1

package spring.security.com;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HomeResource {  
  
 @GetMapping("/")  
 public String home() {  
  
 return ("<h1>Hello World!</h1>");  
 }  
}

package spring.security.com;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringBootSecurityApplication {  
  
 public static void main(String[] args) {  
  
 SpringApplication.*run*(SpringBootSecurityApplication.class, args);  
 }  
  
}

2

package com.spring.security2;  
  
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>");  
 }  
}

package com.spring.security2;  
  
import org.springframework.context.annotation.Bean;  
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.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
@EnableWebSecurity  
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 // Set your configuration on the auth object  
 auth.inMemoryAuthentication()  
 .withUser("ram")  
 .password("win")  
 .roles("USER")  
 .and()  
 .withUser("arjun")  
 .password("won")  
 .roles("ADMIN");  
 }  
  
 @Bean  
 public PasswordEncoder getPasswordEncoder() {  
  
 return NoOpPasswordEncoder.*getInstance*();  
 }

@Override  
protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/admin").hasRole("ADMIN")  
 .antMatchers("/user").hasAnyRole("ADMIN", "USER")  
 .antMatchers("/").permitAll()  
 .and().formLogin()  
 .and().formLogin().loginPage("/login")  
 .failureUrl("/login.jsp?error=1").loginProcessingUrl("/login")  
 .permitAll().and().logout();  
  
 http.csrf().disable();

}  
  
 }

package com.spring.security2;  
  
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";  
 }  
}

package com.spring.security2;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringBootSecurityApplication {  
  
 public static void main(String[] args) {  
  
 SpringApplication.*run*(SpringBootSecurityApplication.class, args);  
 }  
  
}

3

package spring.security.jdbc.springsecurityjdbc;  
  
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>");  
 }  
}

package spring.security.jdbc.springsecurityjdbc;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
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.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
import javax.sql.DataSource;  
  
@EnableWebSecurity  
public class springSecurityConfiguration extends WebSecurityConfigurerAdapter {  
  
@Autowired  
 DataSource dataSource;  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.jdbcAuthentication()  
 .dataSource(dataSource)

.withDefaultSchema()

.withUser("blah")  
 .password("blah")  
 .roles("USER")  
 .and()  
 .withUser("foo")  
 .password("foo")  
 .roles("ADMIN");  
}

}  
  
 @Override  
protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/admin").hasRole("ADMIN")  
 .antMatchers("/user").hasAnyRole("ADMIN", "USER")  
 .antMatchers("/").permitAll()  
 .and().formLogin()  
 .and().formLogin().loginPage("/login")  
 .failureUrl("/login.jsp?error=1").loginProcessingUrl("/login")  
 .permitAll().and().logout();  
  
 http.csrf().disable();

}  
 @Bean  
 public PasswordEncoder getPasswordEncoder() {  
 return NoOpPasswordEncoder.*getInstance*();  
 }  
  
}

package io.javabrains.springbootsecurity;  
  
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";  
 }  
}

package spring.security.jdbc.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);  
 }  
  
}

Q-4,5,6

package spring.security.springsecurity;  
  
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.SimpleUrlAuthenticationFailureHandler;  
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);  
  
 }  
}

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>");  
 }  
}

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";  
 }  
}

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;  
 }  
}

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.UsernameNotFoundException;  
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();  
  
 }  
}

package spring.security.springsecurity;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
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.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
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();  
  
  
 }  
}

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);  
 }  
}

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;  
 }

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);  
 }  
  
}