Spring Security UI Plugin - Reference Documentation

Authors: Burt Beckwith
Version: 1.0-RC2

Table of Contents

1 Introduction to the Spring Security UI Plugin

2 User Management

3 Role Management

4 Requestmap Management

5 User Registration

6 Forgot Password

7 ACL Management

- **7.1** AclClass Management
- **7.2** AclSid Management
- **7.3** AclObjectIdentity Management
- **7.4** AclEntry Management

8 Persistent Cookie Management

9 Security Configuration UI

10 Customization

1 Introduction to the Spring Security UI Plugin

The Spring Security UI plugin provides CRUD screens and other user management workflows. Non-defa
includes the ACL controllers and views which are enabled if the ACL plugin is in
`grails.plugin.springsecurity.securityConfigType` is set to `"Requestmap"` or `Se`
and persistent cookies support which is enabled if it has been configured with the `s2-create-persist`

As of version 0.2 the plugin does not declare any dependencies on plugins, although it has sev

Name	Minimum Version
spring-security-core	2.0
mail	1.0
jquery	1.10.2
jquery-ui	1.10.3
famfamfam	1.0

This is to support both 1.3.x and 2.x applications without specifying a fixed version. Be su
installed before installing this one (using the latest versions of each is your best bet).

Release History

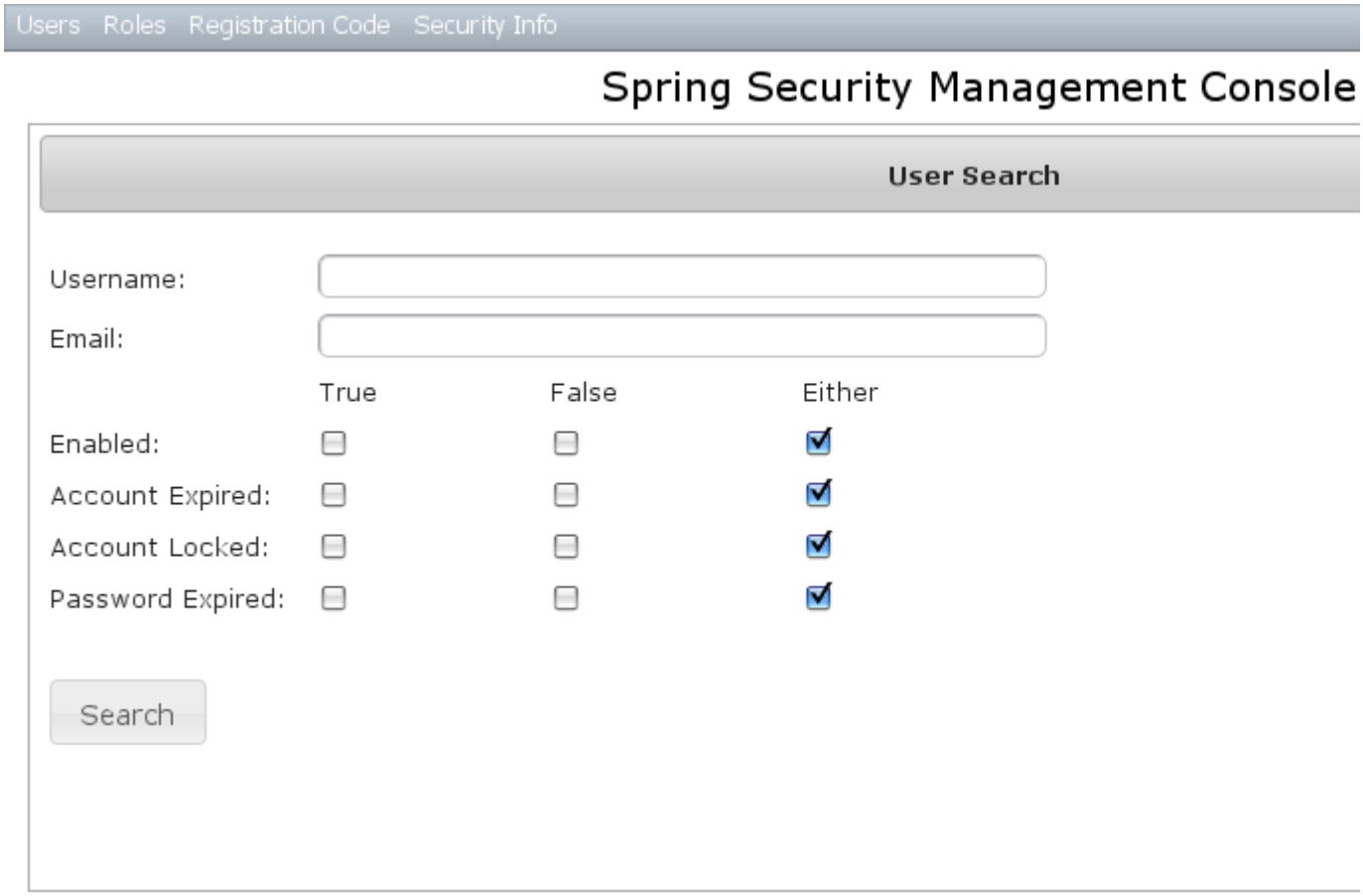
- November 11, 2013
  - 1.0-RC1 release
  - JIRA Issues
- February 12, 2012
  - 0.2 release
  - JIRA Issues
- September 14, 2010
  - 0.1.2 release
  - JIRA Issues
- July 27, 2010
  - 0.1.1 release
- July 26, 2010
  - initial 0.1 release

2 User Management

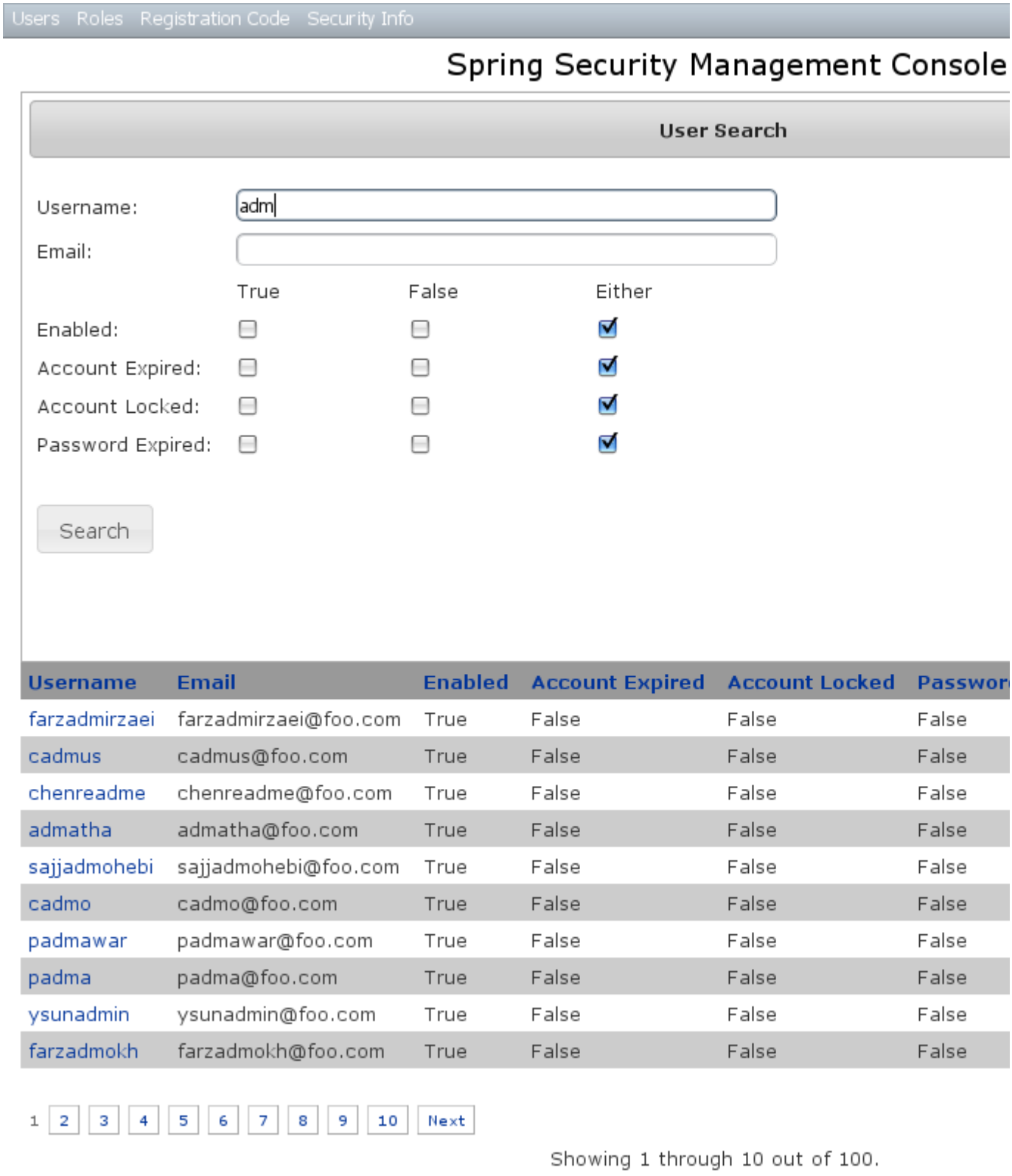
User search

The default action for the User controller is search. By default only the standard fields (`usernam` `passwordExpired`) are available but this is customizable with the `grails s2ui-override` script

You can search by any combination of fields, and the `username` field has an Ajax autocomplete to assis field has been added to the domain class and UI. Leave all fields empty and all checkboxes set at Either to

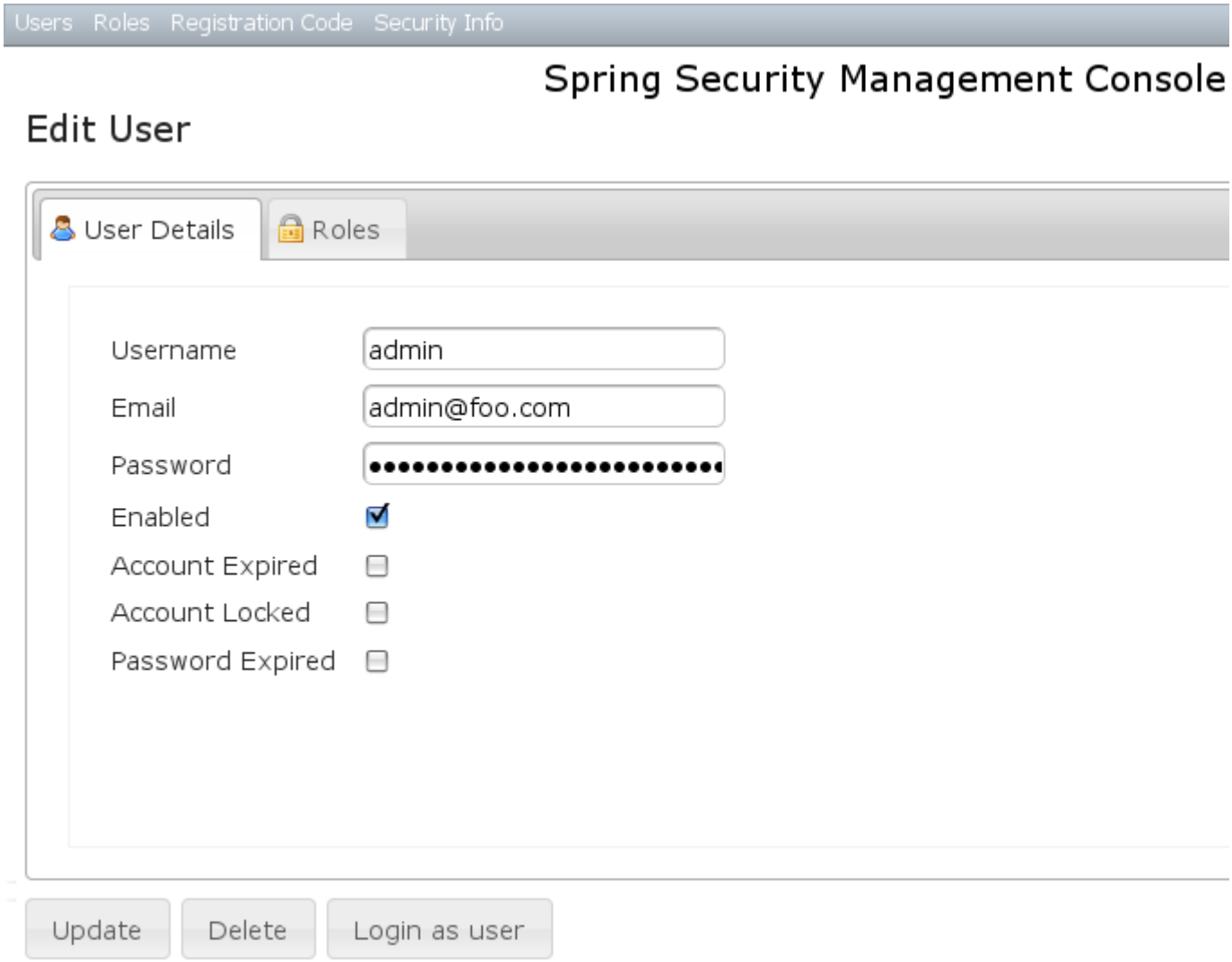


This example shows a search for usernames containing 'adm' (the search is case-insensitive and the search paginated in groups of 10. All of the column headers are clickable and will sort the results by that field:

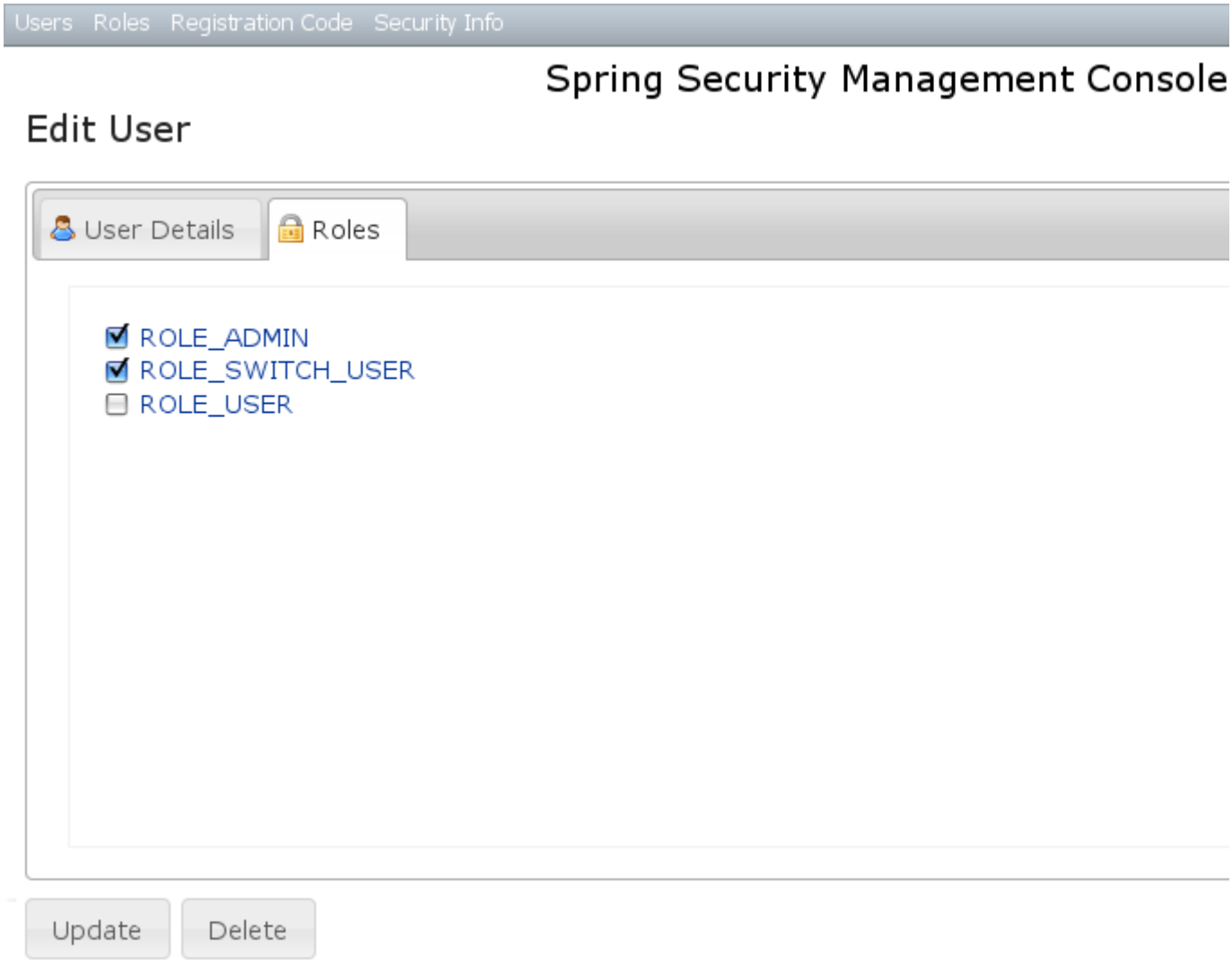


User edit

After clicking through to the 'admin' user you get to the edit page (there are no view pages):



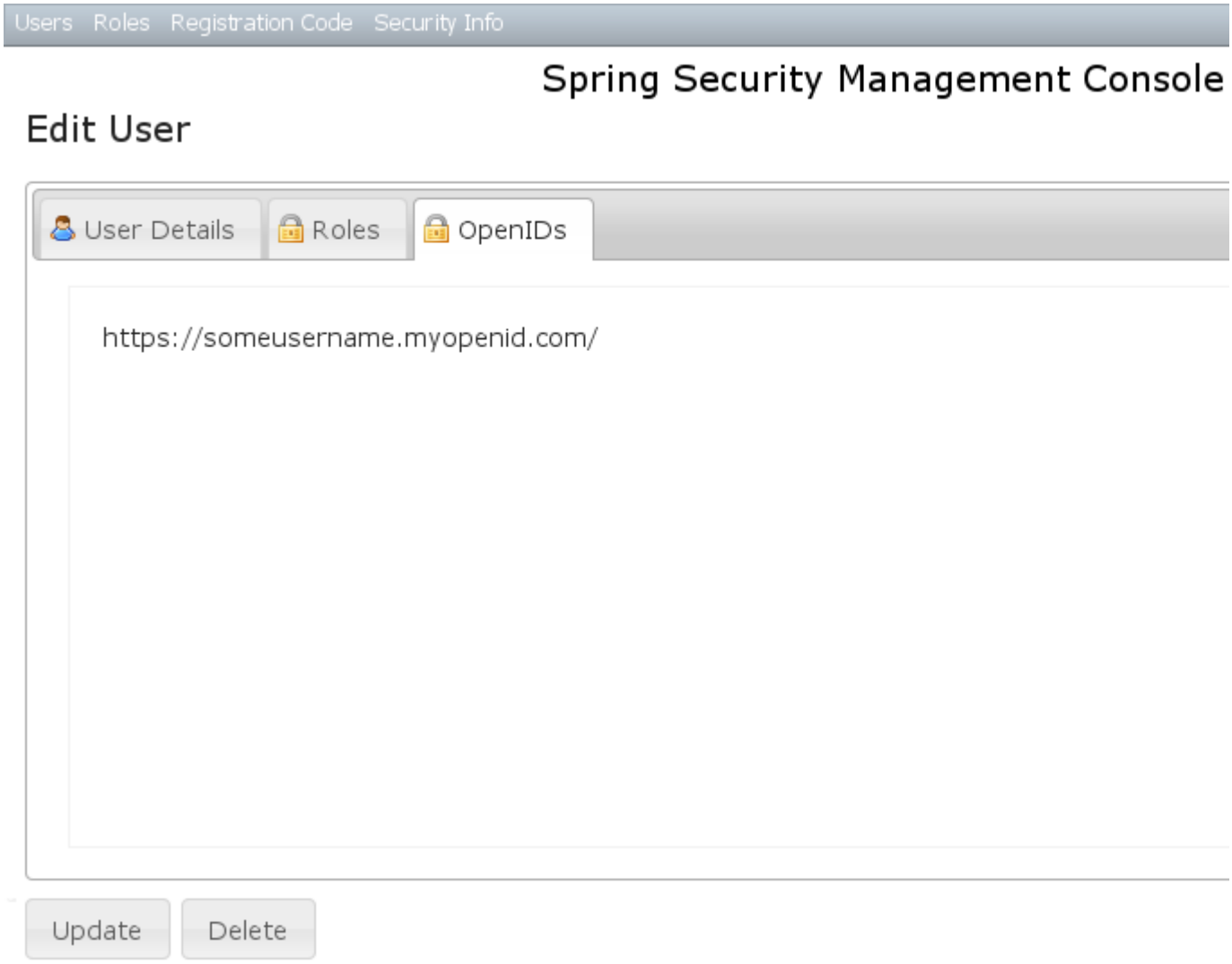
You can update any of the attributes or delete the user. You can see that there's a "Login as user" button h
is granted `ROLE_SWITCH_USER`:



This allows you to temporarily assume the identity of another user (see the Spring Security Core plugi
"Logged in as ..." information in the top right of the screen will change to show that you're running as a
arbitrary and can be changed by overriding `edit.gsp`.

