Name: Deepti Sardar
Reg no.:11915397
Course: MCA[LE]

SPRING SECURITY ASSIGNMENT

Q1. Design and develop a Spring security Hello World application by using default login form provided by spring security to secure URL access say, for example to access the content of an "admin" page, user needs to enter valid credentials. User must also logged out if successfully logged in. Use Java Based and annotation based configuration and In-memory authentication.

HomeResource.java

```
package.com.capgemini.SpringSecurity;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeResource {
        // for others



         @RequestMapping(value="/", method=RequestMethod.GET)
public String
         sayHelloWorld() {

         return ("<h1>HELLO WORLD</h1>"); }

         // for users

         @RequestMapping(value="/user",
method=RequestMethod.GET) public String
         sayHelloUser() {

         return ("<h1>HELLO USER</h1>"); }

         // for admins

         @RequestMapping(value="/admin",
method=RequestMethod.GET) public String
         sayHelloAdmin() {

         return ("<h1>HELLO ADMIN</h1>"); }

}
```

SecurityConfiguration.java

```
package.com.cg.SpringSe
curity;

import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authe
nticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSe
curity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurity
ConfigurerAdapter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter{


	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

		// authentication using embedded database H2


		auth.inMemoryAuthentication().withUser("akshat")
	 .password("akshat")
	 .roles("USER")
	 .and()
	 .withUser("admin")
	 .password("admin")
	 .roles("ADMIN");
		//auth.userDetailsService(userDetailsService);
	}


	// authorization
	@Override
	protected void configure(HttpSecurity http) throws Exception {
```

```
                // setting the authorization for the roles USER and ADMIN

                 http.authorizeRequests()
        .antMatchers("/admin").hasRole("ADMIN")

        .antMatchers("/user").hasAnyRole("USER","ADMIN")

        .antMatchers("/").permitAll()
                                                             .and()
                                                             .formLogin() ;

            }

            @Bean
            public PasswordEncoder getEncoder() {

                return NoOpPasswordEncoder.getInstance();
            }


        }
```

SpringSecurityJdbcApplication.java

```
package
io.javabrains.springsecurityjdbc;

                import org.springframework.boot.SpringApplication;
                import
                org.springframework.boot.autoconfigure.SpringBootApplication;

                @SpringBootApplication
                public class SpringSecurityJdbcApplication {

                    public static void main(String[] args) {

                    SpringApplication.run(SpringSecurityJdbcApplication.class,
                args);
                    }

                }
```

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
          https://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>2.3.8.RELEASE</version>
	<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>io.javabrains</groupId>
<artifactId>spring-security-jdbc</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring-security-jdbc</name>
<description>Demo project for Spring Boot</description>
<properties>
	<java.version>15</java.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-jdbc</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-security</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>
	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
		<scope>runtime</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>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.springframework.security</groupId>
		<artifactId>spring-security-test</artifactId>
		<scope>test</scope>
```

```
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
        </dependencies>

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

</project>
```

Q2. Modify the above application to use custom login form instead of default login form provided by spring security. Use Java Based and annotation based configuration and In-memory authentication.

HomeResource.java

```
package.com.cg.springsecurity;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeResource {

    @GetMapping("/")
    public String home() {
        return ("<h1>Welcome</h1>");
    }

    @GetMapping("/user")
    public String user() {
        return ("<h1>Welcome User</h1>");
    }

    @GetMapping("/admin")
    public String admin() {
        return ("<h1>Welcome Admin</h1>");
    }
}
```

SecurityConfiguration.java

```
package.com.cg.springsecurity;
 import
 javax.sql.DataSource;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authe
nticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSe
curity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurity
ConfigurerAdapter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

       @Autowired
       DataSource dataSource;



       @Override
       protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

              // authentication using embedded database H2

                auth.jdbcAuthentication().dataSource(dataSource)
.withDefaultSchema()

.withUser(User.withUsername("akshat").password("akshat").roles("USER"))

