(String firstname);

    // par requête native SQL
    @Query(value = "select * from user where first_name = ?1", nativeQuery = true)
    List<User> findUsersSQL(String firstname);

}
```

CRUD par défaut :
- User save(User user);
- User findOne(Long idUser);
- Iterable<User> findAll();
- Long count();
- void delete(User user);
- boolean exists(Long idUser);

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs,findByFirstnameEquals	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1

User.java

```java
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NotEmpty
    @Size(min = 6, max = 20)
    @Column(name = "username")
    private String username;

    @NotEmpty
    @Size(min = 8)
    @Column(name = "pwd")
    private String password;

    @NotEmpty
    @Column(name = "first_name")
    private String firstname;

    @NotEmpty
    @Column(name = "last_name")
    private String lastname;

    @ManyToMany
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private List<Role> roles;
```

Sécurité

Configuration de la sécurité avec Spring security
⇒ gestion des acl
⇒ authentification interne ou déléguée
⇒ autorisation déléguée à l'application

Prise en main

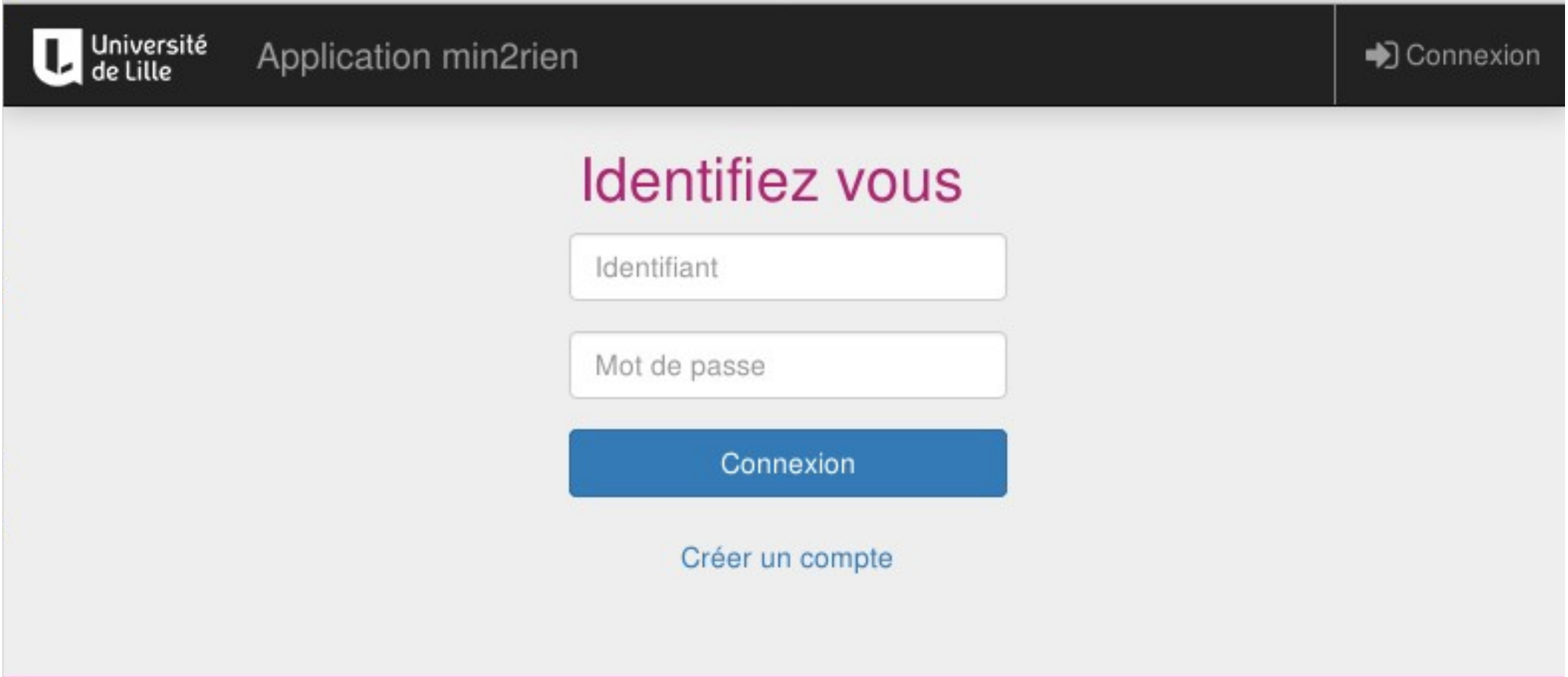
Université de Lille