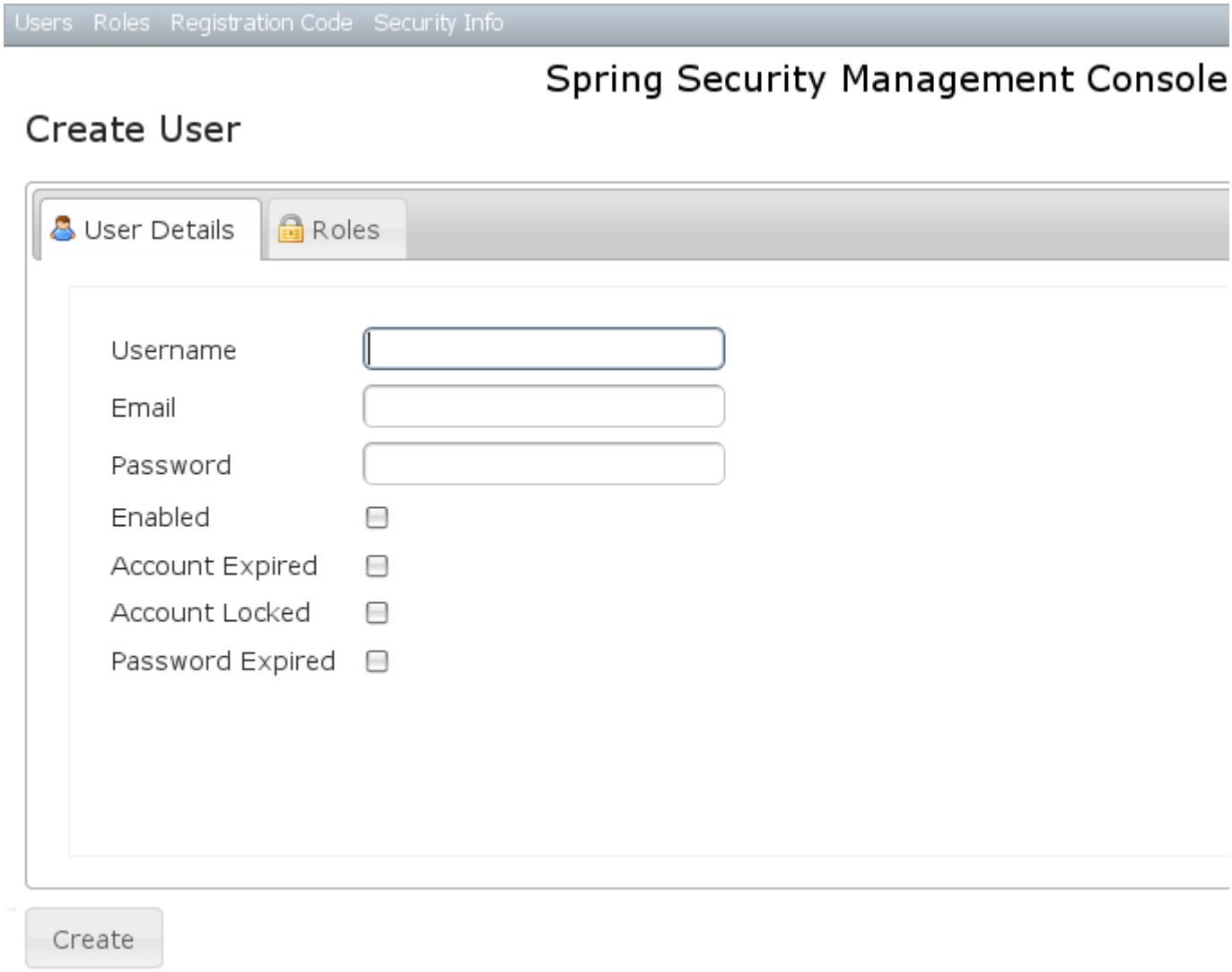
OpenIDs

If the OpenID plugin is installed, you can manage users' associated OpenIDs:



User creation

You can create new users by going to `/user/create` or by clicking the `Create` action in the `Users` n


Users Roles Registration Code Security Info
Spring Security Management Console
Create User
User Details
Roles
Username
Email
Password
Enabled
Account Expired
Account Locked
Password Expired
Create

3 Role Management

Role search

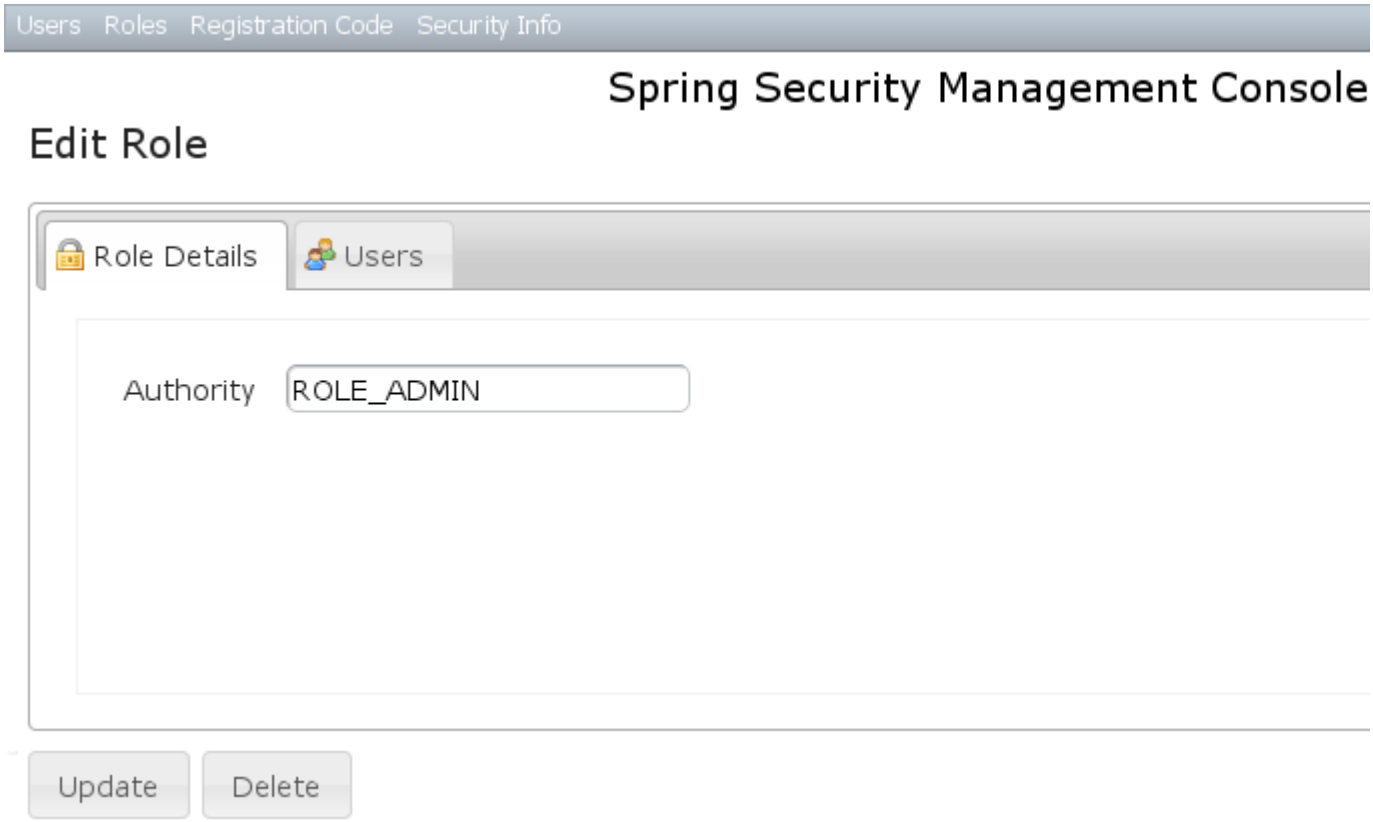
The default action for the Role controller is search. By default only the `authority` field is available
script - see the section on configuration.

The `authority` field has an Ajax autocomplete to assist in finding roles. Leave all fields empty to return

Authority search is case-insensitive and the search string can appear anywhere in the name (and you can c
groups of 10 but if there's only one result you'll be forwarded to the edit page for that role.

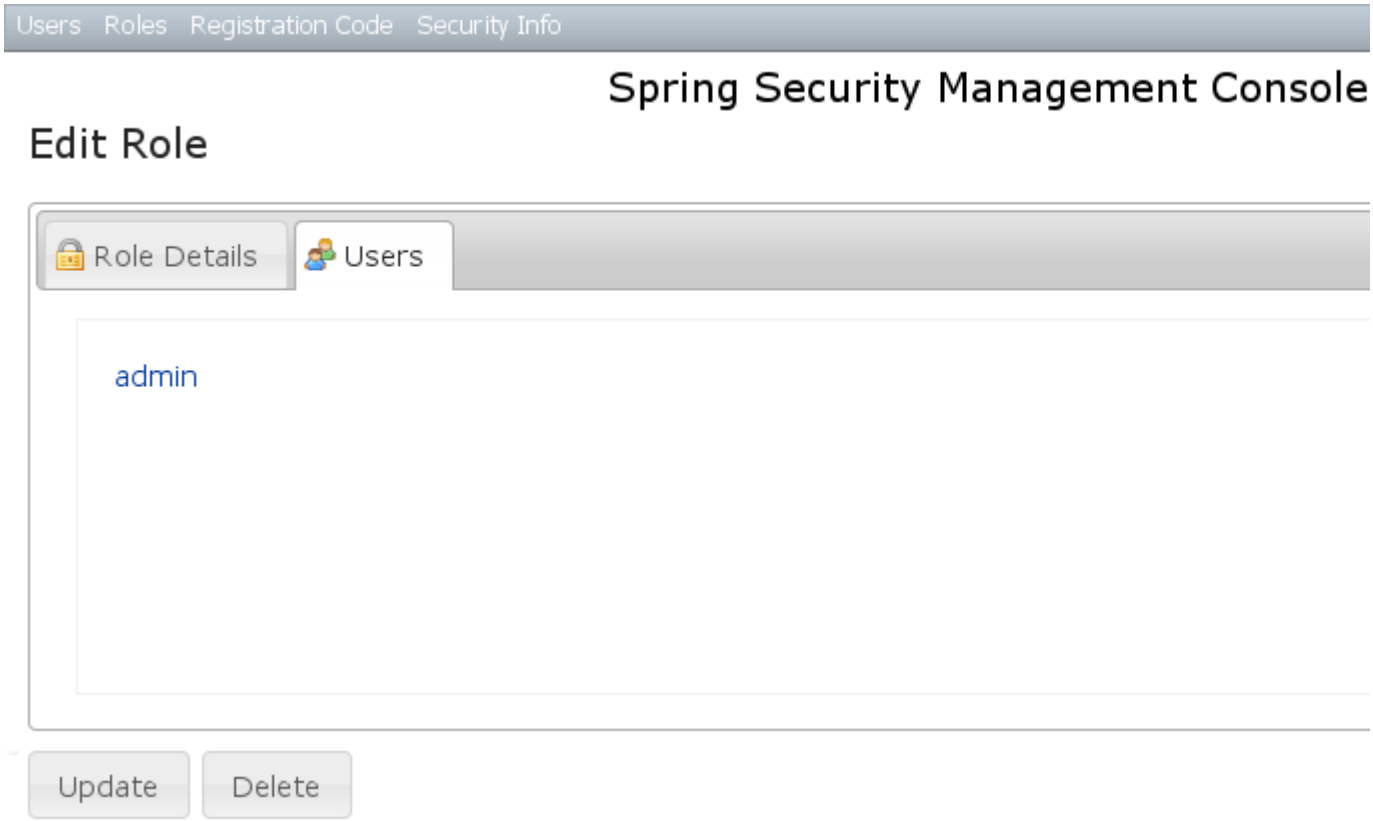
Role edit

After clicking through to a role you get to the edit page (there are no view pages):



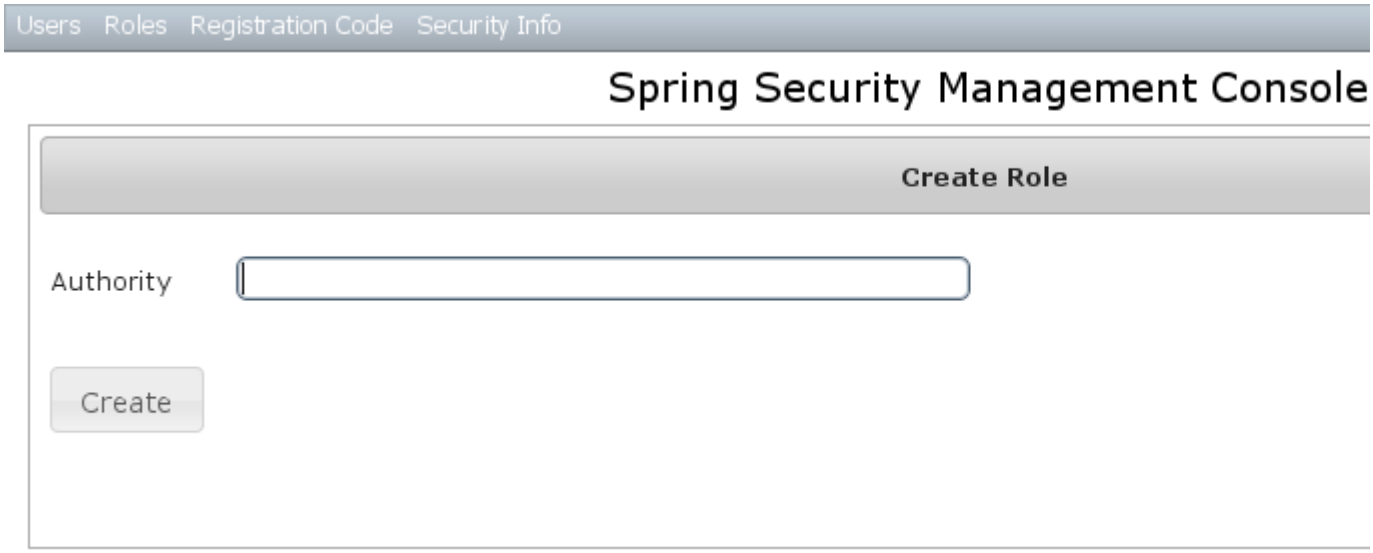
You can update any of the attributes or delete the role. Any user that had been granted the role will lose the

If you click the Users tab you can see which users have a grant for this role and can click through to their e



Role creation

You can create new roles by going to `/role/create` or by clicking the `Create` action in the `Roles` n

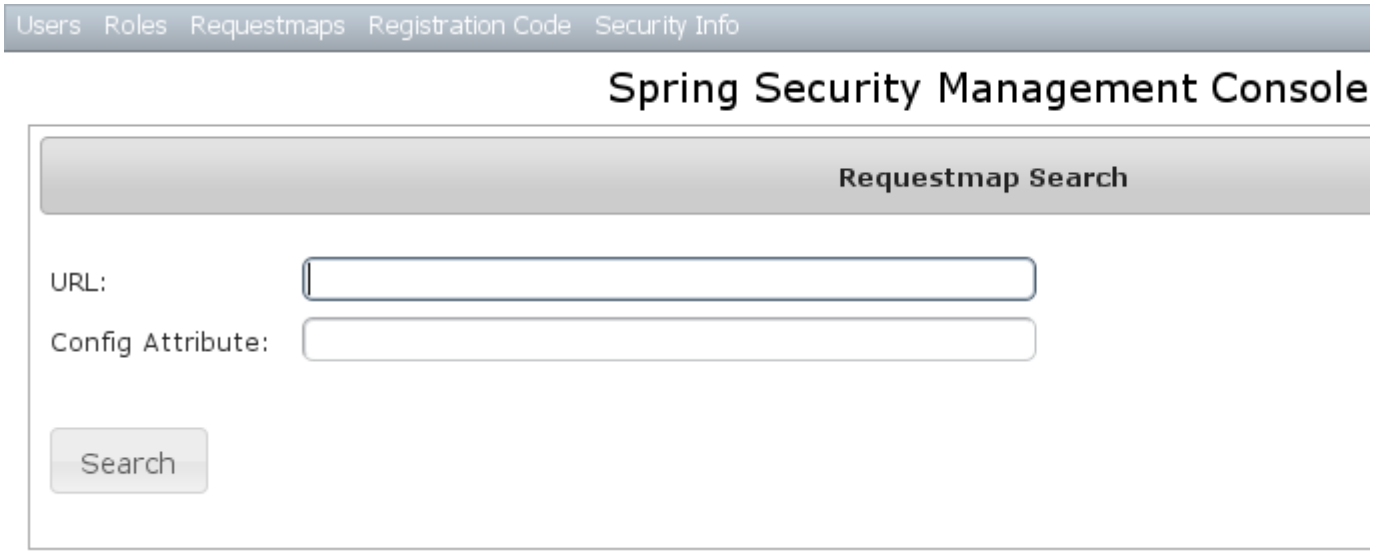

Users Roles Registration Code Security Info
Spring Security Management Console
Create Role
Authority
Create