.withUser(User.withUsername("admin").password("admin").roles("ADMIN")) ;


              //auth.userDetailsService(userDetailsService);
       }



       // authorization
       @Override
       protected void configure(HttpSecurity http) throws Exception {

              // setting the authorization for the roles USER and ADMIN
              /*
```

```
                    * http.authorizeRequests()
.antMatchers("/admin").hasRole("ADMIN")
                    * .antMatchers("/user").hasAnyRole("USER","ADMIN")
                    * .antMatchers("/").permitAll() .and().formLogin() ;
                    */

                                    http.authorizeRequests()
                                            .anyRequest().authenticated()

        .and().formLogin().loginPage("/login").permitAll();
        }



        @Bean
        public PasswordEncoder getEncoder() {

                return NoOpPasswordEncoder.getInstance();
        }


}
```

SpringSecurityJdbcApplication.java

```
package.com.cg.springsecurity;



import
org.springframework.boot.SpringApplicat
ion;
import
org.springframework.boot.autoconfigure.SpringBootApplication
;

@SpringBootApplication
public class SpringSecurityJdbcApplication {

        public static void main(String[] args) {

        SpringApplication.run(SpringSecurityJdbcApplication.
class, args);
        }

}
```

login.html

```
<html
xmlns:th="https://www.thymeleaf.org">
        <head th:include="layout ::
      head(title=~{::title},links=~{})">
```

```
    <title>Please Login</title>
  </head>
  <body th:include="layout :: body"
th:with="content=~{::content}">
    <div th:fragment="content">
        <form name="f" th:action="@{/login}" method="post">
            <fieldset>
                <legend>Custom Login Form</legend>
                <div th:if="${param.error}" class="alert
alert-error">
                    Invalid username and password.
                </div>
                <div th:if="${param.logout}" class="alert
alert-success">
                    You have been logged out.
                </div>
                <label for="username">Username</label>
                <input type="text" id="username"
name="username"/>
                <label for="password">Password</label>
                <input type="password" id="password"
name="password"/>
                <div class="form-actions">
                    <button type="submit" class="btn">Log
in</button>
                </div>
            </fieldset>
        </form>
    </div>
  </body>
</html>
```

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
https://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>2.3.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>io.javabrains</groupId>
	<artifactId>spring-security-jdbc</artifactId>
	<version>0.0.1-SNAPSHOT</version>
```

```
<name>spring-security-jdbc</name>
<description>Demo project for Spring Boot</description>
<properties>
        <java.version>15</java.version>
</properties>
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-jdbc</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-security</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</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>
                </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>

        </project>
```
<u>Q3.</u> Modify the above application and use database authentication using JDBC instead of In-memory authentication. Use Java Based and annotation based configuration and In-memory authentication.

HomeResource.java

```
package.com.capgemini.SpringSecurity;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeResource {
        // for others
                @RequestMapping(value="/",
method=RequestMethod.GET)
                public String sayHelloWorld() {

                        return ("<h1>HELLO WORLD</h1>");
                }

                // only for users
                @RequestMapping(value="/user",
method=RequestMethod.GET)
                public String sayHelloUser() {

                        return ("<h1>HELLO USER</h1>");
                }

                // only for admins
                @RequestMapping(value="/admin",
method=RequestMethod.GET)
                public String sayHelloAdmin() {

                        return ("<h1>HELLO ADMIN</h1>");
                }
}
```

