

# 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-default includes the ACL controllers and views which are enabled if the [ACL plugin](#) is in `grails.plugin.springsecurity.securityConfigType` is set to "Requestmap" or Session and persistent cookies support which is enabled if it has been configured with the `s2-create-persistent`



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

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 sure to install these 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 (username 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 assist. The email field has been added to the domain class and UI. Leave all fields empty and all checkboxes set at Either to

[Users](#) [Roles](#) [Registration Code](#) [Security Info](#)

## Spring Security Management Console

User Search

Username:

Email:

Enabled:

True

False

Either

☐

☐

☒

Account Expired:

True

False

Either

☐

☐

☒

Account Locked:

True

False

Either

☐

☐

☒

Password Expired:

True

False

Either

☐

☐

☒

Search

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:

## Spring Security Management Console

## User Search

Username: Email: 

	True	False	Either
Enabled:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Account Expired:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Account Locked:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Password Expired:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Username	Email	Enabled	Account Expired	Account Locked	Password
farzadmiraiei	farzadmiraiei@foo.com	True	False	False	False
cadmus	cadmus@foo.com	True	False	False	False
chenreadme	chenreadme@foo.com	True	False	False	False
admatha	admatha@foo.com	True	False	False	False
sajjadmohebi	sajjadmohebi@foo.com	True	False	False	False
cadmo	cadmo@foo.com	True	False	False	False
padmawar	padmawar@foo.com	True	False	False	False
padma	padma@foo.com	True	False	False	False
ysunadmin	ysunadmin@foo.com	True	False	False	False
farzadmokh	farzadmokh@foo.com	True	False	False	False

1          

Showing 1 through 10 out of 100.

## User edit

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


### Edit User


[illegible]

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`:

## Spring Security Management Console

## Edit User

 User Details

 Roles

☒ ROLE\_ADMIN

☒ ROLE\_SWITCH\_USER

☐ ROLE\_USER

Update

Delete


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


## OpenIDs


If the [OpenID plugin](#) is installed, you can manage users' associated OpenIDs:

## Spring Security Management Console

### Edit User

 User Details

 Roles

 OpenIDs

`https://someusername.myopenid.com/`

Update


Delete


### User creation

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

## 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

[Users](#) [Roles](#) [Registration Code](#) [Security Info](#)

## Spring Security Management Console

Role Search

Authority:

Search


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):

## Spring Security Management Console

### Edit Role

 Role Details

 Users

Authority

ROLE\_ADMIN


Update


Delete

You can update any of the attributes or delete the role. Any user that had been granted the role will lose the role. If you click the Users tab you can see which users have a grant for this role and can click through to their edit page.

## Spring Security Management Console

### Edit Role

 Role Details

 Users

admin

Update

Delete

### Role creation

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

## 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 `"Requestmap Config.groovy"`.

### Requestmap search

The default action for the Requestmap controller is search. You can search by URL or Config Attribute. Let's see how it works.

[Users](#) [Roles](#) [Requestmaps](#) [Registration Code](#) [Security Info](#)

### Spring Security Management Console

Requestmap Search

URL:

Config Attribute:

Search

Searching is case-insensitive and the search string can appear anywhere in the field. Multiple results are shown in a table with a header to sort by that field:

[Users](#) [Roles](#) [Requestmaps](#) [Registration Code](#) [Security Info](#)

### Spring Security Management Console

Requestmap Search

URL:

Config Attribute:

Search

URL	Config Attribute
<a href="#">/j_spring_security_switch_user</a>	ROLE_SWITCH_USER,IS_AUTHENTICATED_FULLY
<a href="#">/secure/**</a>	ROLE_ADMIN

Showing 1 through 2 out of 2.