4 Requestmap Management

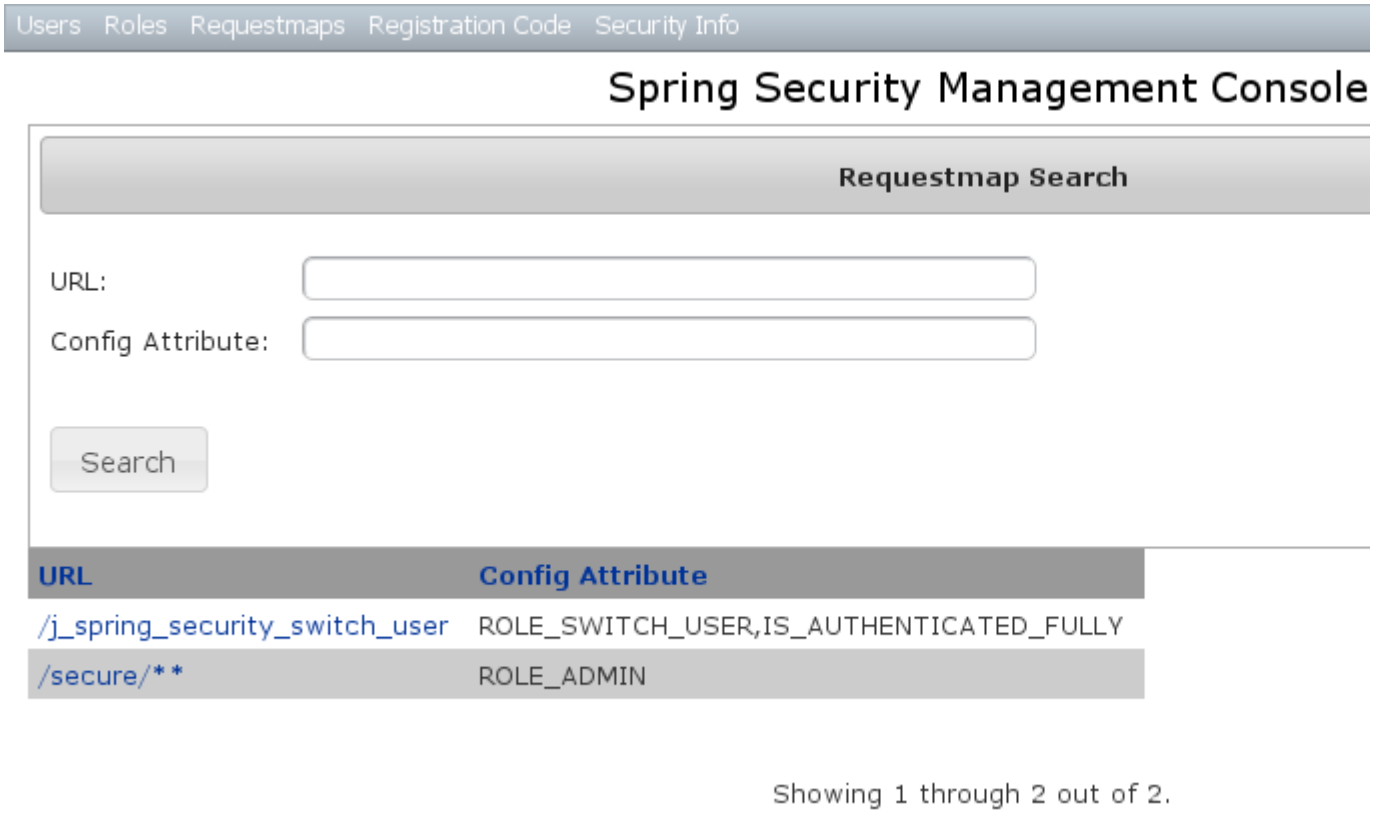
The default approach to securing URLs is with annotations, so the `grails.plugin.springsecurity.securityConfigType` has the value `"Request` `Config.groovy`.

Requestmap search

The default action for the Requestmap controller is search. You can search by URL or Config Attribute. Le

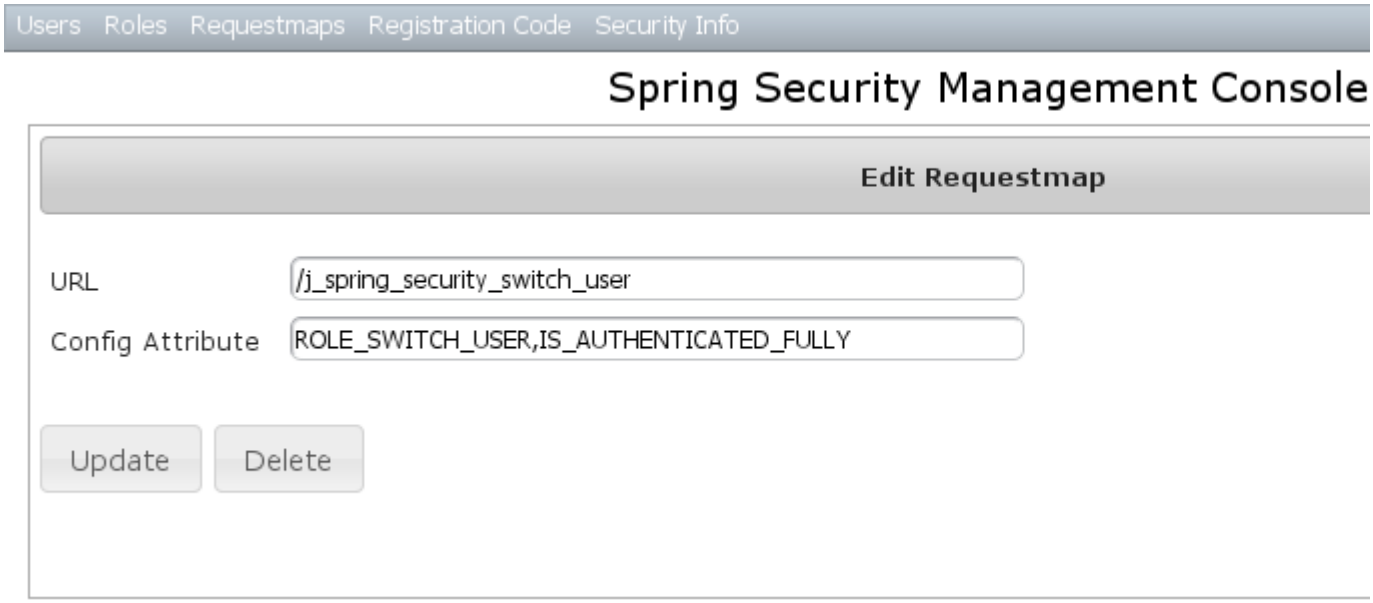


Searching is case-insensitive and the search string can appear anywhere in the field. Multiple results ar header to sort by that field:



Requestmap edit

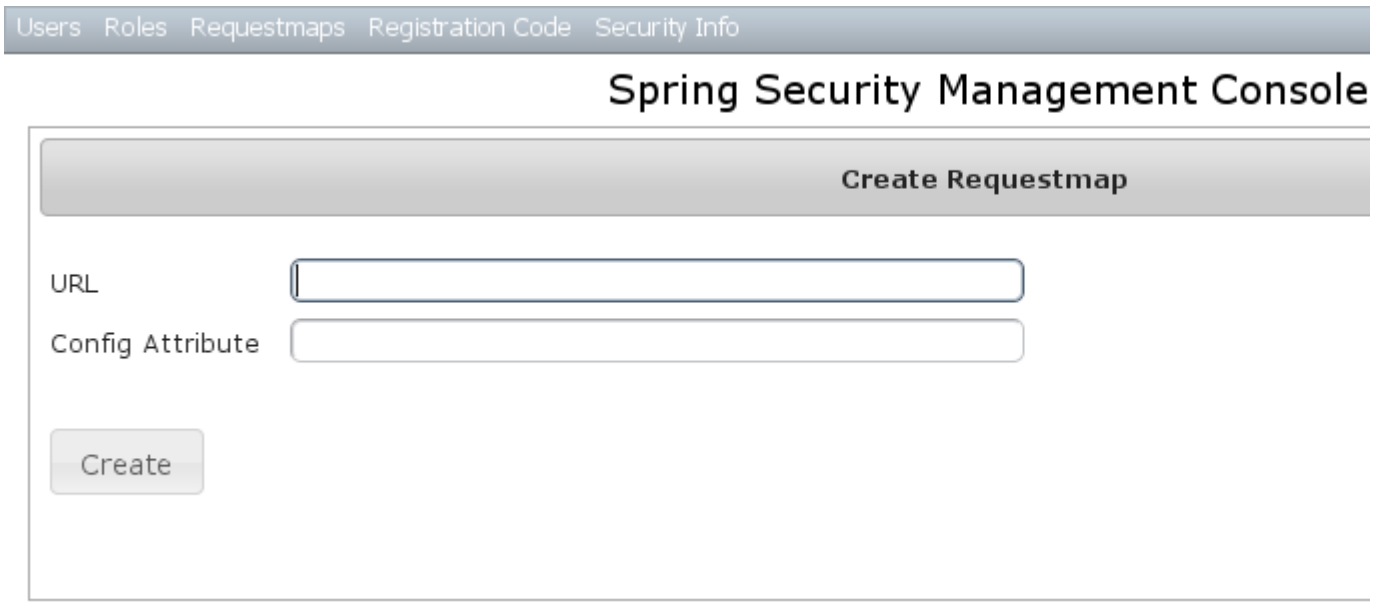
After clicking through to a Requestmap you get to the edit page (there are no view pages):



You can update any of the attributes or delete the Requestmap. Editing or deleting a Requestmap resets immediately.

Requestmap creation

You can create new Requestmaps by going to `/requestmap/create` or by clicking the `Create` actio

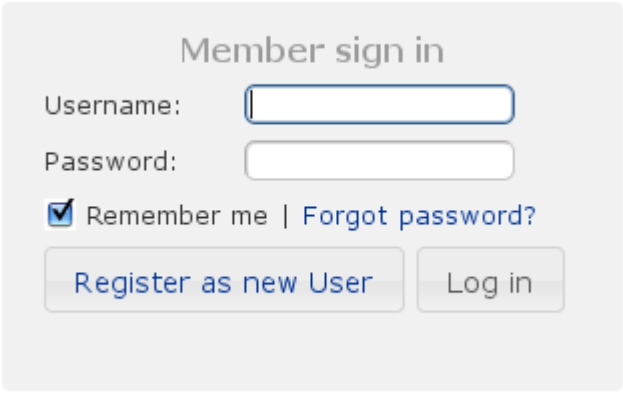


Creating a Requestmap resets the cache of loaded instances, so your changes will take effect immediately.

5 User Registration

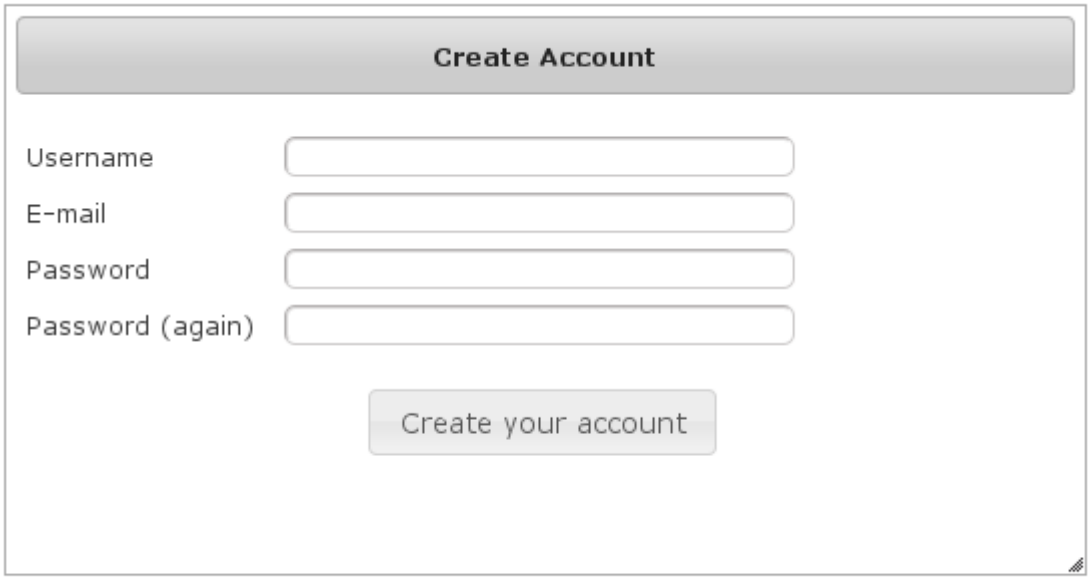
Most of the plugin's controllers are intended to be part of a backend admin application, but the Reg
user-facing. So they're not available in the admin menu like the User, Role, and other backend functionalit

One way to do this is to replace the default `login.gsp` that's provided by the Spring Security Core
`grails s2ui-override auth` - see the section on configuration for more details. If you do this you

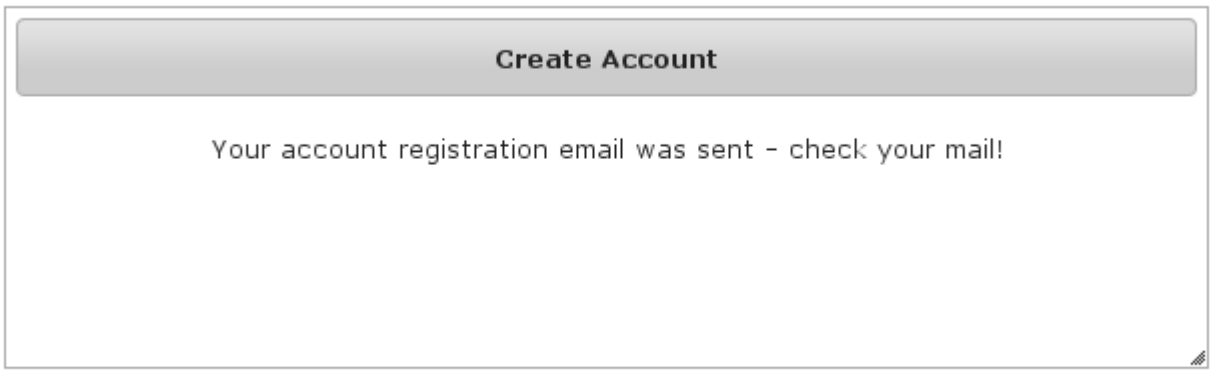


Registration

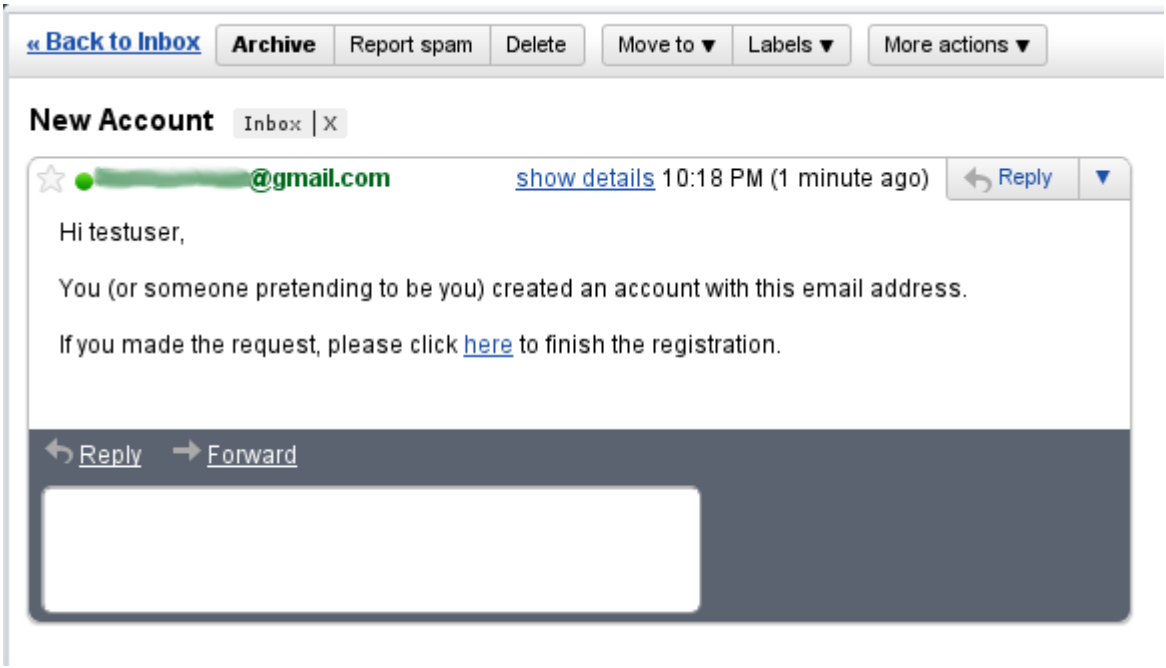
Navigate to `/register/`:



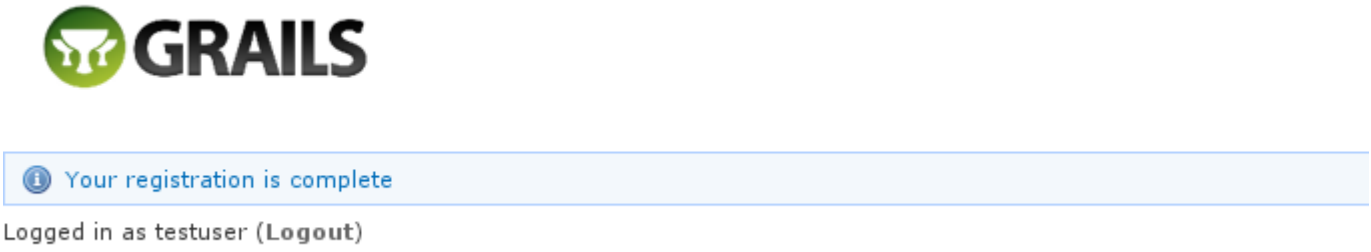
After filling out valid values an email will be sent and you'll see a success screen:



Click on the link in the email:



and you'll finalize the process, which involves enabling the locked user and pre-authenticating, then redirec



Configuration

The post-registration destination url is configurable in `grails-app/conf/Config.groovy` using th

```
grails.plugin.springsecurity.ui.register.postRegisterUrl = '/welcome'
```

If you don't specify a value then the `defaultTargetUrl` value will be used, which is `'/'` by default.