```java
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                    .antMatchers("/resources/**", "/registration", "/hello", "/").permitAll()
                    .anyRequest().authenticated()
                    .and()
                .formLogin()
                    .loginPage("/login")
                    .permitAll()
                    .and()
                .logout()
                    .permitAll();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordEncoder());
    }
}
```

```java
@Service
public class UserDetailsServiceImpl implements UserDetailsService{
    @Autowired
    private UserRepository userRepository;

    @Override
    @Transactional(readOnly = true)
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username);

        Set<GrantedAuthority> grantedAuthorities = new HashSet<>();
        for (Role role : user.getRoles()){
            grantedAuthorities.add(new SimpleGrantedAuthority(role.getName()));
        }

        return new org.springframework.security.core.userdetails.User(user.getUsername(),
                user.getPassword(), grantedAuthorities);
    }
}
```

Université de Lille

Connexion

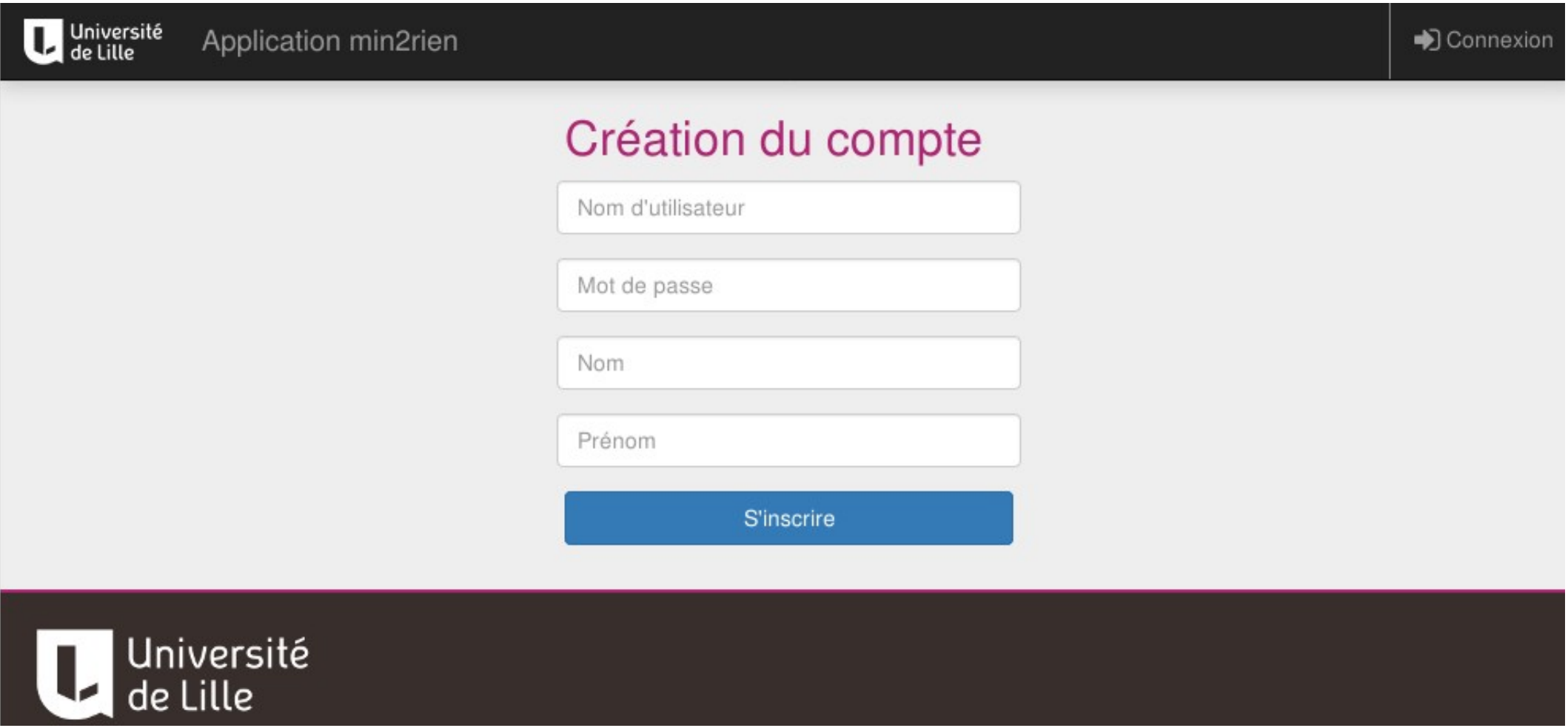


Université de Lille

Inscription – Affichage du formulaire

Université de Lille

Inscription



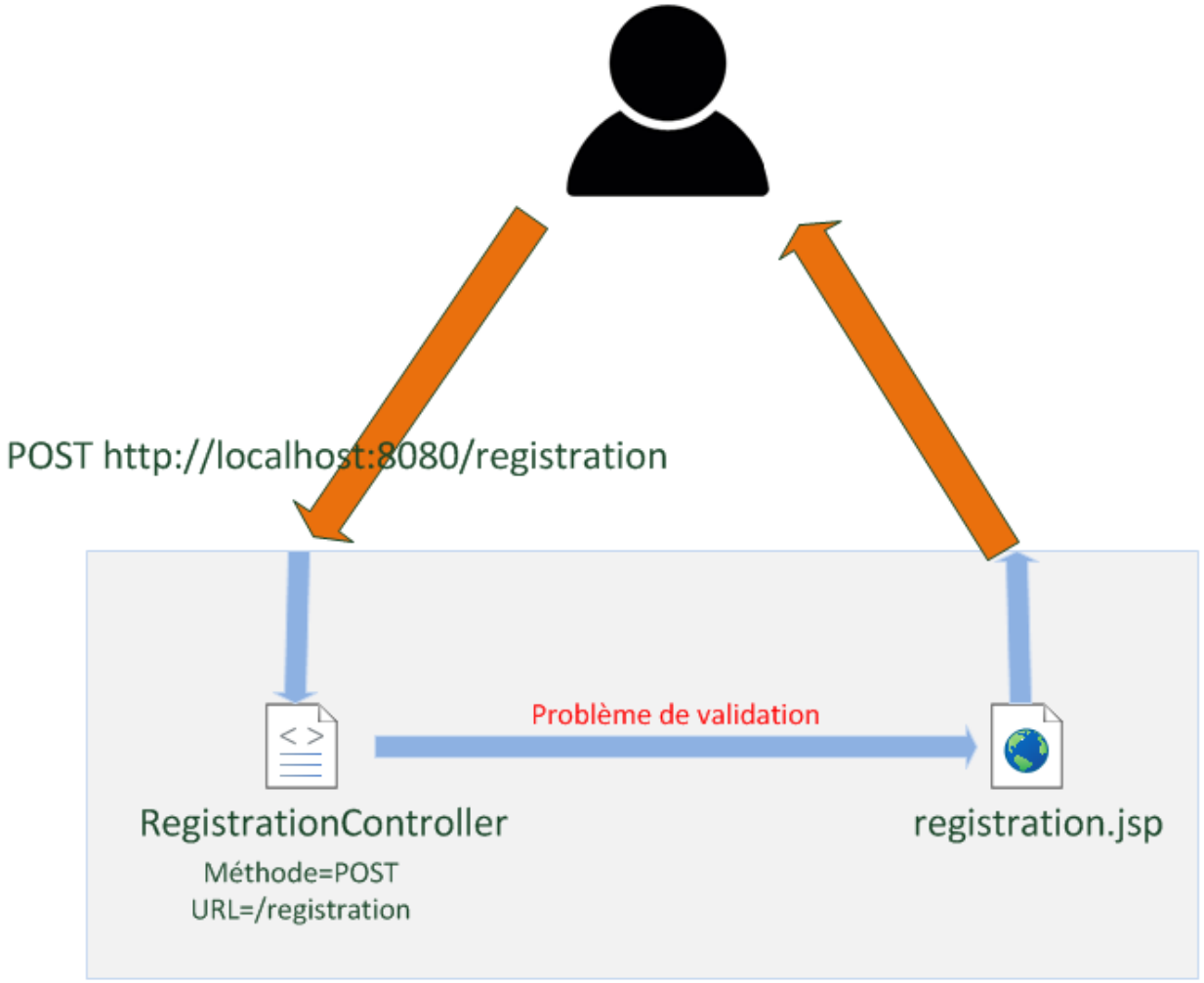
RegistrationController.java

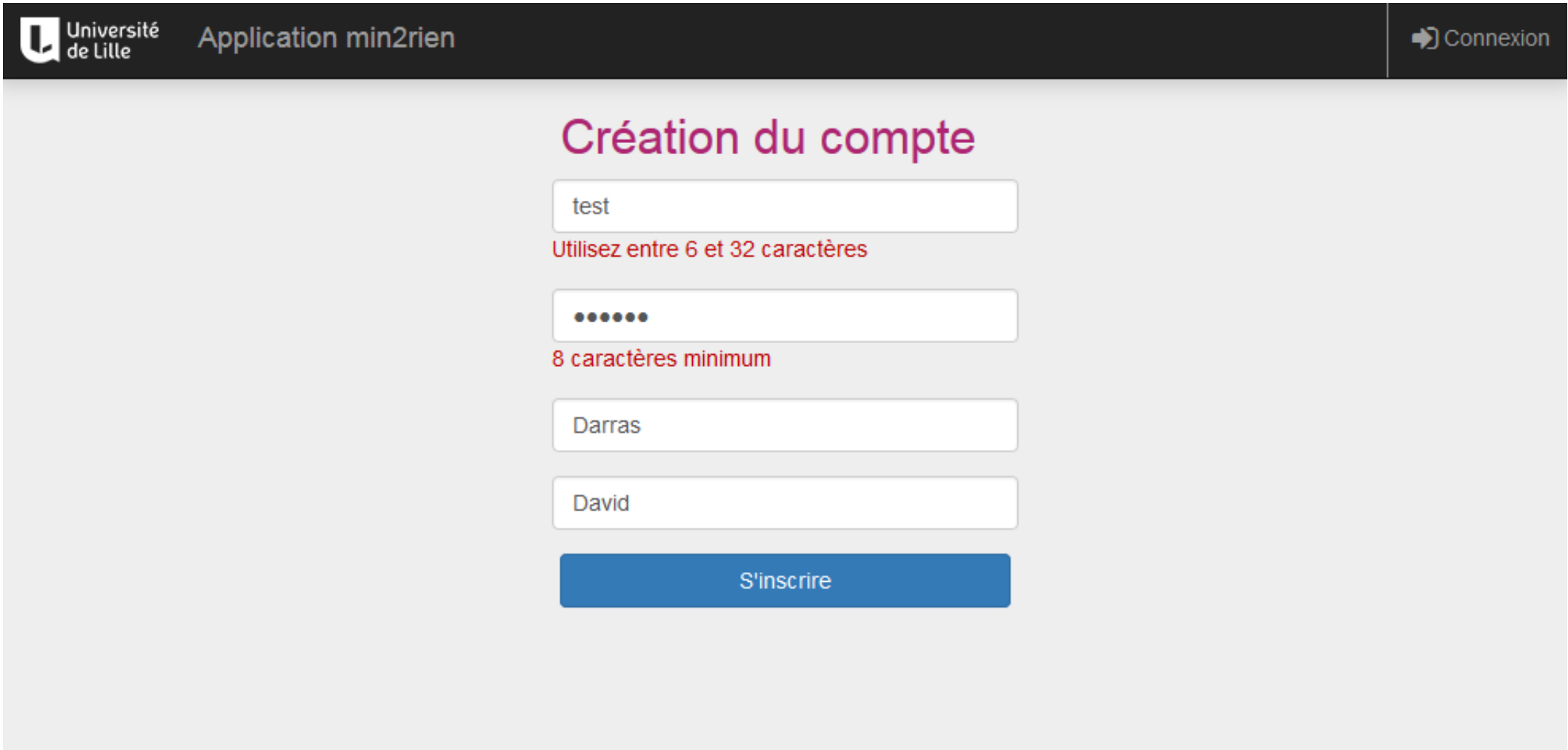
```java
@Controller
public class RegistrationController {
    @Autowired
    private UserService userService;

