

# Registration and Login

Spring Security, Spring Data JPA, Hibernate, MySQL,  
Thymeleaf, Bootstrap, Translations

# Login Page

## Login

Username:

Password:

Log In

Sign Up

# Login Page Validation

## Login

Invalid username or password.

Username:

Password:

Log In

Sign Up

# Registration

## Sign Up

First name

Lastname

Username

Email

Password

Password Confirm

Submit

Login

# Project Dependencies

*<!-- Password Validation -->*

**<dependency>**

**<groupId>**org.passay**</groupId>**






**<artifactId>**passay**</artifactId>**




**<version>**1.0**</version>**

**</dependency>**

# Database Visualization

users_roles	
  <b>users_id</b>	bigint(20)
  <b>roles_id</b>	bigint(20)

users	
  <b>id</b>	bigint(20)
 <b>created_at</b>	datetime
 <b>email</b>	varchar(50)
 <b>first_name</b>	varchar(50)
 <b>last_name</b>	varchar(50)
 <b>password</b>	varchar(255)
 <b>updated_at</b>	datetime
 <b>username</b>	varchar(50)

roles	
  <b>id</b>	bigint(20)
 <b>name</b>	varchar(255)

# Hibernate Entities - User Entity

@Column(length = 50, nullable = false)

private String username;

@Column(length = 50, nullable = false)

private String email;

private String password;

@Transient

private String passwordConfirm;

# Hibernate Entities - Role Entity

@Entity

@Table(name = "roles")

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY, generator = "native")

private Long id;

private String name;

@ManyToMany(mappedBy = "roles")

private Set<User> users;



# Repositories - UserRepository

```
public interface UserRepository extends CrudRepository<User, Long> {  
    Iterable<User> findTop1ByFirstName(String firstName);  
  
    Iterable<User> findTop10ByLastName(String lastName);  
  
    Iterable<User> findTop10ByFirstNameAndLastName(String firstName, String lastName);  
  
    @Query(nativeQuery = true, value = "SELECT * FROM users limit 1")  
    Iterable<User> findAllTopTen();  
  
    Iterable<User> findByFirstName(String firstName);  
  
    User findOneByUsername(String username);  
}
```

# Repositories - RoleRepository

```
public interface RoleRepository extends JpaRepository<Role, Long> {  
  
}
```

# Spring Security UserDetailsService

To implement login/authentication with Spring Security, we need to implement `org.springframework.security.core.userdetails.UserDetailsService` interface

```
public interface UserDetailsService {  
    *  
    * @param username the username identifying the user whose data is required.  
    *  
    * @return a fully populated user record (never null)  
    *  
    * @throws UsernameNotFoundException if the user could not be found or the user has no  
    * GrantedAuthority  
    */  
    UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;  
}
```

# Spring Security UserDetailsService Implementation

@Service

```
public class UserDetailsServiceImpl implements UserDetailsService {
```

@Autowired

```
UserRepository userRepository;
```

@Override

@Transactional(readOnly = true)

```
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
```

```
    User user = userRepository.findOneByUsername(username);
```

```
    if (user == null) {
```

```
        throw new UsernameNotFoundException(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);
```

```
}
```

```
}
```

# Security Service Interface

```
package com.nursultanturdaliev.moneytransferapp.services;
```

```
public interface SecurityService {
```

```
    String findLoggedInUsername();
```

```
    public void autoLogin(String username, String password);  
}
```

# SecurityServiceImpl

```
@Service
public class SecurityServiceImpl implements SecurityService {
    @Override
    public void autoLogin(String username, String password) {
        UserDetails userDetails = userService.loadUserByUsername(username);

        UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(userDetails,
        password, userDetails.getAuthorities());

        authenticationManager.authenticate(usernamePasswordAuthenticationToken);

        if (usernamePasswordAuthenticationToken.isAuthenticated()) {
            SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

            logger.debug(String.format("Auto login %s successull", username));
        }
    }
}
```