You can customize the subject, body, and from address of the registration email by overriding the de
example:

```
grails.plugin.springsecurity.ui.register.emailBody = '...'
grails.plugin.springsecurity.ui.register.emailFrom = '...'
grails.plugin.springsecurity.ui.register.emailSubject = '...'
```

The `emailBody` property should be a GString and will have the User domain class instance in scope in
signup in the `url` variable.

In addition, each new user will be granted `ROLE_USER` after finalizing the registration. If you want to
example if you want an admin to approve new users and explicitly enable new users) then you can custo
List of Strings):

```
grails.plugin.springsecurity.ui.register.defaultRoleNames = [] // no roles
```

or

```
grails.plugin.springsecurity.ui.register.defaultRoleNames = ['ROLE_CUSTOMER']
```

Mail configuration

The plugin uses the Mail plugin to send registration emails, so you'll need to configure an SMTP server. Se

Notes

You should consider the registration code as starter code - every signup workflow will be different, and th
may wish to collect more information than just username and email - first and last name for example
registration controller and GSPs into your application to be customized.

If there are unexpected validation errors during registration (which can happen when there is a
`RegisterController` they will be logged at the `warn` level, so enable logging to ensure that you see

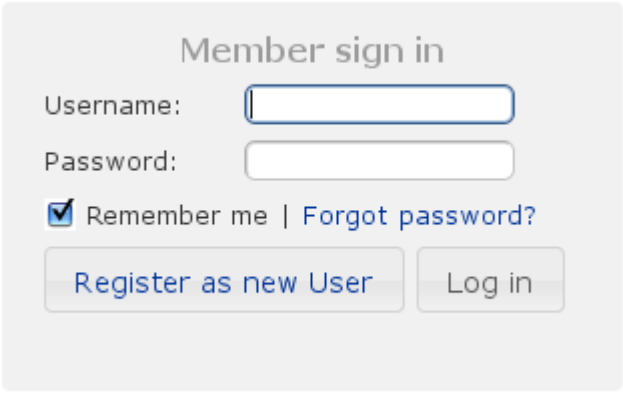
```
log4j = {
   error 'org.codehaus.groovy.grails',
         'org.springframework',
         'org.hibernate',
         'net.sf.ehcache.hibernate'
   warn 'grails.app.services.grails.plugin.springsecurity.ui.SpringSecurityUiServ
}
```

`RegisterController` and its GSPs assume that your User domain class has an `email` fi
(using the `s2ui-override` script) if you don't need an email confirmation step or add an er

6 Forgot Password

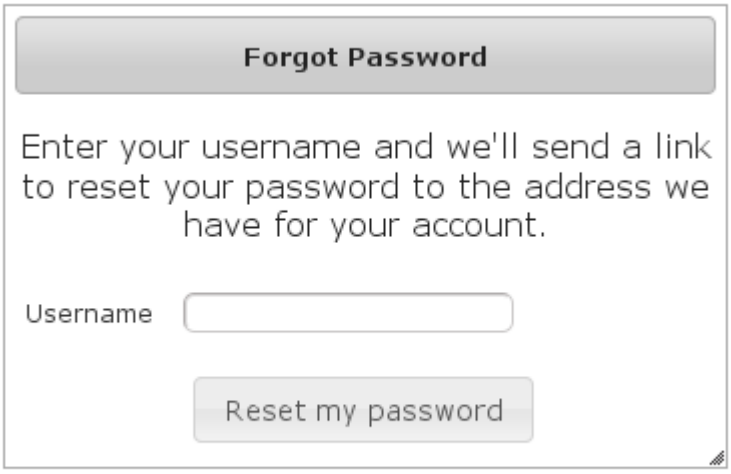
Like the Registration workflow, the Forgot Password workflow is expected to be user-facing. So it's n backend functionality - you'll need to expose them to your users.

One way to do this is to replace the default `login.gsp` that's provided by the Spring Security Core `grails s2ui-override auth` - see the section on configuration for more details. If you do this you

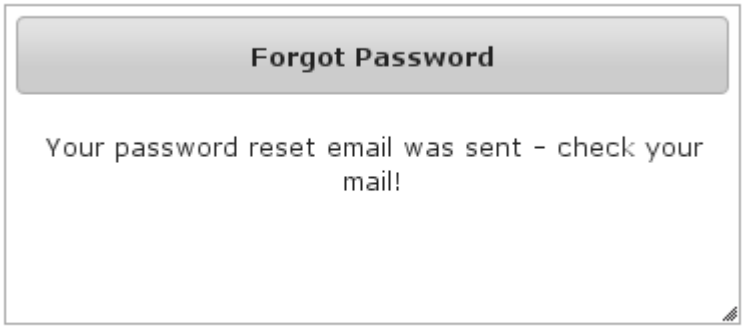


Forgot Password

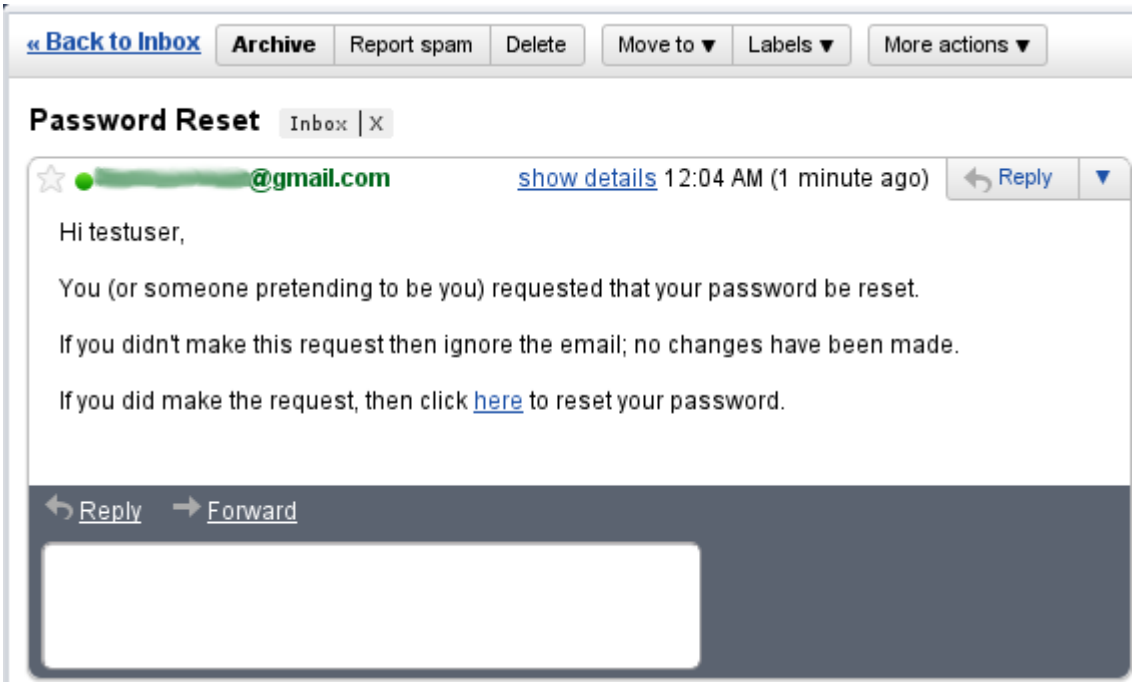
Navigate to `/register/forgotPassword`:



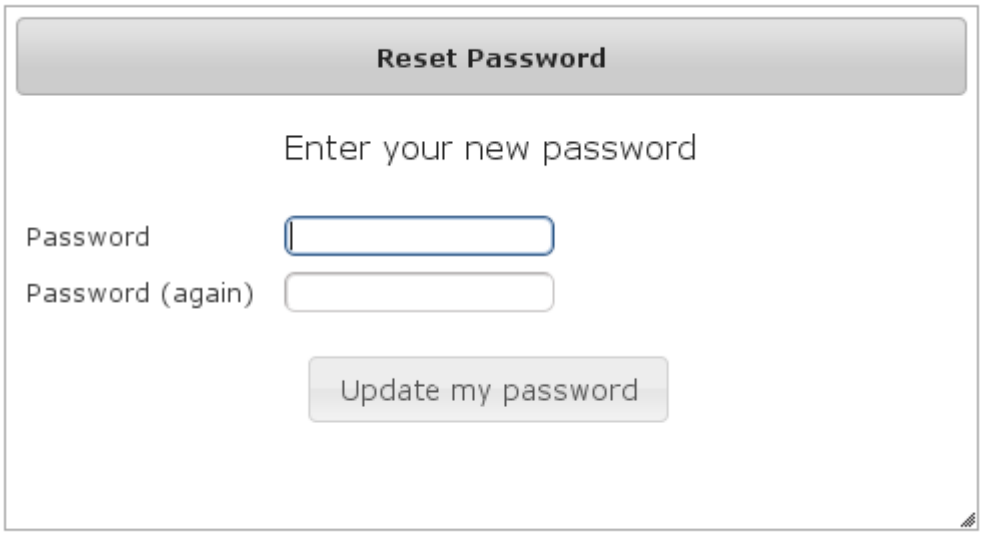
After entering a valid username an email will be sent and you'll see a success screen:



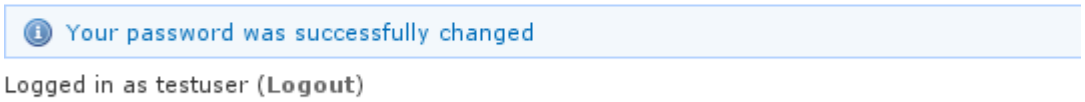
Click on the link in the email:



and you'll open the reset password form:



After entering a valid password you'll finalize the process, which involves storing the new password encry
the configured destination:



Configuration

The post-reset destination url is configurable in `grails-app/conf/Config.groovy` using the `pos`

```
grails.plugin.springsecurity.ui.register.postResetUrl = '/reset'
```

If you don't specify a value then the `defaultTargetUrl` value will be used, which is `'/'` by default.

You can customize the subject, body, and from address of the reset email by overriding the default values i

```
grails.plugin.springsecurity.ui.forgotPassword.emailBody = '...'
grails.plugin.springsecurity.ui.forgotPassword.emailFrom = '...'
grails.plugin.springsecurity.ui.forgotPassword.emailSubject = '...'
```

The `emailBody` property should be a GString and will have the User domain class instance in scope i
password in the `url` variable.

Mail configuration

The plugin uses the Mail plugin to send reset emails, so you'll need to configure an SMTP server. See the p

Notes

Like the registration code, consider this workflow as starter code. Run `grails s2ui-override re`
application to be customized.

> ⚠ `RegisterController` and its GSPs assume that your User domain class has an `email` fi

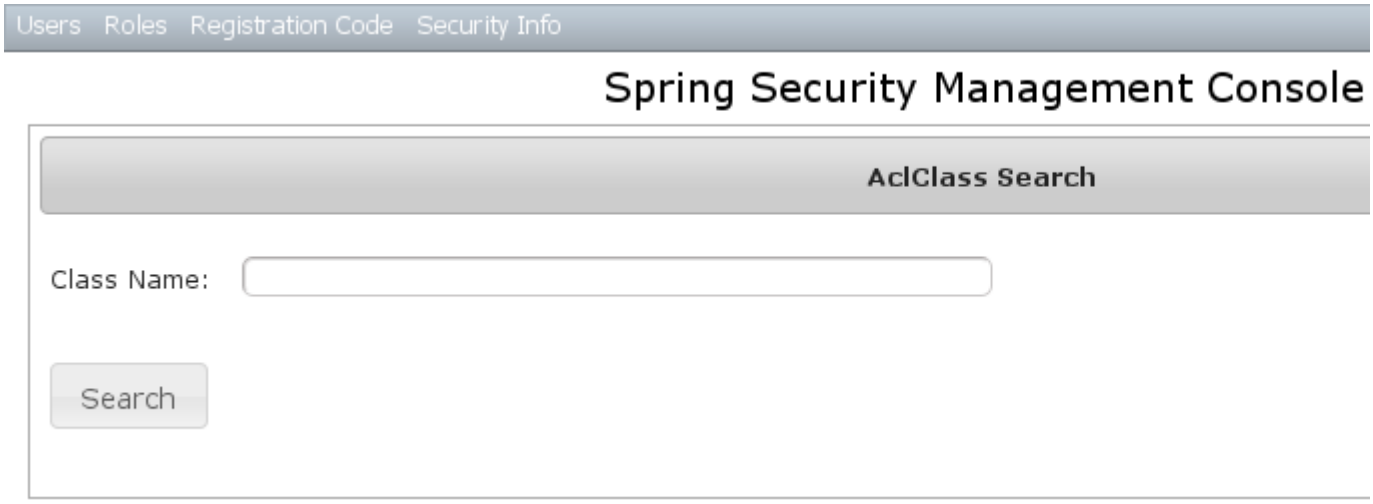
7 ACL Management

ACL management should be done using the API exposed by `AclService` and `AclUtilServic` high-level approach to managing ACLs, ACEs, etc. The functionality in this plugin is to provide a CRUD i

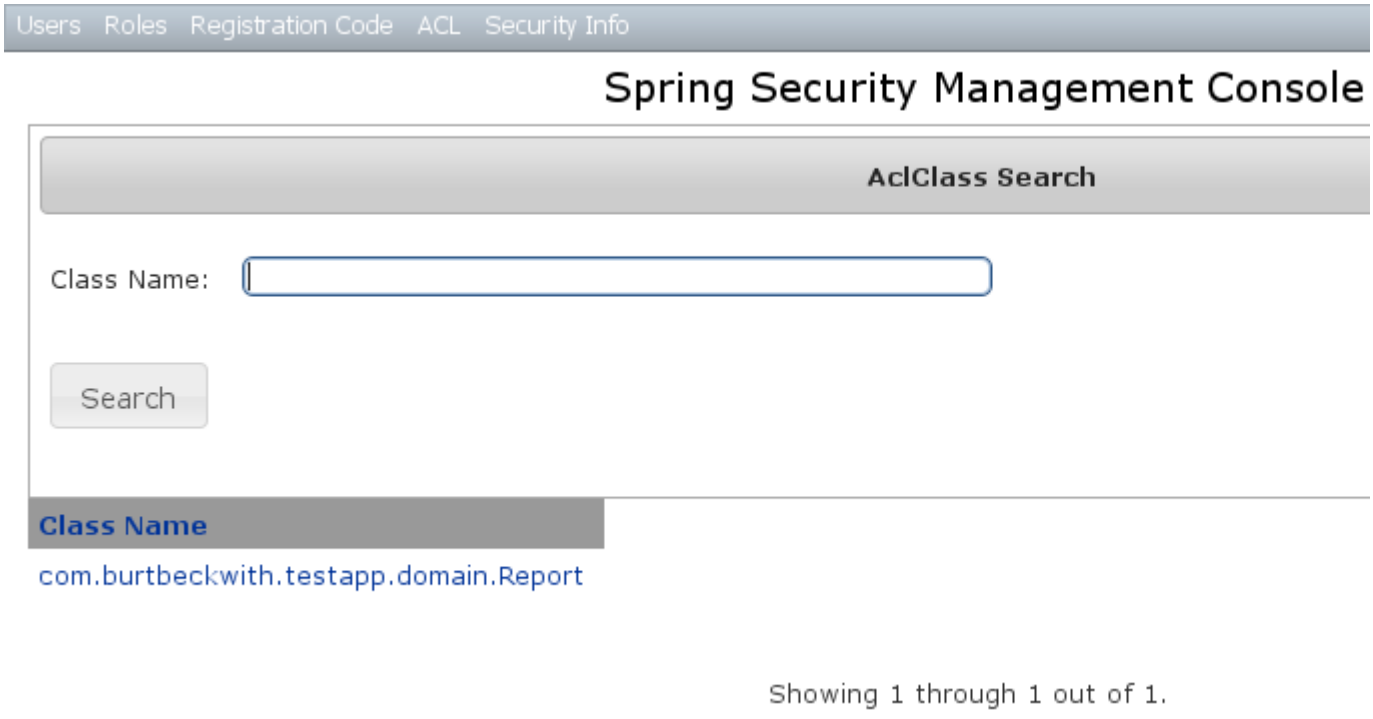
The ACL menu is only available if the ACL plugin is installed.

7.1 AclClass Management

The default action for the AclClass controller is search. The `className` field has an Ajax autocomplete to return all instances.

