Spring MVC Return Previous Page After Successfull Login

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

```
    <!-- enable use-expressions -->
        <http auto-config="true" use-expressions="true">
            <!-- src** matches: src/bar.c src/baz.c src/test/bartest.c-->
            <intercept-url pattern="/problemSolution/home/**" access="hasRole('ROLE_ADMIN')"/>
            <intercept-url pattern="favicon.ico" access="permitAll"/>
            <form-login
                    authentication-success-handler-ref="authenticationSuccessHandler"
                    always-use-default-target="true"
                    login-processing-url="/checkUser"
                    login-page="/problemSolution/index"

                    default-target-url="/problemSolution/home"
                    authentication-failure-url="/problemSolution/index?error"
                    username-parameter="username"
                    password-parameter="password"/>
            <logout logout-url="/problemSolution/logout"
                    logout-success-url="/problemSolution/index?logout"/>
            <!-- enable csrf protection -->
            <csrf/>
        </http>

        <beans:bean id="authenticationSuccessHandler"
                class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticati
onSuccessHandler">
            <beans:property name="defaultTargetUrl" value="/problemSolution/home"/>
        </beans:bean>

        <!-- Select users and user_roles from database -->
        <authentication-manager>
            <authentication-provider user-service-ref="customUserDetailsService">
                <password-encoder hash="plaintext">
                </password-encoder>
            </authentication-provider>
        </authentication-manager>
```

CustomUserDetailsService class

```
@Service
public class CustomUserDetailsService implements UserDetailsService {

     @Autowired
     private UserService userService;

     public UserDetails loadUserByUsername(String userName)
             throws UsernameNotFoundException {
         com.codesenior.telif.local.model.User domainUser = userService.getUser(userName);

         boolean enabled = true;
         boolean accountNonExpired = true;
         boolean credentialsNonExpired = true;
         boolean accountNonLocked = true;

         return new User(
                 domainUser.getUsername(),
                 domainUser.getPassword(),
                 enabled,
                 accountNonExpired,
                 credentialsNonExpired,
                 accountNonLocked,
```

```
                getAuthorities(domainUser.getUserRoleList())
        );
    }

    public Collection<? extends GrantedAuthority> getAuthorities(List<UserRole> userRoleList)
{
        return getGrantedAuthorities(getRoles(userRoleList));
    }

    public List<String> getRoles(List<UserRole> userRoleList) {

        List<String> roles = new ArrayList<String>();

        for(UserRole userRole:userRoleList){
            roles.add(userRole.getRole());
        }
        return roles;
    }

    public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

        for (String role : roles) {
            authorities.add(new SimpleGrantedAuthority(role));
        }
        return authorities;
    }

}
```

User Class

```
import com.codesenior.telif.local.model.UserRole;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
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.ArrayList;
import java.util.Collection;
import java.util.List;


@Service
public class CustomUserDetailsService implements UserDetailsService {

    @Autowired
    private UserService userService;

    public UserDetails loadUserByUsername(String userName)
            throws UsernameNotFoundException {
        com.codesenior.telif.local.model.User domainUser = userService.getUser(userName);

        boolean enabled = true;
        boolean accountNonExpired = true;
        boolean credentialsNonExpired = true;
        boolean accountNonLocked = true;

        return new User(
                domainUser.getUsername(),
                domainUser.getPassword(),
                enabled,
                accountNonExpired,
```

```
                credentialsNonExpired,
                accountNonLocked,
                getAuthorities(domainUser.getUserRoleList())
        );
    }

    public Collection<? extends GrantedAuthority> getAuthorities(List<UserRole> userRoleList)
{
        return getGrantedAuthorities(getRoles(userRoleList));
    }

    public List<String> getRoles(List<UserRole> userRoleList) {

        List<String> roles = new ArrayList<String>();

        for(UserRole userRole:userRoleList){
            roles.add(userRole.getRole());
        }
        return roles;
    }

    public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

        for (String role : roles) {
            authorities.add(new SimpleGrantedAuthority(role));
        }
        return authorities;
    }

}
```

UserRole Class

```
@Entity
public class UserRole {

    @Id
    @GeneratedValue
    private Integer userRoleId;

    private String role;

    @ManyToMany(fetch = FetchType.LAZY, mappedBy = "userRoleList")
    @JsonIgnore
    private List<User> userList;

    public Integer getUserRoleId() {
        return userRoleId;
    }

    public void setUserRoleId(Integer userRoleId) {
        this.userRoleId= userRoleId;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role= role;
    }

    @Override
    public String toString() {
        return String.valueOf(userRoleId);
    }
```

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