### Requestmap edit

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

[Users](#) [Roles](#) [Requestmaps](#) [Registration Code](#) [Security Info](#)

## Spring Security Management Console

Edit Requestmap

URL

Config Attribute

Update

Delete

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 action

[Users](#) [Roles](#) [Requestmaps](#) [Registration Code](#) [Security Info](#)

## Spring Security Management Console

Create Requestmap

URL

Config Attribute

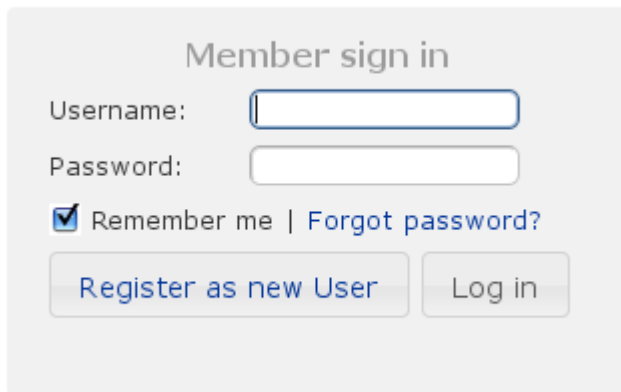
Create

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 functionality.

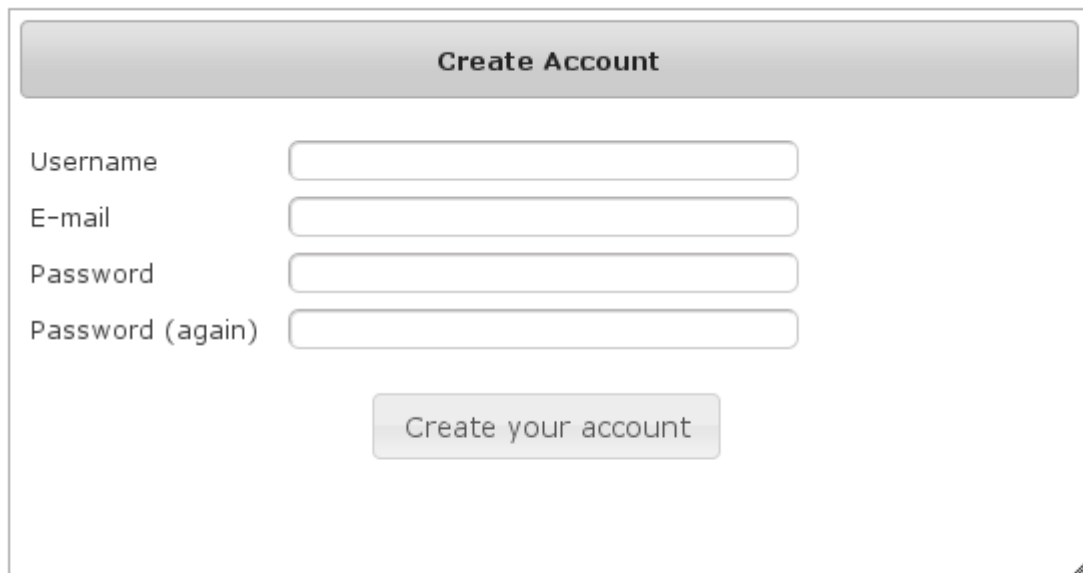
One way to do this is to replace the default `login.gsp` that's provided by the Spring Security Core `grails s2ui-override auth` - see [the section on configuration](#) for more details. If you do this you



A login form titled "Member sign in". It contains two input fields: "Username:" and "Password:". Below the password field is a checkbox labeled "Remember me" which is checked, followed by a link "Forgot password?". At the bottom are two buttons: "Register as new User" and "Log in".

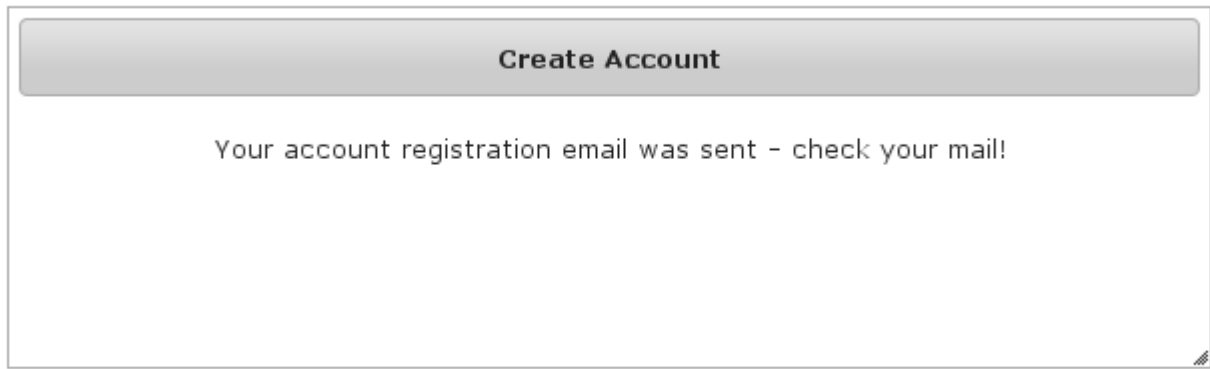
### Registration

Navigate to `/register/`:

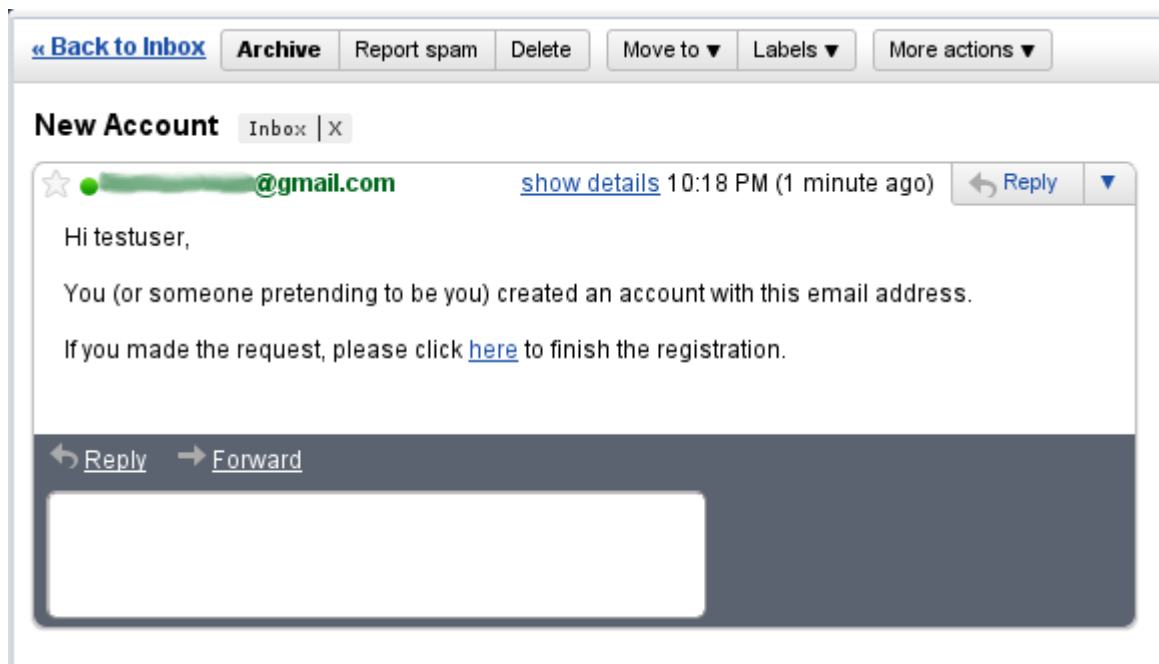


A registration form titled "Create Account". It contains four input fields: "Username", "E-mail", "Password", and "Password (again)". Below these fields is a button labeled "Create your account".

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 redirect



**i** Your registration is complete

Logged in as testuser (Logout)

## 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 default 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 customize (for example if you want an admin to approve new users and explicitly enable new users) then you can customize the 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. See [Mail plugin](#) for more details.

## Notes

You should consider the registration code as starter code - every signup workflow will be different, and they may wish to collect more information than just username and email - first and last name for example. You can customize the 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 them).

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



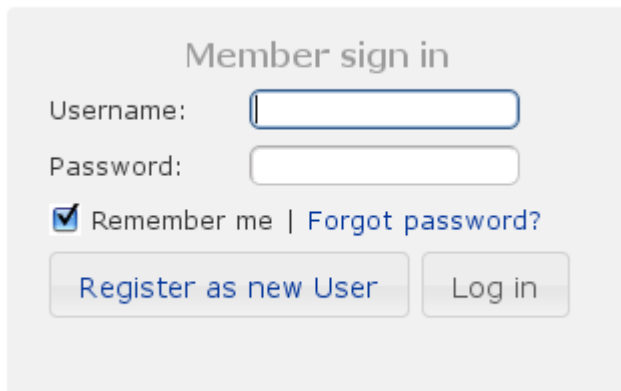
`RegisterController` and its GSPs assume that your `User` domain class has an `email` field (using the `s2ui-override` script) if you don't need an email confirmation step or add an email confirmation step.