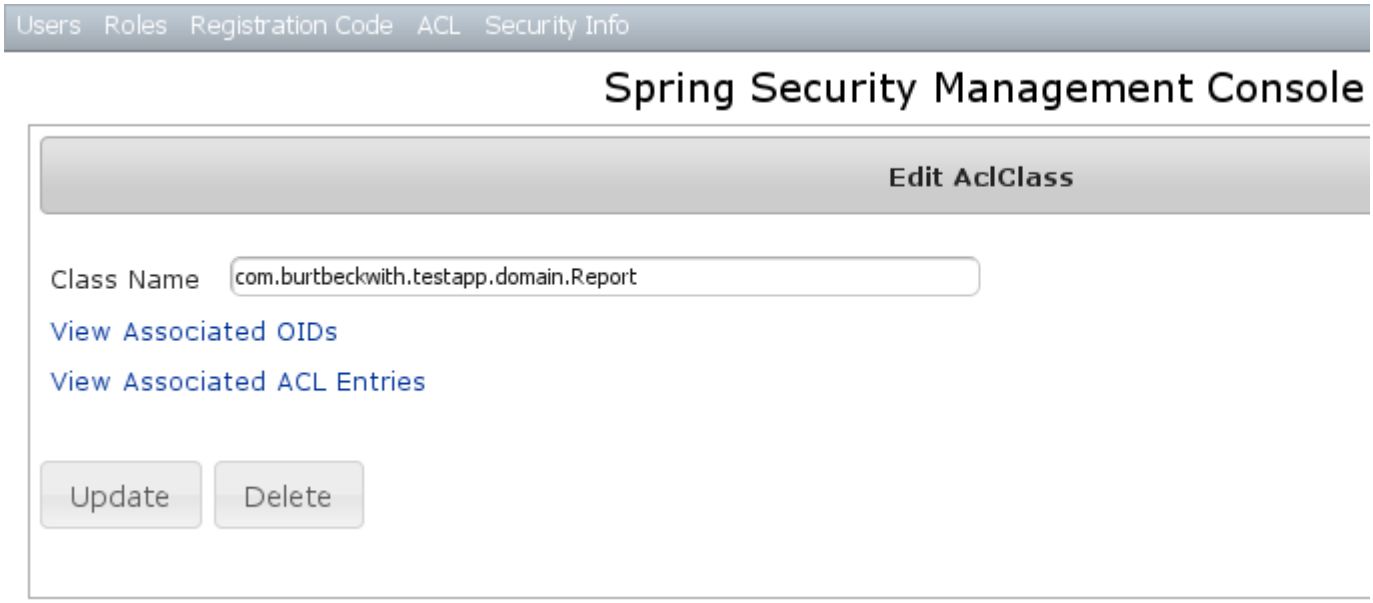


Results are shown paginated in groups of 10. The class name column header is clickable and will sort the r



AclClass Edit

After clicking through an instance you get to the edit page (there are no view pages):

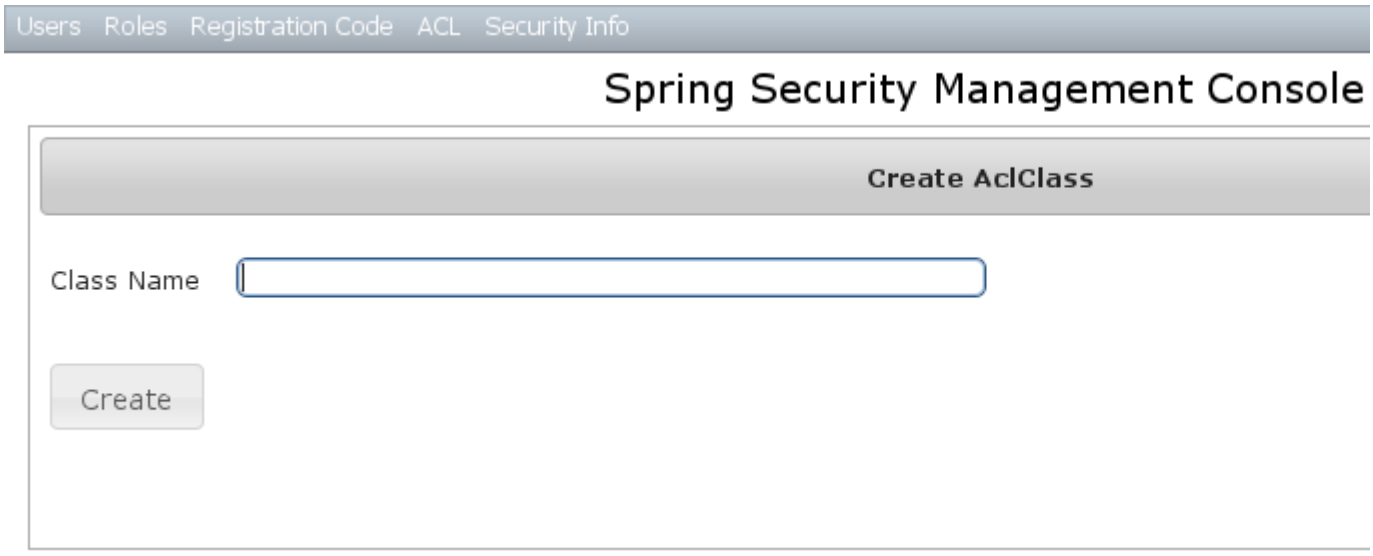


You can update the name, and delete the instance if there aren't any associated `AclObjectIdentit` cascading.

You can also see the associated `AclObjectIdentity` instances (OIDs) or `AclEntry` instances.

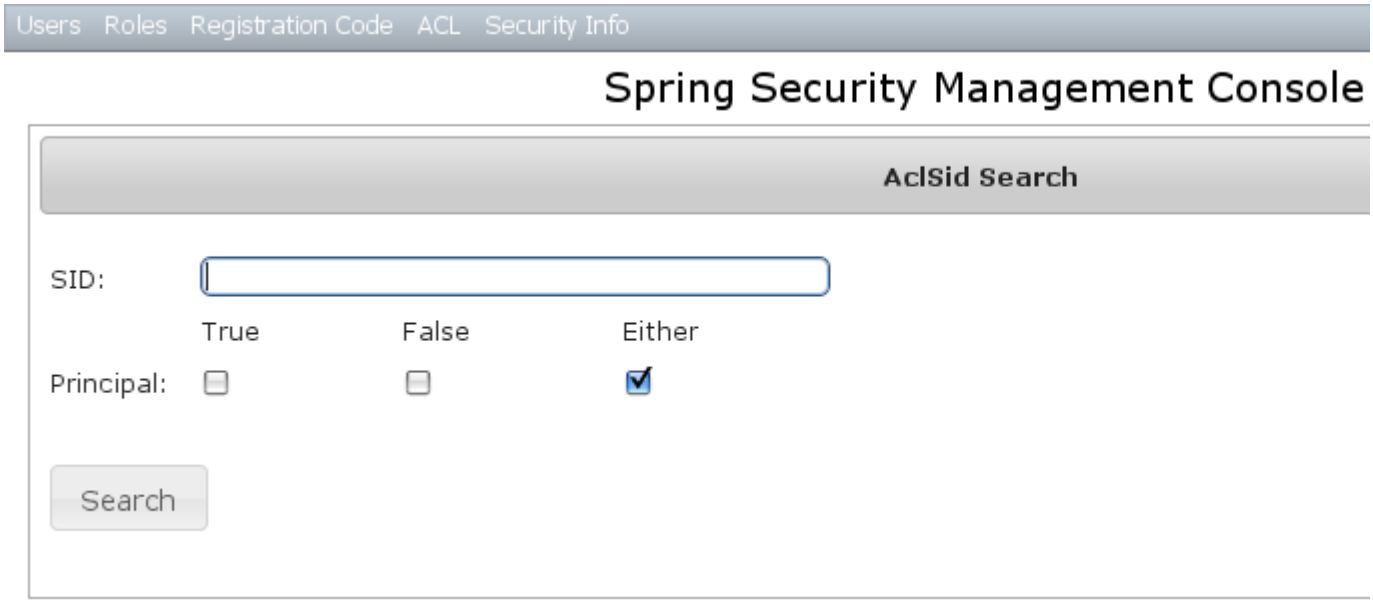
AclClass Create

You can create new instances by going to `/aclClass/create` or by clicking the `Create` action in the

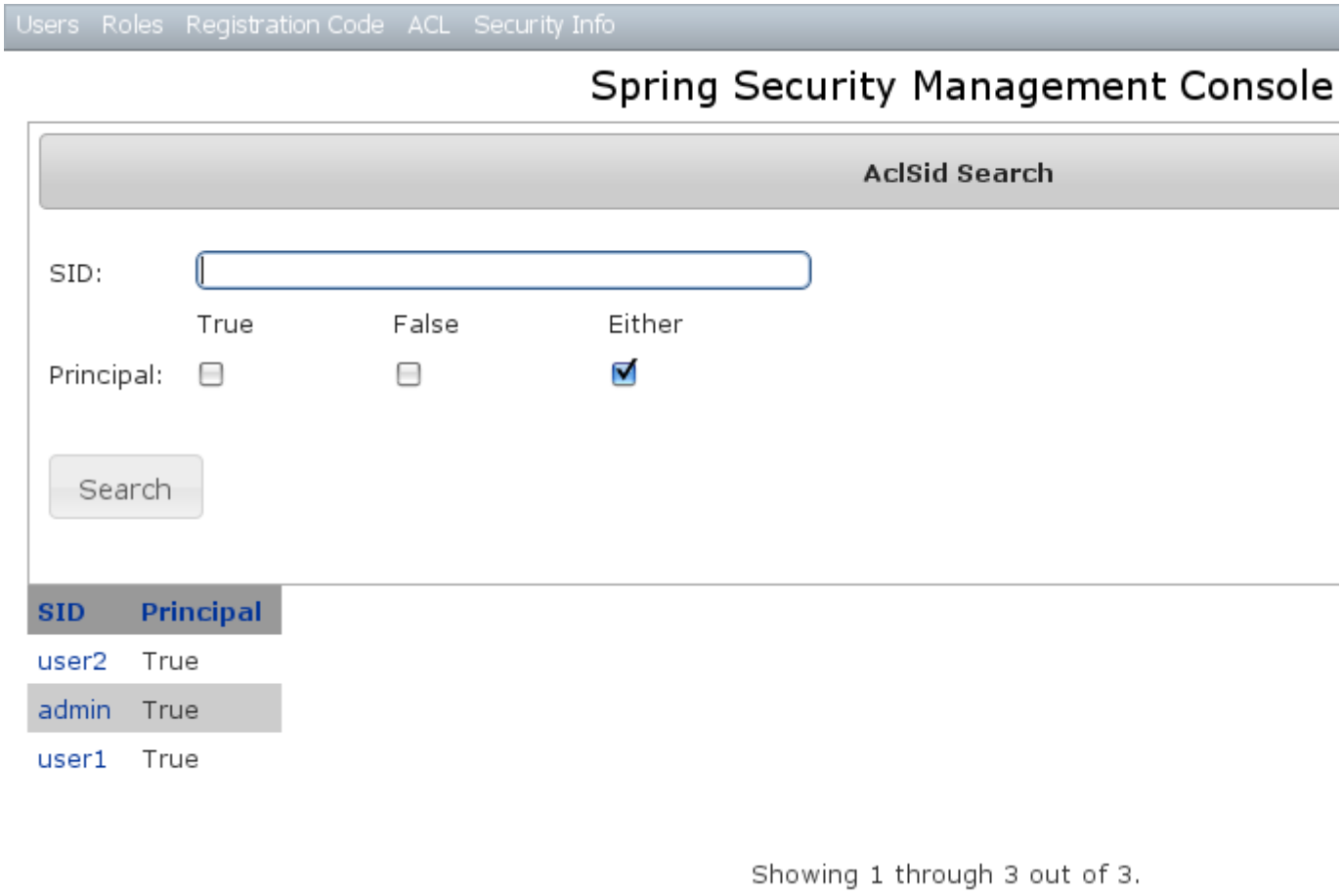


7.2 AclSid Management

The default action for the AclSid controller is search. The `sid` field has an Ajax autocomplete to assist i set to Either to return all instances.

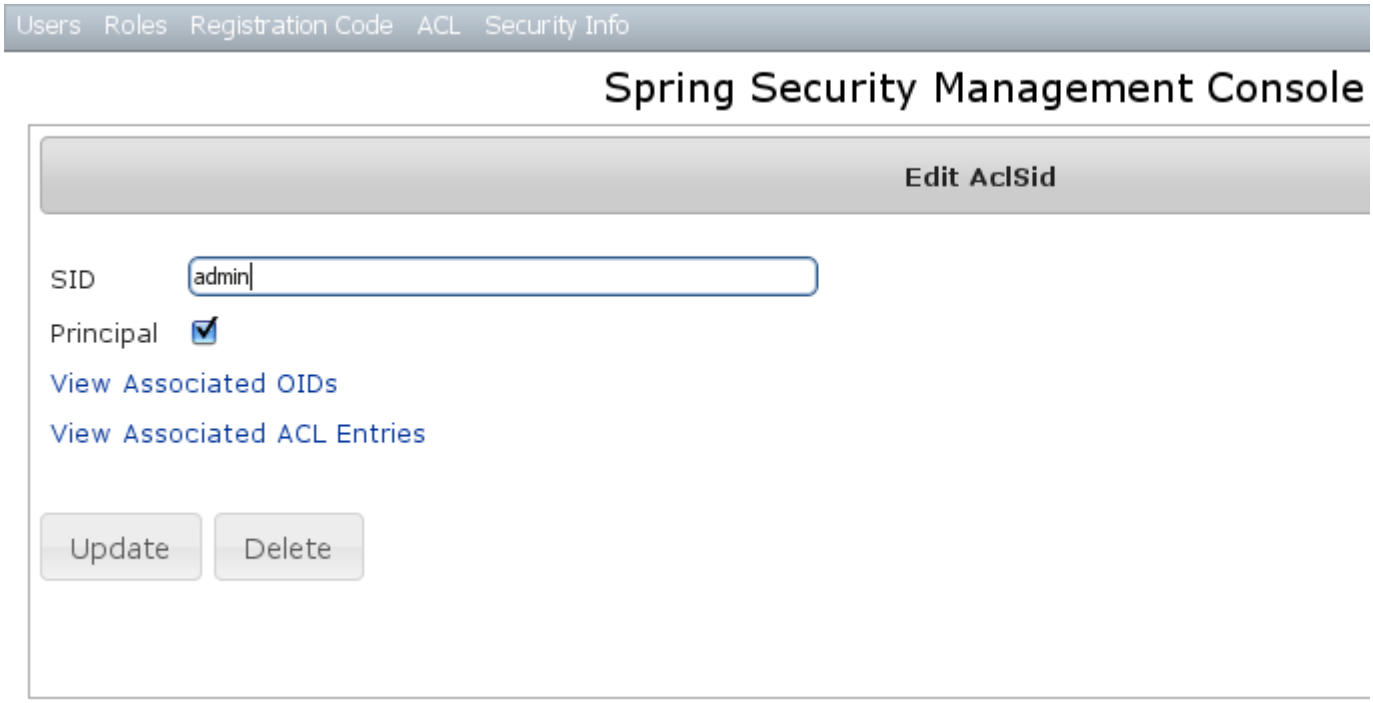


Results are shown paginated in groups of 10. The column headers are clickable and will sort the results by



AclSid Edit

After clicking through to a sid you get to the edit page (there are no view pages):

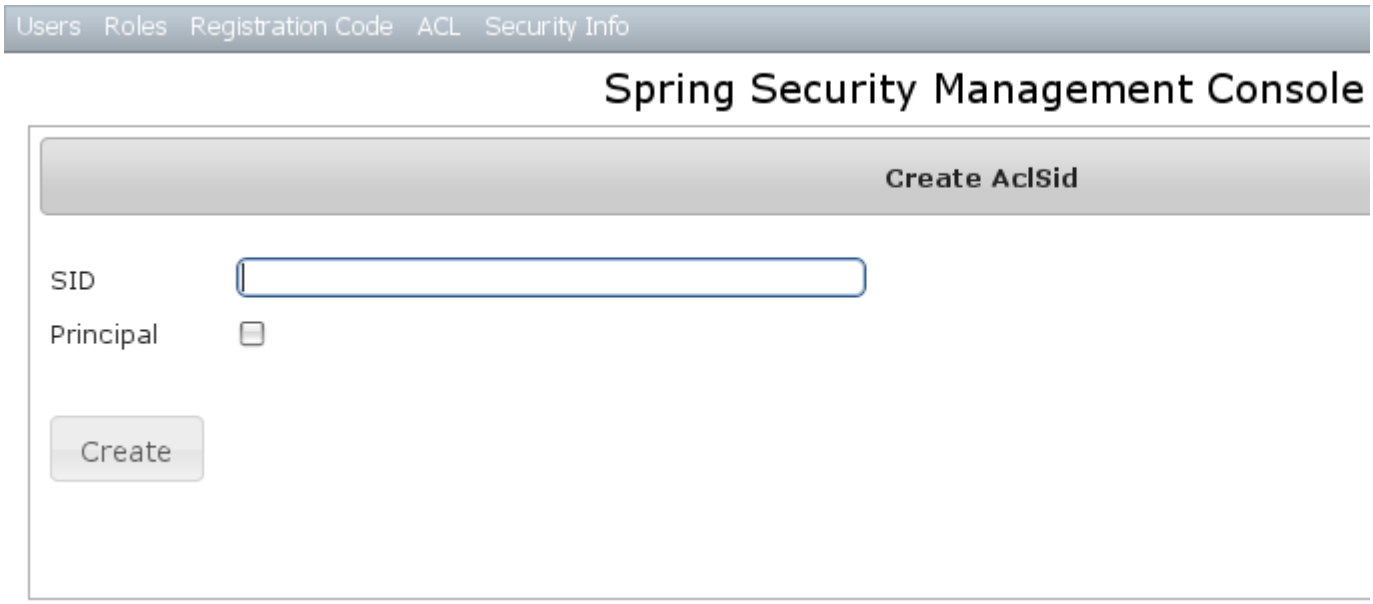


You can update the name and whether it's a Principal sid or a Role sid, and delete the instance if there instances - by default there is no support for cascading.

You can also see the associated `AclObjectIdentity` instances (OIDs) or `AclEntry` instances.