<https://gist.github.com/nursultanturdaliev/215526d740c8f208b6085655bcebd9ea>

# UserService Interface

```
package com.nursultanturdaliev.moneytransferapp.services;  
  
import com.nursultanturdaliev.moneytransferapp.model.User;  
  
public interface UserService {  
  
    void save(User user);  
  
    User findOneByUsername(String username);  
}
```

# UserService Implementation

@Service

```
public class UserServiceImpl implements UserService {
```

@Override

```
public void save(User user) {  
    user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));  
    user.setFirstName(user.getFirstName());  
    user.setLastName(user.getLastName());  
    user.setEmail(user.getEmail());  
    user.setUsername(user.getUsername());  
    user.setRoles(new HashSet<>(roleRepository.findAll()));  
    userRepository.save(user);  
}
```

@Override

```
public User findOneByUsername(String username) {  
    return userRepository.findOneByUsername(username);  
}
```



# Spring Validator

@Component

```
public class UserValidator implements Validator {
```

@Autowired

```
private UserService userService;
```

@Override

```
public boolean supports(Class<?> aClass) {return User.class.equals(aClass); }
```

@Override

```
public void validate(Object o, Errors errors) {
```

```
    User user = (User) o;
```

```
    ValidationUtils.rejectIfEmptyOrWhitespace(errors, "password", "NotEmpty");
```

```
    if (user.getPassword().length() < 8 || user.getPassword().length() > 32) {
```

```
        errors.rejectValue("password", "Size.userForm.password");
```

```
    }
```

```
    if (!user.getPasswordConfirm().equals(user.getPassword())) {
```

```
        errors.rejectValue("passwordConfirm", "Diff.userForm.passwordConfirm");
```

```
    }
```

```
}
```

```
}
```

# Registration Controller

```
@GetMapping("/registration")
public String registration(Model model) {
    model.addAttribute("user", new User());
    return "registration";
}
```

```
@PostMapping("/registration")
public String registration(@ModelAttribute("user") User user, Errors errors) {
    validator.validate(user, errors);

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

    userService.save(user);
    securityService.autoLogin(user.getUsername(), user.getPasswordConfirm());
    return "redirect:/";
}
```

```
@GetMapping("/login")
public String login() {
    return "login";
}
```

# Registration Template

<https://gist.github.com/nursultanturdaliev/a4cfc17e5d4db40976e4590ce22f7d45>

# Login Template

<https://gist.github.com/nursultanturdaliev/5d5b2a2c5df7e3f0b241f4d713e4e2da>

# Web Security Configuration

@Override

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

# Web Security Configuration

@Bean

```
public BCryptPasswordEncoder bCryptPasswordEncoder() {  
    return new BCryptPasswordEncoder();  
}
```

@Autowired

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

# Translations - messages.properties

label.user.firstName=First name

label.user.lastName=Lastname

label.user.username=Username

label.user.email=Email

label.user.password=Password

label.user.confirmPass=Password Confirm

label.form.submit=Submit

label.form.loginLink=Login

label.form.title=Sign Up

error.wordLength=Your password is too short

error.wordNotEmail=Do not use your email as your password

error.wordSequences=Your password contains sequences

error.wordLowercase=Use lower case characters

error.wordUppercase=Use upper case characters

error.wordOneNumber=Use numbers

error.wordOneSpecialChar=Use special characters

PasswordMatches.user=Password does not match!

NotEmpty=This field is required.

Size.userForm.username=Please use between 6 and 32 characters.

Duplicate.userForm.username=Someone already has that username.

Size.userForm.password=Try one with at least 8 characters.

Diff.userForm.passwordConfirm=These passwords don't match.

# References

- <https://stackoverflow.com/questions/14014086/what-is-difference-between-crudrepository-and-jpaRepository-interfaces-in-spring>
- <https://medium.com/@devquora/login-registration-example-with-spring-boot-f5f76459c59d>
- <https://www.baeldung.com/transaction-configuration-with-jpa-and-spring>
- <https://spring.io/guides/gs/handling-form-submission/>