SecurityConfiguration.java

```
package.com.capgemini.Spri
ngSecurity;

import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import
org.springframework.security.config.annotation.authentication.builders.A
uthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity
;
import
org.springframework.security.config.annotation.web.configuration.EnableW
ebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

       @Autowired
       DataSource dataSource;

       // authentication using embedded database H2
       @Override
       protected void configure(AuthenticationManagerBuilder auth)
throws Exception {

              auth.jdbcAuthentication().dataSource(dataSource)

.withDefaultSchema()

.withUser(User.withUsername("akshat").password("akshat").roles("USER"))

.withUser(User.withUsername("admin").password("admin").roles("ADMIN"))
                                                         ;
       }



       // authorization
       @Override
       protected void configure(HttpSecurity http) throws Exception {

              // setting the authorization for the roles USER and ADMIN
                            http.authorizeRequests()

       .antMatchers("/admin").hasRole("ADMIN")

       .antMatchers("/user").hasAnyRole("USER","ADMIN")
                                   .antMatchers("/").permitAll()
                                   .and().formLogin();
```

```
        }

        @Bean
        public PasswordEncoder getEncoder() {

                return NoOpPasswordEncoder.getInstance();
        }


}
```

SpringSecurityJdb.java

```
package
com.capgemini.SpringSecurity;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringSecurityJdbcApplication {

        public static void main(String[] args) {

        SpringApplication.run(SpringSecurityJdbcApplication.class,
args);
        }

}
```

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
https://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>2.3.8.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>io.javabrains</groupId>
        <artifactId>spring-security-jdbc</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>spring-security-jdbc</name>
        <description>Demo project for Spring Boot</description>
        <properties>
```

```xml
                <java.version>15</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-jdbc</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-web</artifactId>
                </dependency>

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

</project>
```

Q4. Modify the above application to limit the number of login attempts. ,

Q5.Modify the above application to implement “remember me” functionality. ,

Q6. Modify the above application to secure the password by encoding it. (You may use some encryption technique)

User.java

```
package
spring.security.springsecu
rity;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.LockedException;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.web.authentication.SimpleUrlAuthenticationFa
ilureHandler;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class CustomLoginFailureHandler extends
SimpleUrlAuthenticationFailureHandler {

    @Autowired
    UserService userService;

    @Override
    public void onAuthenticationFailure(HttpServletRequest request,
HttpServletResponse response, AuthenticationException exception) throws
IOException, ServletException {

        String username=request.getParameter("username");
        User user =userService.getUserByName(username);
        if(user!=null){
            if(user.isActive() && user.isAccountNonLocked()){
                if(user.getFailedAttempt()<3){
                    userService.increaseFailedAttempts(user);
                }
                else{
                    userService.lock(user);
                    exception = new LockedException("Account has been
locked due to 3 unsuccessful login attempt");
                }

            }
        }
        else{
            System.out.println("username not exists");
        }

        super.setDefaultFailureUrl("/login?error");
        super.onAuthenticationFailure(request, response, exception);

    }
}
```

HomeController.java

```
package
spring.security.springsecurity;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeResource {

    @GetMapping("/")
    public String home() {
        return ("<h1>Welcome</h1>");
    }

    @GetMapping("/user")
    public String user() {
        return ("<h1>Welcome User</h1>");
    }

    @GetMapping("/admin")
    public String admin() {
        return ("<h1>Welcome Admin</h1>");
    }
}
```

LoginController.java

```
package
spring.security.springsecurity;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class LoginController {

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String login(Model model, String error, String logout) {
        if (error != null)
            model.addAttribute("errorMsg", "Your username and
password are invalid.");

        if (logout != null)
            model.addAttribute("msg", "You have been logged out
successfully.");

        return "login";
    }

    @RequestMapping(value = "/failure")
    public String failure(Model model){

        model.addAttribute("loginError",true);
        model.addAttribute("exception",true);
        return "login";
    }
}
```

UserDetails.java

```
package
spring.security.springsecurity;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;

public class MyUserDetails implements UserDetails {
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return null;
    }

    @Override
    public String getPassword() {
        return null;
    }

    @Override
    public String getUsername() {
        return null;
    }

    @Override
    public boolean isAccountNonExpired() {
        return false;
    }

    @Override
    public boolean isAccountNonLocked() {
        return false;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return false;
    }

    @Override
    public boolean isEnabled() {
        return false;
    }
}
```

UserDetailService.java