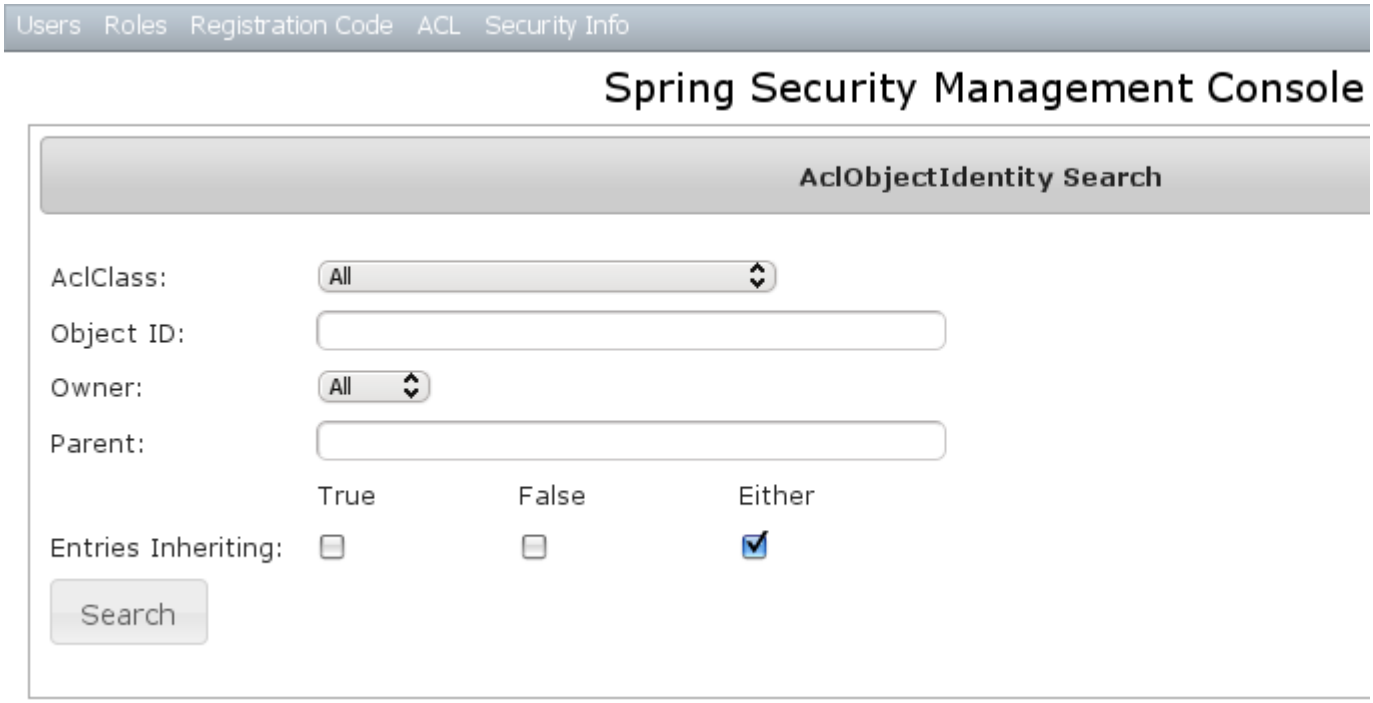
AclSid Create

You can create new instances by going to `/aclSid/create` or by clicking the `Create` action in the S

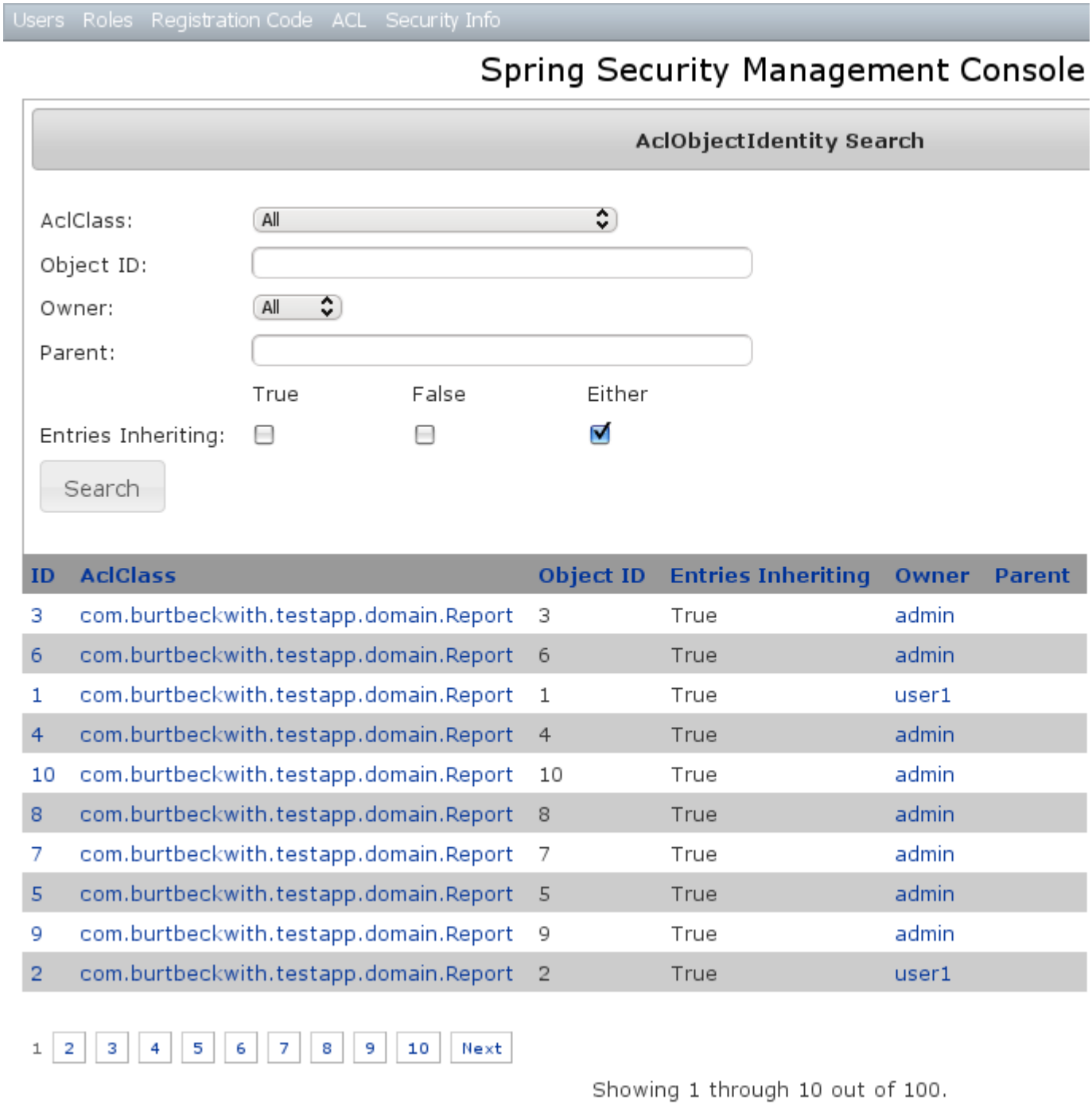


7.3 AclObjectIdentity Management

The default action for the AclObjectIdentity controller is search. Leave all fields at their default values to r

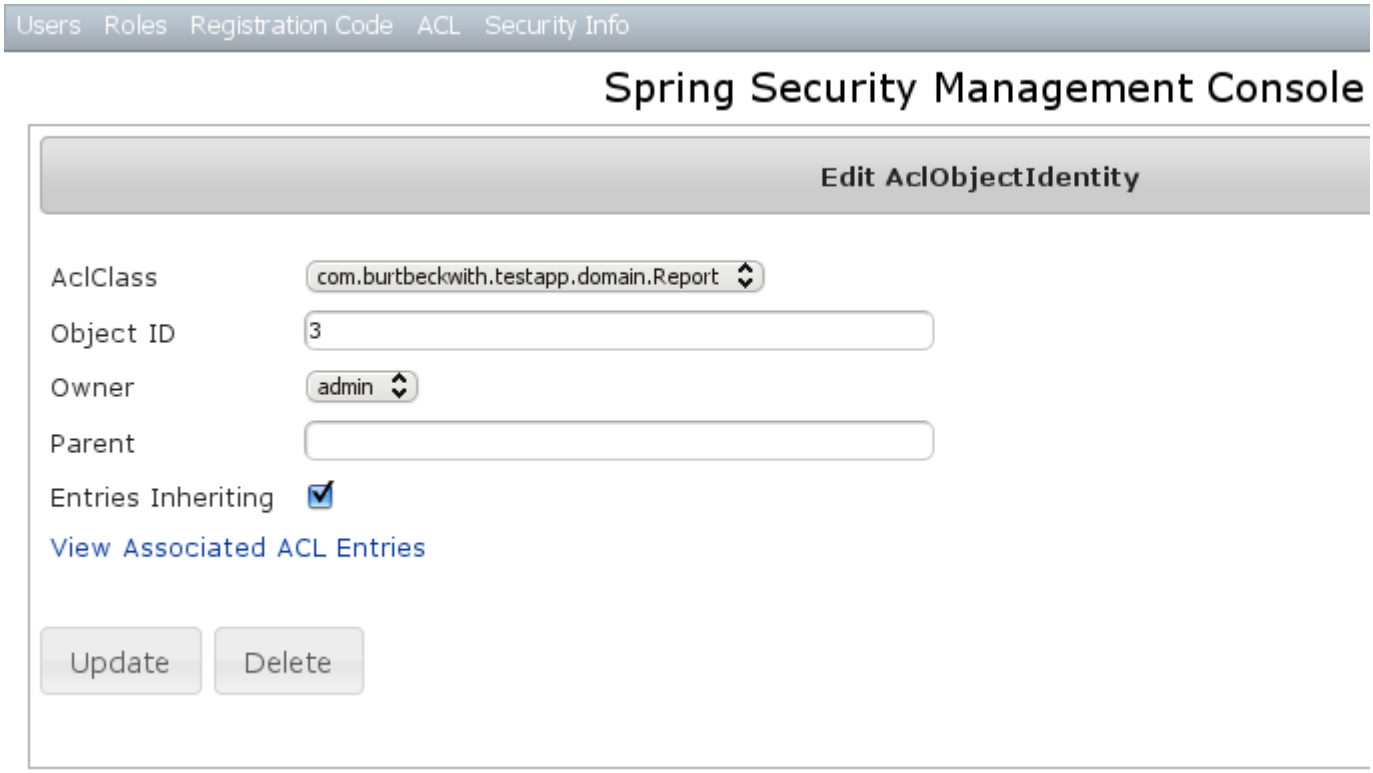


Results are shown paginated in groups of 10. The column headers are clickable and will sort the results by



AclObjectIdentity Edit

After clicking through to an instance you get to the edit page (there are no view pages):

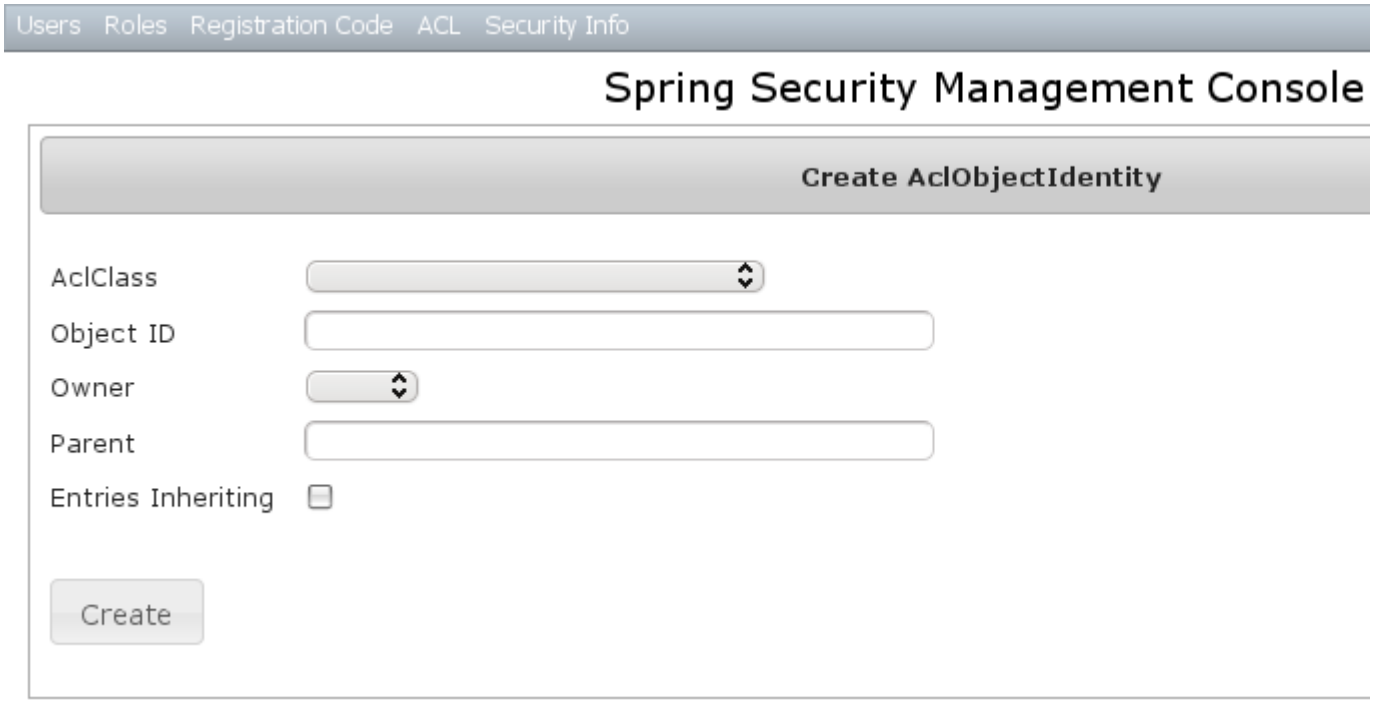


You can update the values and delete the instance if there aren't any associated `AclEntry` instances - by

You can also see the associated `AclEntry` instances.

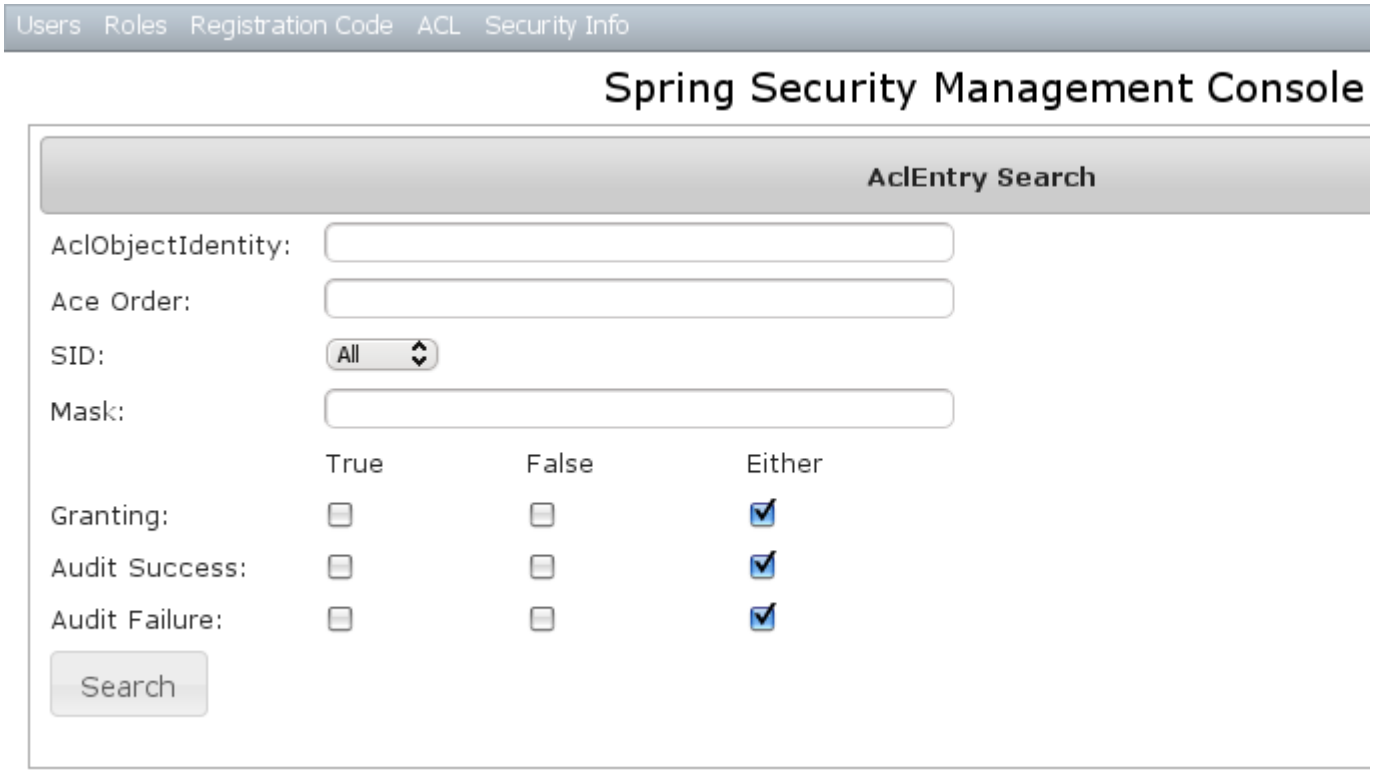
AclObjectIdentity Create

You can create new instances by going to `/aclObjectIdentity/create` or by clicking the `Create`

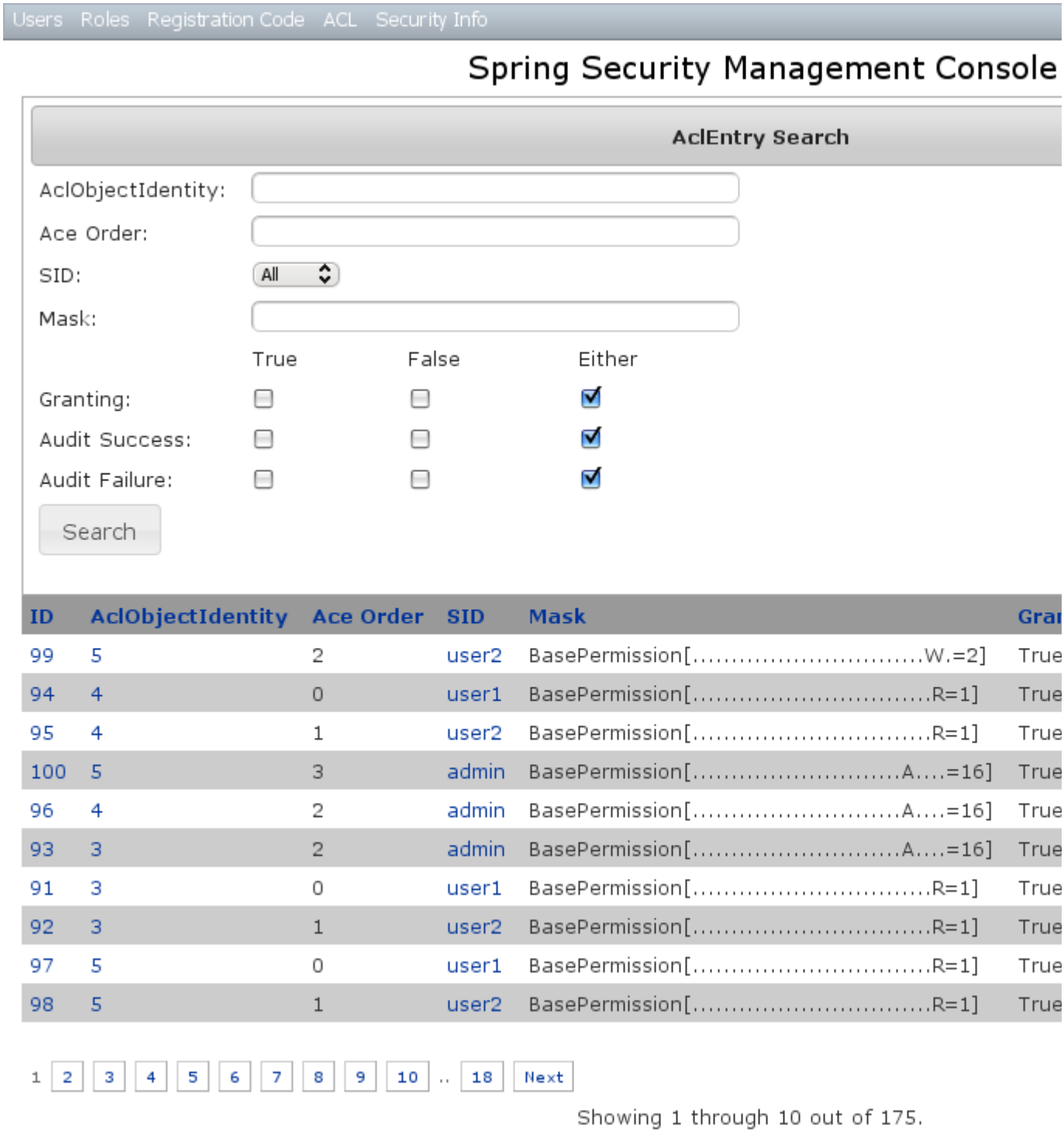


7.4 AclEntry Management

The default action for the AclEntry controller is search. Leave all fields at their default values to return all

