

Spring Security Multi-Role Support

Question:

... but would you update yours with different roles? cuz multi roles with registratioj still not working for me.

Solution:

This example shows how to handle multi-role support for new user registration.

On registration form, we want to add a user with a different role. Instead of using default "employee" role.

On the registration form, there would be a drop down as shown below

The image displays two versions of a 'Register New User' form. Both forms have a blue header with the title 'Register New User'. Below the header, there are two input fields: 'username' (with a person icon) and 'password' (with a lock icon). Below the password field is a dropdown menu. In the left screenshot, the dropdown menu is closed and shows 'Manager'. In the right screenshot, the dropdown menu is open, showing a list of options: 'Employee' (with a checkmark), 'Manager', 'Admin', and 'Register'. A red callout bubble with the text 'Select role' points to the dropdown menu in both screenshots.

Download Solution Source Code

<http://www.luv2code.com/spring-security-multi-role>

For this solution, I made updates to the following files:

1. CrmUser.java
2. RegistrationController.java
3. registration-form.jsp

Details

1. CrmUser.java

CrmUser is a model attribute for the form. We need to modify the file have a new field for role. Simply add a new field and appropriate getter/setter methods.

```
public class CrmUser {  
  
    private String formRole;  
  
    ...  
  
    public String getFormRole() {  
        return formRole;  
    }  
  
    public void setFormRole(String formRole) {  
        this.formRole = formRole;  
    }  
  
}
```

2. RegistrationController.java – Load Roles

In this file, we need to create a collection of roles that will be displayed on the registration form. We can make use of `@PostConstructor` to initialize the collection of roles.

```
public class RegistrationController {  
  
    ...  
  
    private Map<String, String> roles;  
  
    @PostConstruct  
    protected void loadRoles() {  
  
        // using hashmap, could also read this info from a database  
  
        roles = new LinkedHashMap<String, String>();  
  
        // key=the role, value=display to user  
        roles.put("ROLE_EMPLOYEE", "Employee");  
        roles.put("ROLE_MANAGER", "Manager");  
        roles.put("ROLE_ADMIN", "Admin");  
    }  
  
    ...  
}
```

We also need to add the roles to the model when we show the form initially

```
@GetMapping("/showRegistrationForm")  
public String showMyLoginPage(Model theModel) {  
  
    theModel.addAttribute("crmUser", new CrmUser());  
  
    // add roles to the model for form display  
    theModel.addAttribute("roles", roles);  
  
    return "registration-form";  
}
```

3. registration-form.jsp

We need to update the registration form to display the list of roles. The roles were populated in the controller.

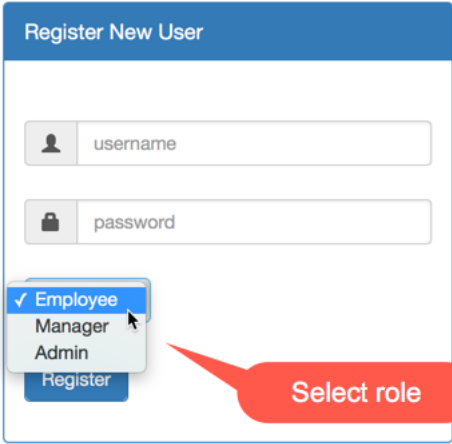
In the form we can use the code

```
<!-- Roles -->
<div style="margin-bottom: 25px" class="input-group">

    <form:select path="formRole" items="${roles}" class="form-control" />

</div>
```

This will give the HTML view



The screenshot shows a web form titled "Register New User". It contains three input fields: "username" (with a person icon), "password" (with a lock icon), and a role selection dropdown. The dropdown menu is open, showing four options: "Employee" (selected with a checkmark), "Manager", "Admin", and "Register". A red callout bubble with the text "Select role" points to the dropdown menu.

4. RegistrationController.java – Process Form

Once the user enters data in the form, then the controller needs to add the new user.

This is accomplished in the method:

```
@PostMapping("/processRegistrationForm")
public String processRegistrationForm(
    @Valid @ModelAttribute("crmUser") CrmUser theCrmUser,
    BindingResult theBindingResult,
    Model theModel) {
    ...

    // give user default role of "employee"
    List<GrantedAuthority> authorities = AuthorityUtils.createAuthorityList();
    authorities.add(new SimpleGrantedAuthority("ROLE_EMPLOYEE"));

    // if the user selected role other than employee
    // then add that one too (multiple roles)
    String formRole = theCrmUser.getFormRole();

    if (!formRole.equals("ROLE_EMPLOYEE")) {
        authorities.add(new SimpleGrantedAuthority(formRole));
    }

    // create user object (from Spring Security framework)
    User tempUser = new User(userName, encodedPassword, authorities);
    ...
}
```

The important piece is that the user needs to have ROLE_EMPLOYEE plus the other roles selected from the form. This is because our home page of the secure site is based on ROLE_EMPLOYEE.

Also, make note of code mods for the case of form validation fails, you need to also add the roles to the model.

```
// form validation
if (theBindingResult.hasErrors()) {

    theModel.addAttribute("crmUser", new CrmUser());

    // add roles to the model for form display
    theModel.addAttribute("roles", roles);
    ...
}
```

That's it!