## 6 Forgot Password

Like the Registration workflow, the Forgot Password workflow is expected to be user-facing. So it's not backend functionality - you'll need to expose them to your users.

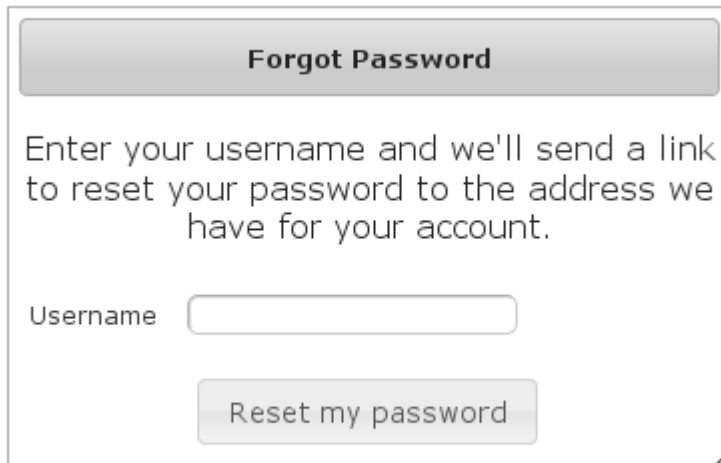
One way to do this is to replace the default `login.gsp` that's provided by the Spring Security Core with `grails.s2ui.override.auth` - see [the section on configuration](#) for more details. If you do this you



A user sign-in form titled "Member sign in". It contains two input fields: "Username:" and "Password:". Below the password field is a checkbox labeled "Remember me" which is checked, followed by a link "Forgot password?". At the bottom are two buttons: "Register as new User" and "Log in".

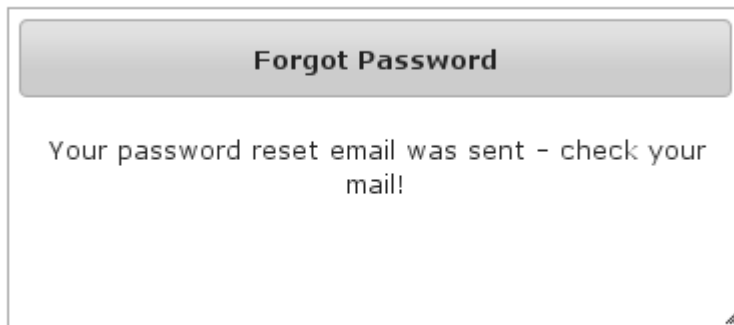
### Forgot Password

Navigate to `/register/forgotPassword`:

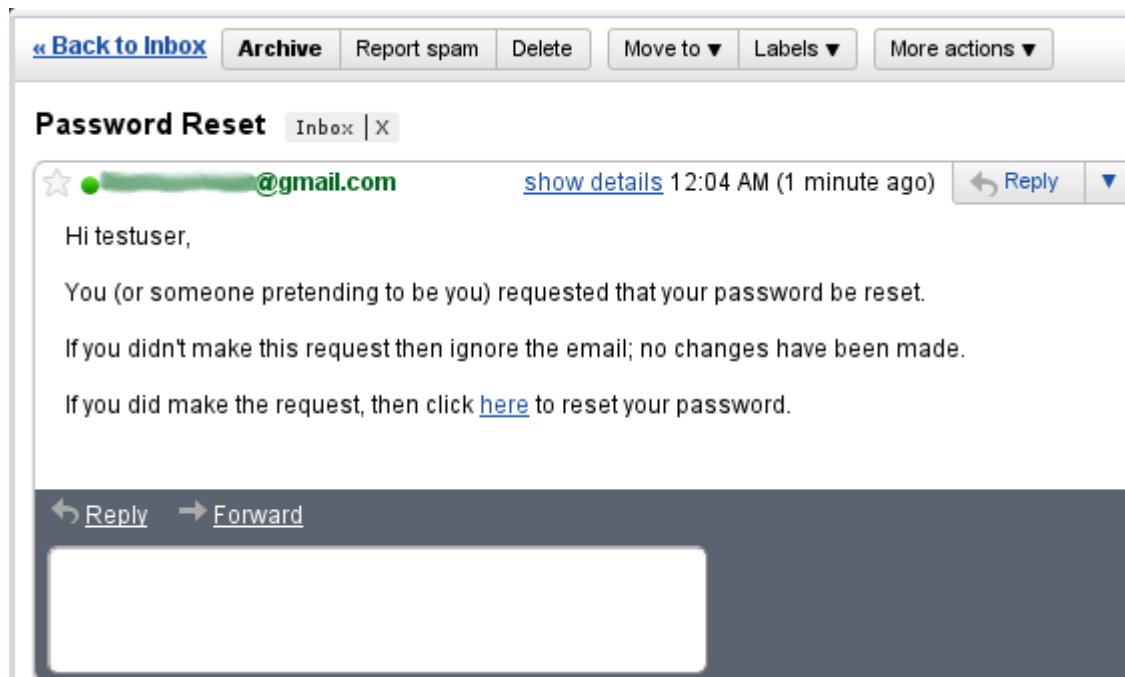


A form titled "Forgot Password" with a header bar. Below the header, it says "Enter your username and we'll send a link to reset your password to the address we have for your account." There is a "Username" label followed by an input field. At the bottom is a button labeled "Reset my password".

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:

A screenshot of a "Reset Password" form. At the top, there is a grey button labeled "Reset Password". Below it, the text reads: "Enter your new password". There are two input fields: "Password" and "Password (again)". Below the input fields, there is a grey button labeled "Update my password". The form is enclosed in a light grey border with a small icon in the bottom right corner.

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

 Your password was successfully changed

Logged in as testuser (**Logout**)

## 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 `AclUtilService`. This plugin provides a high-level approach to managing ACLs, ACEs, etc. The functionality in this plugin is to provide a CRUD interface.

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.

[Users](#) [Roles](#) [Registration Code](#) [Security Info](#)

Spring Security Management Console

AclClass Search

Class Name:

Search

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

[Users](#) [Roles](#) [Registration Code](#) [ACL](#) [Security Info](#)

Spring Security Management Console

AclClass Search

Class Name:

Search

Class Name
<a href="#">com.burtbeckwith.testapp.domain.Report</a>

Showing 1 through 1 out of 1.

### AclClass Edit

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

## Spring Security Management Console

### Edit AclClass

Class Name

[View Associated OIDs](#)

[View Associated ACL Entries](#)

Update

Delete

You can update the name, and delete the instance if there aren't any associated `AclObjectIdentity` instances 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

## Spring Security Management Console

### Create AclClass

Class Name

Create

## 7.2 AclSid Management

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

## Spring Security Management Console

**AclSid Search**

SID:

Principal: ☐ True ☐ False ☒ Either

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

## Spring Security Management Console

**AclSid Search**

SID:

Principal: ☐ True ☐ False ☒ Either

SID	Principal
-----	-----------

<a href="#">user2</a>	True
-----------------------	------

<a href="#">admin</a>	True
-----------------------	------

<a href="#">user1</a>	True
-----------------------	------

Showing 1 through 3 out of 3.

### AclSid Edit

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

## Spring Security Management Console

### Edit AclSid

SID

Principal ☒

[View Associated OIDs](#)

[View Associated ACL Entries](#)

Update

Delete

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:

## Spring Security Management Console

### Create AclSid

SID

Principal ☐

Create

## 7.3 AclObjectIdentity Management

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

## Spring Security Management Console

**AclObjectIdentity Search**

AclClass:

All

Object ID:

Owner:

All

Parent:

Entries Inheriting:

True

False

Either

☐

☐

☒

Search

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



## Spring Security Management Console

AclObjectIdentity Search

AclClass: All ⬆ ⬇ ⬆  
Object ID:   
Owner: All ⬆ ⬇ ⬆  
Parent:   

True
False
Either

Entries Inheriting: ☐ True ☐ False ☒ Either  

Search

ID	AclClass	Object ID	Entries Inheriting	Owner	Parent
3	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	3	True	<a href="#">admin</a>	
6	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	6	True	<a href="#">admin</a>	
1	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	1	True	<a href="#">user1</a>	
4	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	4	True	<a href="#">admin</a>	
10	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	10	True	<a href="#">admin</a>	
8	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	8	True	<a href="#">admin</a>	
7	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	7	True	<a href="#">admin</a>	
5	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	5	True	<a href="#">admin</a>	
9	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	9	True	<a href="#">admin</a>	
2	<a href="#">com.burtbeckwith.testapp.domain.Report</a>	2	True	<a href="#">user1</a>	

1 2 3 4 5 6 7 8 9 10 Next

Showing 1 through 10 out of 100.