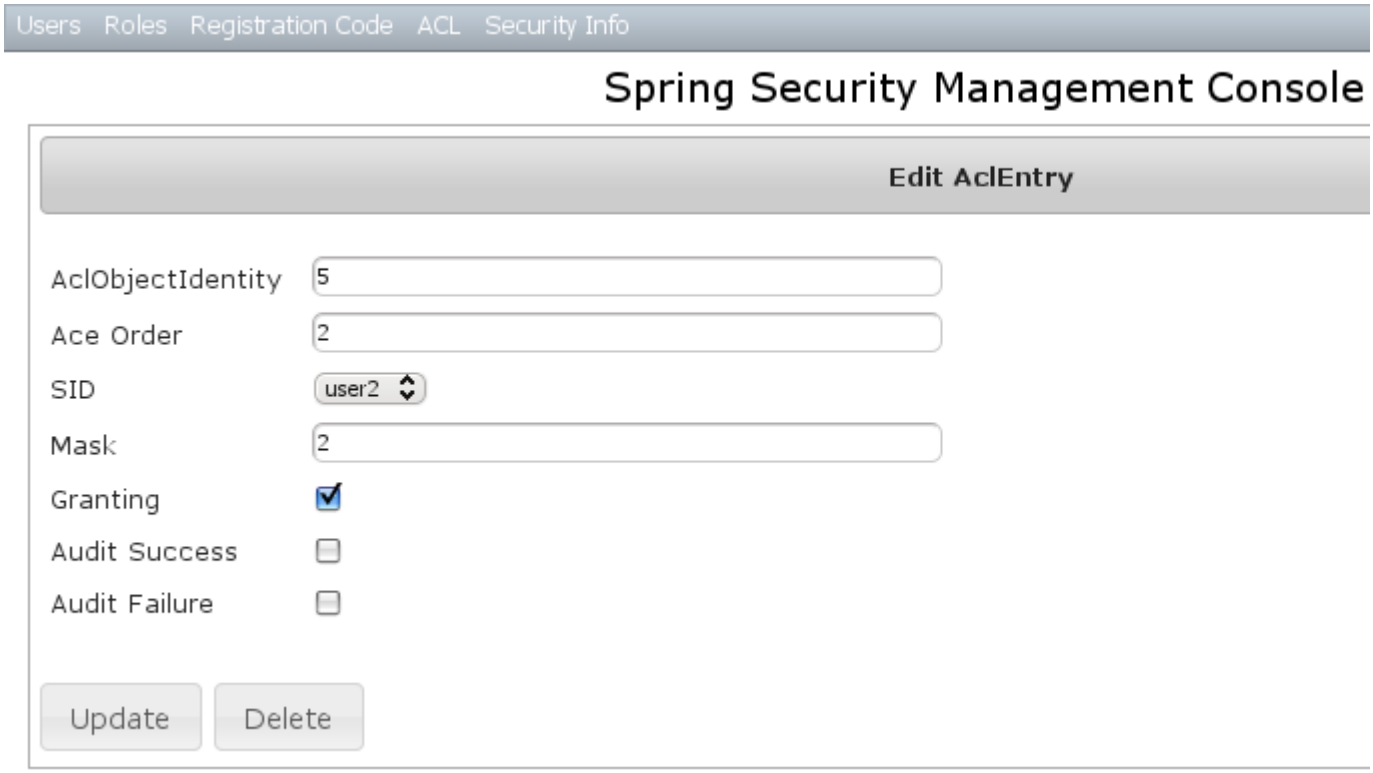


Results are shown paginated in groups of 10. The column headers are clickable and will sort the results by



AclEntry Edit

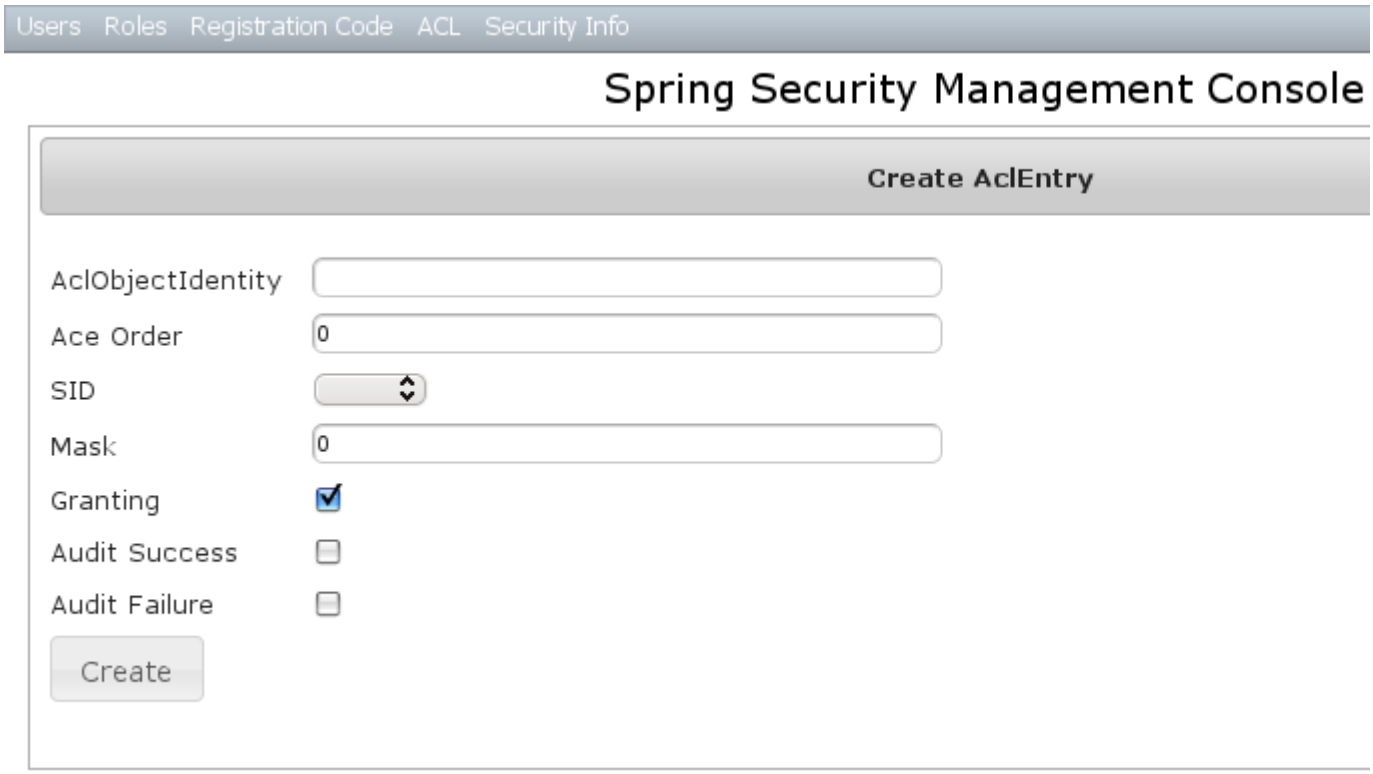
After clicking through to an instance you get to the edit page (there are no view pages):



You can update the values and delete the instance if there aren't any associated `AclEntry` instances - by

AclEntry Create

You can create new instances by going to `/aclEntry/create` or by clicking the `Create` action in the

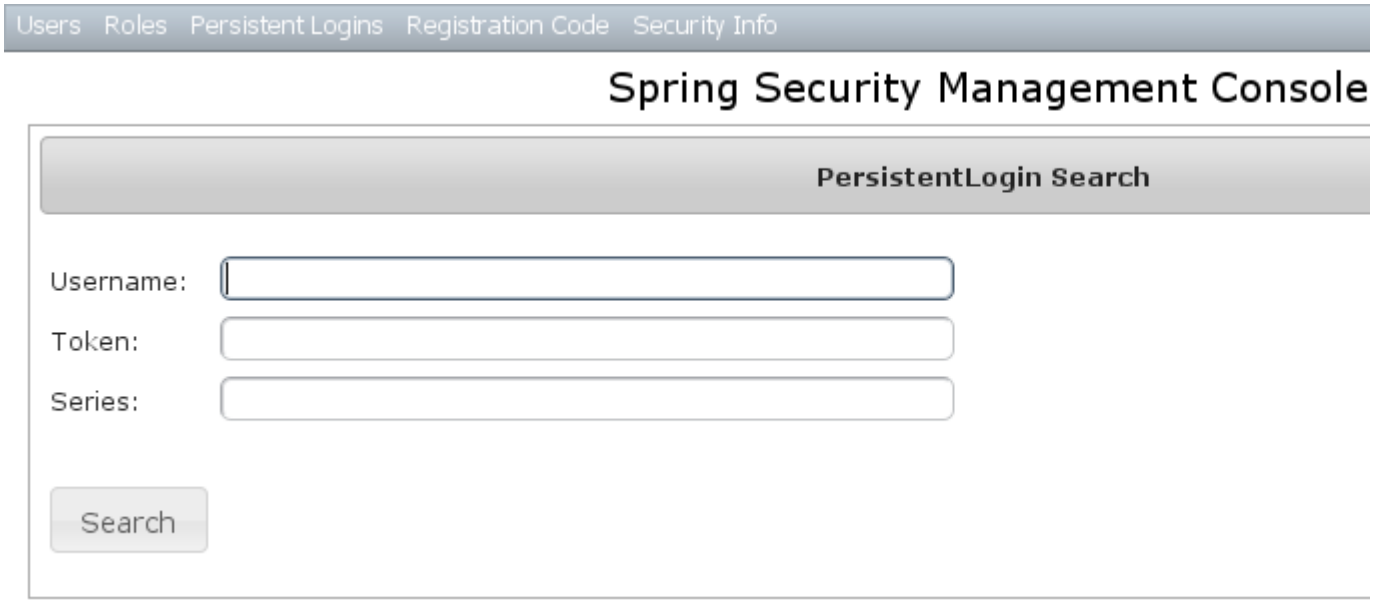


8 Persistent Cookie Management

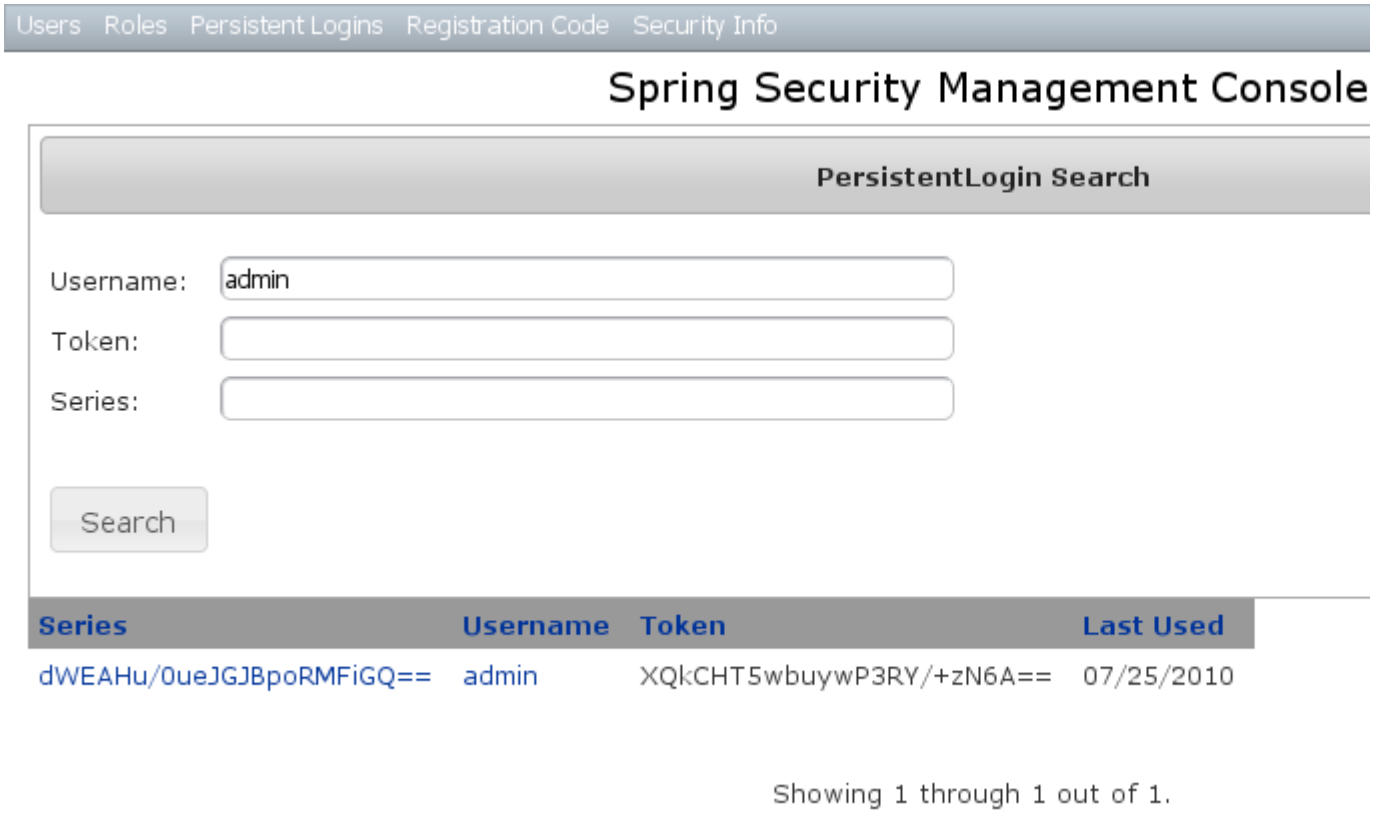
Persistent cookies aren't enabled by default - you must enable them by running the `s2-create-pers`
documentation for details about this feature.

Persistent logins search

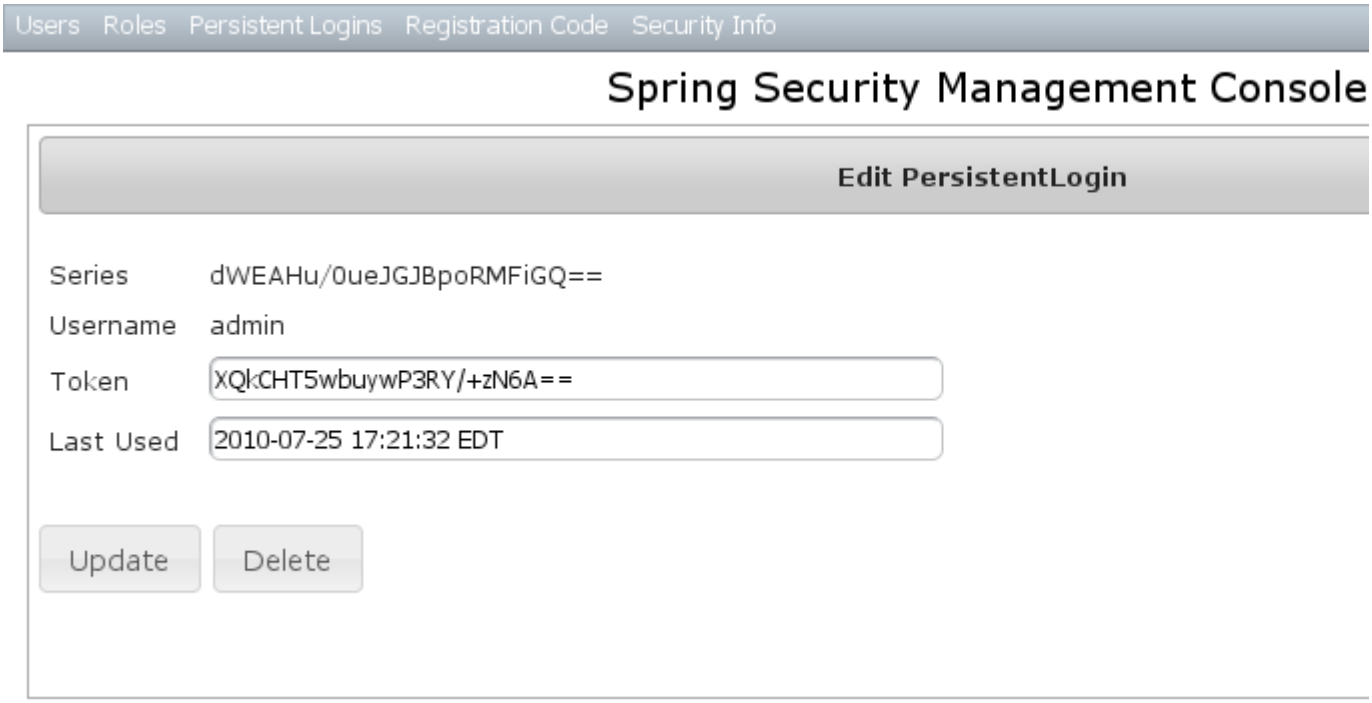
The Persistent Logins menu is only shown if this feature is enabled. Navigate to `/persistentLogin/s`



You can search by any field, and there's an Ajax autocomplete for the `username` field to assist with user

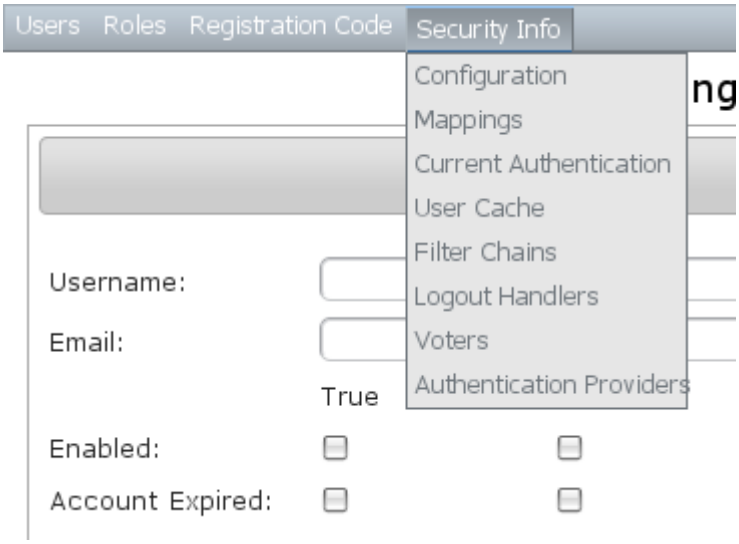


Click the series to get to the edit page:


