

Spring MVC Return Previous Page After Successful Login

Back to previous page after successful login, we can use following custom authentication manager as follows:

```
1 <!-- enable use-expressions -->
2 <http auto-config="true" use-expressions="true">
3 <!-- src** matches: src/bar.c src/baz.c src/test/bartest.c-->
4 <intercept-url pattern="/problemSolution/home/**" access="hasRole('ROLE_ADMIN')"/>
5 <intercept-url pattern="favicon.ico" access="permitAll"/>
6 <form-login
7     authentication-success-handler-ref="authenticationSuccessHandler"
8     always-use-default-target="true"
9     login-processing-url="/checkUser"
10    login-page="/problemSolution/index"
11
12    default-target-url="/problemSolution/home"
13    authentication-failure-url="/problemSolution/index?error"
14    username-parameter="username"
15    password-parameter="password"/>
16 <logout logout-url="/problemSolution/logout"
17     logout-success-url="/problemSolution/index?logout"/>
18 <!-- enable csrf protection -->
19 <csrf/>
20 </http>
21
22 <beans:bean id="authenticationSuccessHandler"
23     class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticati
24 onSuccessHandler">
25     <beans:property name="defaultTargetUrl" value="/problemSolution/home"/>
26 </beans:bean>
27
28 <!-- Select users and user_roles from database -->
29 <authentication-manager>
30     <authentication-provider user-service-ref="customUserDetailsService">
31         <password-encoder hash="plaintext">
32             </password-encoder>
33     </authentication-provider>
</authentication-manager>
```

CustomUserDetailsService class

```
1 @Service
2 public class CustomUserDetailsService implements UserDetailsService {
3
4     @Autowired
5     private UserService userService;
6
7     public UserDetails loadUserByUsername(String userName)
8         throws UsernameNotFoundException {
9         com.codesenior.telif.local.model.User domainUser = userService.getUser(userName);
10
11         boolean enabled = true;
12         boolean accountNonExpired = true;
13         boolean credentialsNonExpired = true;
14         boolean accountNonLocked = true;
15
16         return new User(
17             domainUser.getUsername(),
18             domainUser.getPassword(),
19             enabled,
20             accountNonExpired,
21             credentialsNonExpired,
22             accountNonLocked,
```

```

23         getAuthorities(domainUser.getUserRoleList())
24     );
25 }
26
27 public Collection<? extends GrantedAuthority> getAuthorities(List<UserRole> userRoleList)
28 {
29     return getGrantedAuthorities(getRoles(userRoleList));
30 }
31
32 public List<String> getRoles(List<UserRole> userRoleList) {
33     List<String> roles = new ArrayList<String>();
34
35     for(UserRole userRole:userRoleList){
36         roles.add(userRole.getRole());
37     }
38     return roles;
39 }
40
41 public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
42     List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
43
44     for (String role : roles) {
45         authorities.add(new SimpleGrantedAuthority(role));
46     }
47     return authorities;
48 }
49 }
50 }

```

User Class

```

1  import com.codesenior.telif.local.model.UserRole;
2  import org.springframework.beans.factory.annotation.Autowired;
3  import org.springframework.security.core.GrantedAuthority;
4  import org.springframework.security.core.authority.SimpleGrantedAuthority;
5  import org.springframework.security.core.userdetails.User;
6  import org.springframework.security.core.userdetails.UserDetails;
7  import org.springframework.security.core.userdetails.UserDetailsService;
8  import org.springframework.security.core.userdetails.UsernameNotFoundException;
9  import org.springframework.stereotype.Service;
10
11 import java.util.ArrayList;
12 import java.util.Collection;
13 import java.util.List;
14
15
16 @Service
17 public class CustomUserDetailsService implements UserDetailsService {
18
19     @Autowired
20     private UserService userService;
21
22     public UserDetails loadUserByUsername(String userName)
23         throws UsernameNotFoundException {
24         com.codesenior.telif.local.model.User domainUser = userService.getUser(userName);
25
26         boolean enabled = true;
27         boolean accountNonExpired = true;
28         boolean credentialsNonExpired = true;
29         boolean accountNonLocked = true;
30
31         return new User(
32             domainUser.getUsername(),
33             domainUser.getPassword(),
34             enabled,
35             accountNonExpired,

```

```

36         credentialsNonExpired,
37         accountNonLocked,
38         getAuthorities(domainUser.getUserRoleList())
39     );
40 }
41
42 public Collection<? extends GrantedAuthority> getAuthorities(List<UserRole> userRoleList)
43 {
44     return getGrantedAuthorities(getRoles(userRoleList));
45 }
46
47 public List<String> getRoles(List<UserRole> userRoleList) {
48     List<String> roles = new ArrayList<String>();
49
50     for(UserRole userRole:userRoleList){
51         roles.add(userRole.getRole());
52     }
53     return roles;
54 }
55
56 public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
57     List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
58
59     for (String role : roles) {
60         authorities.add(new SimpleGrantedAuthority(role));
61     }
62     return authorities;
63 }
64 }
65 }

```

UserRole Class

```

1  @Entity
2  public class UserRole {
3
4      @Id
5      @GeneratedValue
6      private Integer userRoleId;
7
8      private String role;
9
10     @ManyToMany(fetch = FetchType.LAZY, mappedBy = "userRoleList")
11     @JsonIgnore
12     private List<User> userList;
13
14     public Integer getUserRoleId() {
15         return userRoleId;
16     }
17
18     public void setUserRoleId(Integer userRoleId) {
19         this.userRoleId= userRoleId;
20     }
21
22     public String getRole() {
23         return role;
24     }
25
26     public void setRole(String role) {
27         this.role= role;
28     }
29
30     @Override
31     public String toString() {
32         return String.valueOf(userRoleId);
33     }

```

```
34
35     public List<User> getUserList() {
36         return userList;
37     }
38
39     public void setUserList(List<User> userList) {
40         this.userList = userList;
41     }
42 }
```