    @RequestMapping(value = "/registration", method = RequestMethod.GET)
    public String registration(Model model) {
        model.addAttribute("userForm", new User());
        return "registration";
    }

    @RequestMapping(value = "/registration", method = RequestMethod.POST)
    public String registration(@Valid @ModelAttribute("userForm") User userForm,
            BindingResult bindingResult, RedirectAttributes redirectAttributes, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }

        userService.save(userForm);

        redirectAttributes.addFlashAttribute("resultMessage", "Inscription effectuée");
        return "redirect:/";
    }
}
```

Registration.jsp

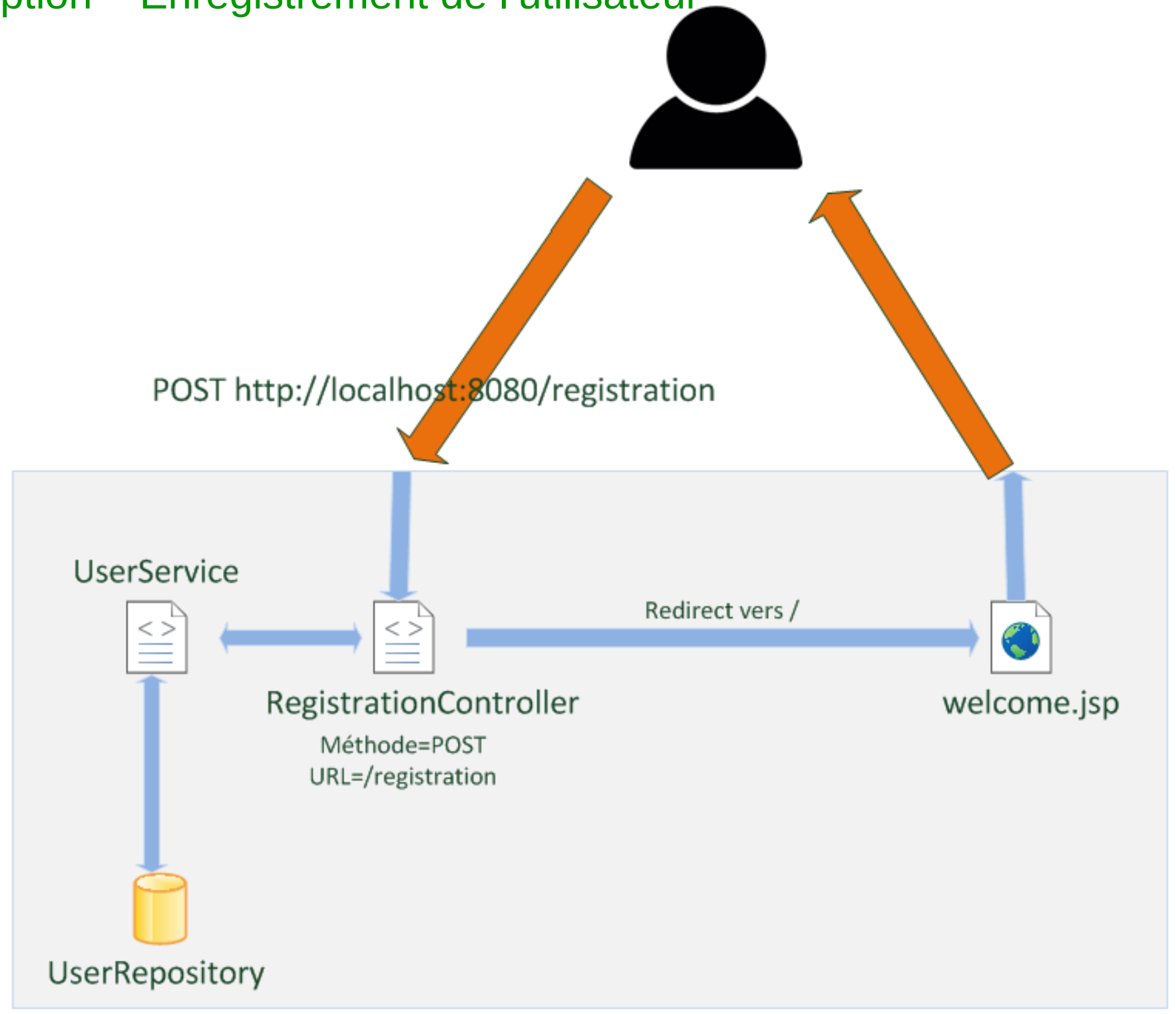
```
<form:form method="POST" modelAttribute="userForm" class="form-horizontal">
    <div class="col-sm-4 col-sm-offset-4">
        <h2 class="form-signin-heading">Création du compte</h2>
    </div>