## AclObjectIdentity Edit

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

## Spring Security Management Console

Edit AclObjectIdentity

AclClass	<input type="text" value="com.burtbeckwith.testapp.domain.Report"/>
Object ID	<input type="text" value="3"/>
Owner	<input type="text" value="admin"/>
Parent	<input type="text"/>
Entries Inheriting	<input checked="" type="checkbox"/>

[View Associated ACL Entries](#)

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

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` button.

## Spring Security Management Console

Create AclObjectIdentity

AclClass	<input type="text"/>
Object ID	<input type="text"/>
Owner	<input type="text"/>
Parent	<input type="text"/>
Entries Inheriting	<input type="checkbox"/>

## 7.4 AclEntry Management

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

## Spring Security Management Console

**AclEntry Search**

AclObjectIdentity:

Ace Order:

SID:

All
↕

Mask:

	True	False	Either
Granting:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Audit Success:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Audit Failure:	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>


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

## Spring Security Management Console

**AclEntry Search**

AclObjectIdentity:

Ace Order:

SID: All 

Mask:

Granting: ☐ True ☐ False ☒ Either

Audit Success: ☐ True ☐ False ☒ Either

Audit Failure: ☐ True ☐ False ☒ Either

ID	AclObjectIdentity	Ace Order	SID	Mask	Grant
99	5	2	user2	BasePermission[.....W.=2]	True
94	4	0	user1	BasePermission[.....R=1]	True
95	4	1	user2	BasePermission[.....R=1]	True
100	5	3	admin	BasePermission[.....A....=16]	True
96	4	2	admin	BasePermission[.....A....=16]	True
93	3	2	admin	BasePermission[.....A....=16]	True
91	3	0	user1	BasePermission[.....R=1]	True
92	3	1	user2	BasePermission[.....R=1]	True
97	5	0	user1	BasePermission[.....R=1]	True
98	5	1	user2	BasePermission[.....R=1]	True

1 2 3 4 5 6 7 8 9 10 .. 18 Next

Showing 1 through 10 out of 175.

**AclEntry Edit**

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

## Spring Security Management Console

## Edit AclEntry

AclObjectIdentity	<input type="text" value="5"/>
Ace Order	<input type="text" value="2"/>
SID	<input type="text" value="user2"/>
Mask	<input type="text" value="2"/>
Granting	<input checked="" type="checkbox"/>
Audit Success	<input type="checkbox"/>
Audit Failure	<input type="checkbox"/>

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

## AclEntry Create

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

## Spring Security Management Console

## Create AclEntry

AclObjectIdentity	<input type="text"/>
Ace Order	<input type="text" value="0"/>
SID	<input type="text"/>
Mask	<input type="text" value="0"/>
Granting	<input checked="" type="checkbox"/>
Audit Success	<input type="checkbox"/>
Audit Failure	<input type="checkbox"/>

## 8 Persistent Cookie Management

Persistent cookies aren't enabled by default - you must enable them by running the `s2-create-per:` [documentation](#) for details about this feature.

### Persistent logins search

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

Users Roles Persistent Logins Registration Code Security Info

Spring Security Management Console

PersistentLogin Search

Username:

Token:

Series:

Search

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

Users Roles Persistent Logins Registration Code Security Info

Spring Security Management Console

PersistentLogin Search

Username:

Token:

Series:

Search

Series	Username	Token	Last Used
dWEAHu/0ueJGJBpoRMFiGQ==	admin	XQkCHT5wbuywP3RY/+zN6A==	07/25/2010

Showing 1 through 1 out of 1.