Users Roles Persistent Logins Registration Code Security Info
Spring Security Management Console
Edit PersistentLogin
Series dWEAHu/0ueJGJBpoRMFiGQ==
Username admin
Token XQkCHT5wbuywP3RY/+zN6A==
Last Used 2010-07-25 17:21:32 EDT
Update
Delete

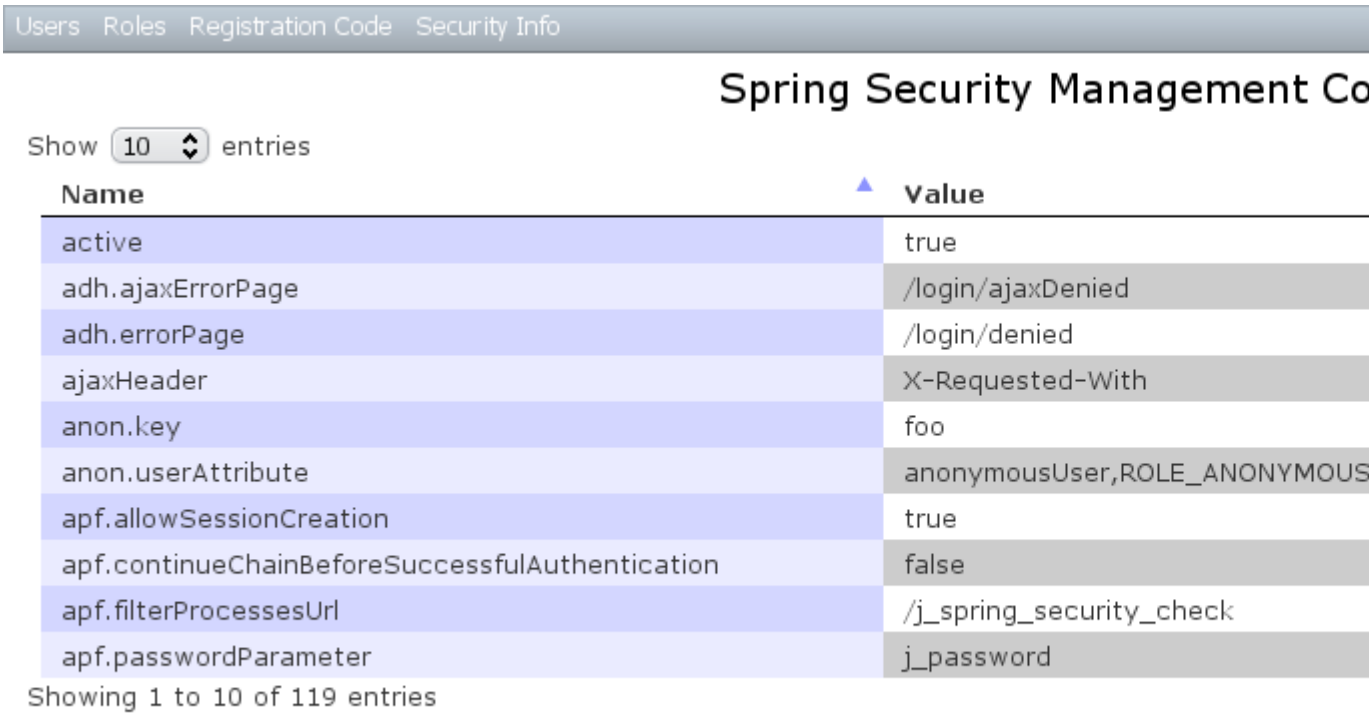
9 Security Configuration UI

The Security Info menu has links for several pages that contain read-only views of much of the Spring Sec



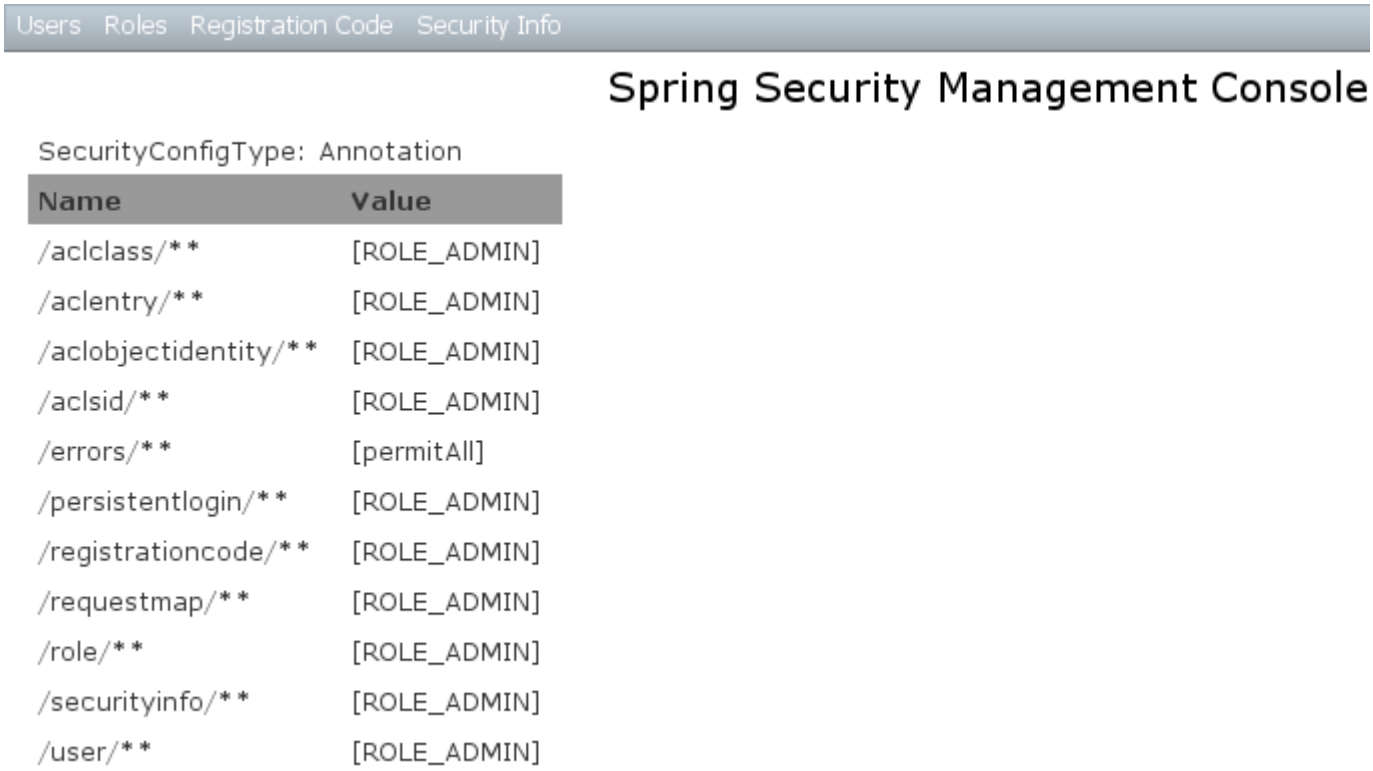
Configuration

The Configuration menu item displays all security-related attributes in `Config.groovy`. The names om



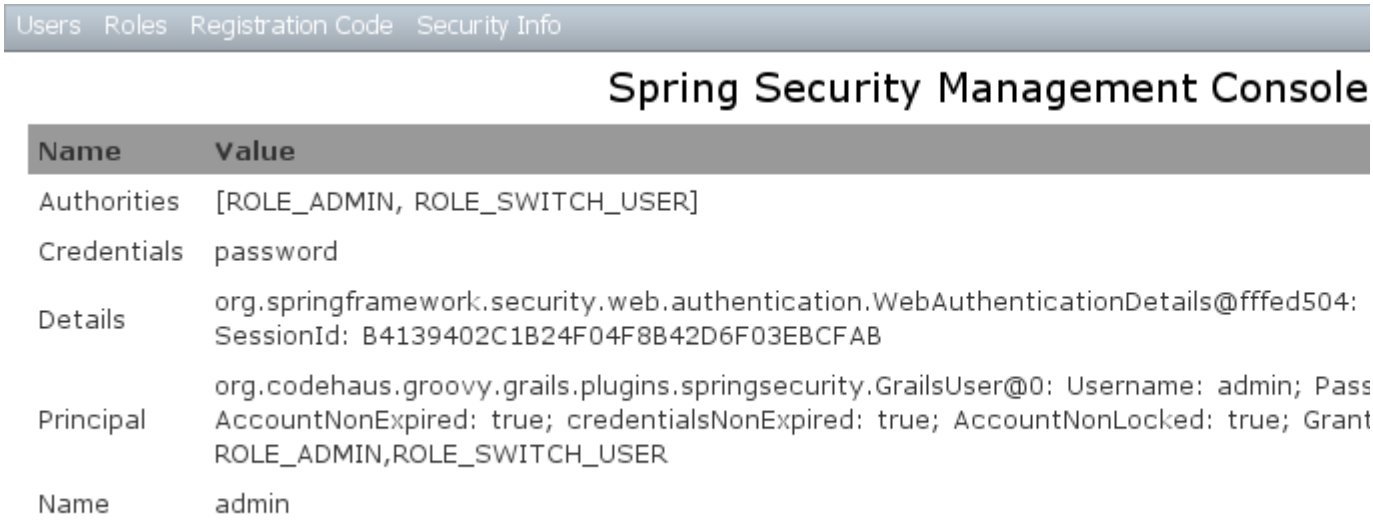
Mappings

The Mappings menu item displays the current request mapping mode (Annotation, Requestmap, or Static)



Current Authentication

The Current Authentication menu item displays your `Authentication` information, mostly for referenc



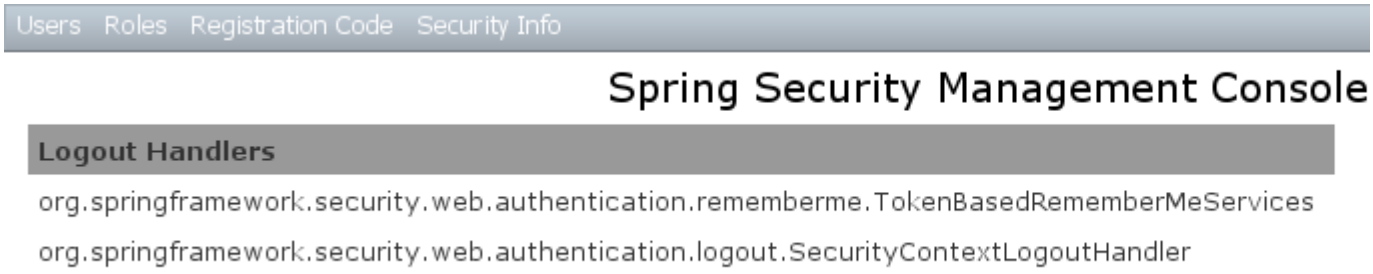
Filter Chains

The Filter Chains menu item displays your configured Filter chains. Typically there is just one chain, appli

It is possible to have multiple URL patterns each with its own filter chain, for example when using HTTP I

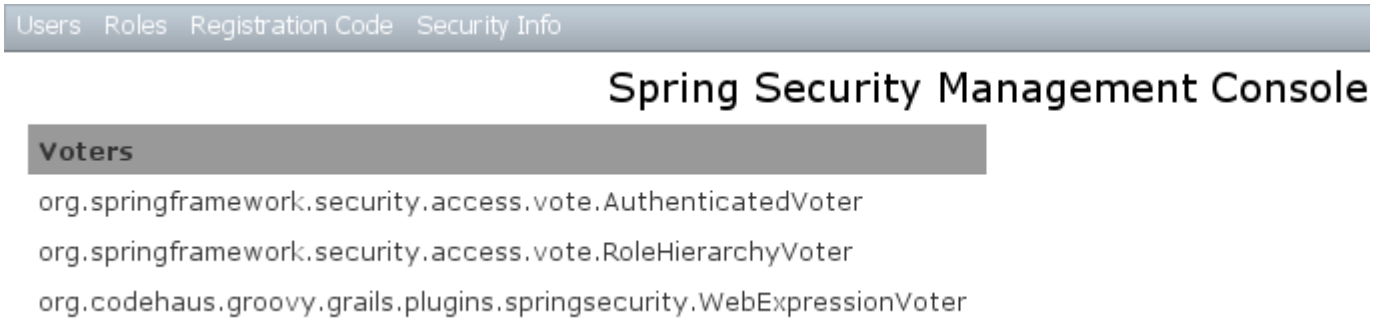
Logout Handlers

The Logout Handlers menu item displays your registered `LogoutHandler`s. Typically there will be j implementations, or a plugin might contribute one or more:



Voters

The Voters menu item displays your registered `AccessDecisionVoter`s. Typically there will be ju implementations, or a plugin might contribute one or more:



Authentication Providers

The Authentication Providers menu item displays your registered `AuthenticationProvider`s. Typi your own custom implementations, or a plugin (e.g. LDAP) might contribute one or more:

Spring Security Management Console

Authentication Providers
org.springframework.security.authentication.dao.DaoAuthenticationProvider
org.springframework.security.authentication.AnonymousAuthenticationProvider
org.springframework.security.authentication.RememberMeAuthenticationProvider

10 Customization

Most aspects of the plugin are configurable.

`s2ui-override` script

The plugin's controllers and GSPs are easily overridden using the `s2ui-override` script. The general s

```
grails s2ui-override <type> <controller-package>
```

The script will copy an empty controller that extends the corresponding plugin controller into your app needed. It also copies the controller's GSPs. The exceptions are 'auth' and 'layout' which only copy GSPs.

The files copied for each type are summarized here:

- aclclass
  - `controller/AclClassController.groovy`
  - `views/aclClass/create.gsp`
  - `views/aclClass/edit.gsp`
  - `views/aclClass/search.gsp`
- aclentry
  - `controller/AclEntryController.groovy`
  - `views/aclEntry/create.gsp`
  - `views/aclEntry/edit.gsp`
  - `views/aclEntry/search.gsp`
- aclobjectidentity
  - `controller/AclObjectIdentityController.groovy`
  - `views/aclObjectIdentity/create.gsp`
  - `views/aclObjectIdentity/edit.gsp`
  - `views/aclObjectIdentity/search.gsp`
- aclsid
  - `controller/AclSidController.groovy`
  - `views/aclSid/create.gsp`
  - `views/aclSid/edit.gsp`
  - `views/aclSid/search.gsp`
- auth
  - `views/login/auth.gsp`

- layout
  - `views/layouts/springSecurityUI.gsp`
  - `views/includes/_ajaxLogin.gsp`
- persistentlogin
  - `controller/PersistentLoginController.groovy`
  - `views/persistentLogin/edit.gsp`
  - `views/persistentLogin/search.gsp`
- register
  - `controller/RegisterController.groovy`
  - `views/register/forgotPassword.gsp`
  - `views/register/index.gsp`
  - `views/register/resetPassword.gsp`
- registrationcode
  - `controller/RegistrationCodeController.groovy`
  - `views/registrationCode/edit.gsp`
  - `views/registrationCode/search.gsp`
- requestmap
  - `controller/RequestmapController.groovy`
  - `views/requestmap/create.gsp`
  - `views/requestmap/edit.gsp`
  - `views/requestmap/search.gsp`
- role
  - `controller/RoleController.groovy`
  - `views/role/create.gsp`
  - `views/role/edit.gsp`
  - `views/role/search.gsp`
- securityinfo

    - `controller/SecurityInfoController.groovy`
    - `views/securityInfo/config.gsp`
    - `views/securityInfo/currentAuth.gsp`
    - `views/securityInfo/filterChain.gsp`
    - `views/securityInfo/logoutHandlers.gsp`
    - `views/securityInfo/mappings.gsp`
    - `views/securityInfo/providers.gsp`
    - `views/securityInfo/usercache.gsp`
    - `views/securityInfo/voters.gsp`
- user
    - `controller/UserController.groovy`
    - `views/user/create.gsp`
    - `views/user/edit.gsp`
    - `views/user/search.gsp`

I18N

Most of the plugin's displayed strings are localized and stored in the plugin's `grails-app/i18n/me` can override any of these values by putting an override in your application's `grails-app/i18n/mess`

`Config.groovy` attributes

There are a few configuration options specified in `DefaultUiSecurityConfig.gr` `grails-app/conf/Config.groovy`

- grails.plugin.springsecurity.ui.register.emailBody
- grails.plugin.springsecurity.ui.register.emailFrom
- grails.plugin.springsecurity.ui.register.emailSubject
- grails.plugin.springsecurity.ui.register.defaultRoleNames
- grails.plugin.springsecurity.ui.register.postRegisterUrl
- grails.plugin.springsecurity.ui.forgotPassword.emailBody
- grails.plugin.springsecurity.ui.forgotPassword.emailFrom
- grails.plugin.springsecurity.ui.forgotPassword.emailSubject
- grails.plugin.springsecurity.ui.forgotPassword.postResetUrl

See Section 5 and Section 6 for more details on these attributes.

CSS

The plugin defines its CSS styles in `web-app/css/spring-security-ui.css` and most of th overridden by overriding the `springSecurityUI.gsp` template and including your CSS file(s).

Password Encryption

In recent versions of the Spring Security Core plugin, the "User" domain class is generated by the `s2-q` This makes the code simpler (for example in controllers where you create users or update user passwords) presents a problem for plugins like this one since it's not possible to reliably determine if the domain c explicitly calling `springSecurityService.encodePassword()`.

The unfortunate consequence of mixing a newer domain class that does p `springSecurityService.encodePassword()` is the the passwords get double-encrypted, a configuration option you can set to tell this plugin's controllers whether to encrypt or not: `grails.plug`

This option defaults to `false`, so if you have an older domain class that doesn't handle encryption just ena

```
grails.plugin.springsecurity.ui.encodePassword = true
```

Password Verification

By default the registration controller has rather strict requirements for valid passwords; they must be betw letter, at least one number, and at least one symbol from "!@#$%^&". You can customize these rules with

Property	Default Value
grails.plugin.springsecurity. ui.password.minLength	8
grails.plugin.springsecurity. ui.password.maxLength	64
grails.plugin.springsecurity. ui.password.validationRegex	"^.*(?=.*\\d)(?=.*[a-zA-Z])(?=.*[!@#$%^&]).