    <div class="form-group">
        <div class="col-sm-4 col-sm-offset-4">
            <form:input type="text" path="username" class="form-control" placeholder="Nom d'utilisateur"
                        autofocus="true"></form:input>
            <form:errors path="username"></form:errors>
        </div>
    </div>

    <div class="form-group">
        <div class="col-sm-4 col-sm-offset-4">
            <form:input type="password" path="password" class="form-control" placeholder="Mot de passe" />
            <form:errors path="password"></form:errors>
        </div>
    </div>

    ....

    <div class="col-sm-4 col-sm-offset-4">
        <button class="btn btn-primary btn-block" type="submit">S'inscrire</button>
    </div>
</form:form>
```

Inscription – Enregistrement de l'utilisateur




Université de Lille
Application min2rien
Connexion
Création du compte
test
Utilisez entre 6 et 32 caractères
8 caractères minimum
Darras
David
S'inscrire

Inscription – Enregistrement de l'utilisateur



RegistrationController.java

```java
@Controller
public class RegistrationController {
    @Autowired
    private UserService userService;

    @RequestMapping(value = "/registration", method = RequestMethod.GET)
    public String registration(Model model) {
        model.addAttribute("userForm", new User());
        return "registration";
    }

    @RequestMapping(value = "/registration", method = RequestMethod.POST)
    public String registration(@Valid @ModelAttribute("userForm") User userForm,
            BindingResult bindingResult, RedirectAttributes redirectAttributes, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }

        userService.save(userForm);

        redirectAttributes.addFlashAttribute("resultMessage", "Inscription effectuée");
        return "redirect:/";
    }
}
```

User.java

```java
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NotEmpty
    @Size(min = 6, max = 20)
    @Column(name = "username")
    private String username;

    @NotEmpty
    @Size(min = 8)
    @Column(name = "pwd")
    private String password;

    @NotEmpty
    @Column(name = "first_name")
    private String firstname;

    @NotEmpty
    @Column(name = "last_name")
    private String lastname;

    @ManyToMany
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private List<Role> roles;
```

UserServiceImpl.java

```java
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleRepository roleRepository;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Override
    @Transactional
    public void save(User user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        user.setRoles(roleRepository.findAll());
        userRepository.save(user);
    }

    @Override
    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
    }
}
```

Journée « Retours d'expériences »

Questions