```
package
spring.security.springsecurity
;
```

```
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundExcepti
on;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class MyuserDetailService implements UserDetailsService {
    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {

        Optional<User> user =userRepository.findByUsername(username);
        user.orElseThrow(()-> new UsernameNotFoundException("Not
found"));

        return user.map(MyUserDetails::new).get();

    }
}
```

SpringSecurityConfig.java

```
package
spring.security.springs
ecurity;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import
org.springframework.security.config.annotation.authentication.builders.Authe
nticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSe
curity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurity
ConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

import javax.sql.DataSource;

@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Autowired
    PasswordEncoder passwordEncoder;

    @Autowired
    private DataSource dataSource;

    @Autowired
    CustomLoginFailureHandler customLoginFailureHandler;

    @Autowired
    UserDetailsService userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.userDetailsService(userDetailsService);
    }

    @Bean
    public PasswordEncoder getPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()

                .antMatchers("/admin").hasRole("ADMIN")
                .antMatchers("/user").hasAnyRole("ADMIN", "USER")
                .antMatchers("/").permitAll()
                .antMatchers("/login**").permitAll()
                .and().formLogin()
                .loginPage("/login")
                .permitAll()
                .failureHandler(customLoginFailureHandler)
                .permitAll()
                .and()
                .logout()
                .permitAll();


    }
}
```

UserRepository.java

```
package
spring.security.springsecurity;

import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User,Integer>
{
    Optional<User> findByUsername(String username);
}


package spring.security.springsecurity;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Service;

@Service
public class UserService {
    @Autowired
    UserRepository userRepository;

    public User getUserByName(String username){
        return  userRepository.findByUsername(username).get();
    }

    public void increaseFailedAttempts(User user) {
        int newFailedAttemp = user.getFailedAttempt()+1;
        user.setFailedAttempt(newFailedAttemp);
        userRepository.save(user);
    }

    public void lock(User user) {
        user.setAccountNonLocked(false);
        user.setLockTime(new Date());
        userRepository.save(user);
    }
}
```

UserService.java

```
package
spring.security.springsecurity;

@Entity
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String username;
    private String password;
    private boolean active;
    private String roles;

    public User(){}

    public User(int id, String username, String password, boolean
active, String roles, boolean accountNonLocked, int failedAttempt,
Date lockTime) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.active = active;
        this.roles = roles;
        this.accountNonLocked = accountNonLocked;
        this.failedAttempt = failedAttempt;
        this.lockTime = lockTime;
    }

    @Column(name = "account_no_locker")
    private boolean accountNonLocked;

    @Column(name = "failed_attempt")
    private int failedAttempt;

    @Column(name = "lock_time")
    private Date lockTime;



    public boolean isAccountNonLocked() {
        return accountNonLocked;
    }

    public void setAccountNonLocked(boolean accountNonLocked) {
        this.accountNonLocked = accountNonLocked;
    }

    public int getFailedAttempt() {
        return failedAttempt;
    }

    public void setFailedAttempt(int failedAttempt) {
        this.failedAttempt = failedAttempt;
    }

    public Date getLockTime() {
        return lockTime;
    }

    public void setLockTime(Date lockTime) {
        this.lockTime = lockTime;
    }

    public int getId() {
        return id;
    }

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

    public String getUserName() {
        return username;
    }

    public void setUserName(String userName) {
        this.username = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public boolean isActive() {
        return active;
    }
    public void setActive(boolean active) {
        this.active = active;
    }

    public String getRoles() {
        return roles;
    }

    public void setRoles(String roles) {
        this.roles = roles;
    }
```

SpringSecurityJPA.java

```
package
io.javabrains.springsecurityjpa
;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.data.jpa.repository.config.EnableJpaRepositories
;

@SpringBootApplication
@EnableJpaRepositories(basePackageClasses = UserRepository.class)
public class SpringSecurityJpaApplication {

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

}
```