Click the series to get to the edit page:

## Spring Security Management Console

### Edit PersistentLogin

Series dWEAHu/0ueJGJBpoRMFiGQ==

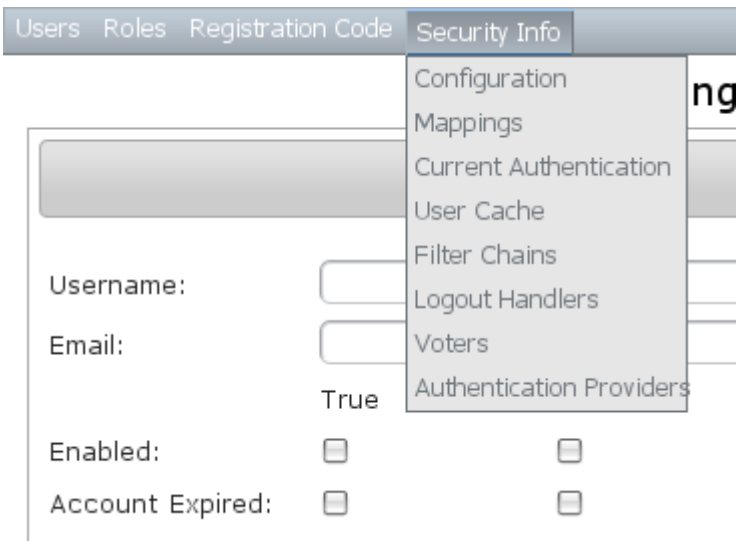
Username admin

Token

Last Used

# 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

Spring Security Management Co	
Users Roles Registration Code Security Info	
Show 10 entries	
Name	Value
active	true
adh.ajaxErrorPage	/login/ajaxDenied
adh.errorPage	/login/denied
ajaxHeader	X-Requested-With
anon.key	foo
anon.userAttribute	anonymousUser,ROLE_ANONYMOUS
apf.allowSessionCreation	true
apf.continueChainBeforeSuccessfulAuthentication	false
apf.filterProcessesUrl	/j_spring_security_check
apf.passwordParameter	j_password
Showing 1 to 10 of 119 entries	

## Mappings

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



## Spring Security Management Console

SecurityConfigType: Annotation

Name	Value
/aclclass/**	[ROLE_ADMIN]
/aclentry/**	[ROLE_ADMIN]
/aclobjectidentity/**	[ROLE_ADMIN]
/aclsid/**	[ROLE_ADMIN]
/errors/**	[permitAll]
/persistentlogin/**	[ROLE_ADMIN]
/registrationcode/**	[ROLE_ADMIN]
/requestmap/**	[ROLE_ADMIN]
/role/**	[ROLE_ADMIN]
/securityinfo/**	[ROLE_ADMIN]
/user/**	[ROLE_ADMIN]

### Current Authentication

The Current Authentication menu item displays your Authentication information, mostly for reference.

## Spring Security Management Console

Name	Value
Authorities	[ROLE_ADMIN, ROLE_SWITCH_USER]
Credentials	password
Details	org.springframework.security.web.authentication.WebAuthenticationDetails@fffed504: SessionId: B4139402C1B24F04F8B42D6F03EBCFAB
Principal	org.codehaus.groovy.grails.plugins.springsecurity.GrailsUser@0: Username: admin; Password: AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true; Granted Authorities: ROLE_ADMIN,ROLE_SWITCH_USER
Name	admin

### Filter Chains

The Filter Chains menu item displays your configured Filter chains. Typically there is just one chain, applied to all requests.

## Spring Security Management Console

### URL Pattern Filters

/**	org.springframework.security.web.context.SecurityContextPersistenceFilter org.codehaus.groovy.grails.plugins.springsecurity.MutableLogoutFilter org.codehaus.groovy.grails.plugins.springsecurity.RequestHolderAuthenticationFilter org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter org.springframework.security.web.authentication.rememberme.RememberMeAuthenticationFilter org.springframework.security.web.authentication.AnonymousAuthenticationFilter org.springframework.security.web.access.ExceptionTranslationFilter org.springframework.security.web.access.intercept.FilterSecurityInterceptor
-----	---

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

## Logout Handlers

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

## Spring Security Management Console

### Logout Handlers

org.springframework.security.web.authentication.rememberme.TokenBasedRememberMeServices org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler
--

## Voters

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

## Spring Security Management Console

### Voters

org.springframework.security.access.vote.AuthenticatedVoter org.springframework.security.access.vote.RoleHierarchyVoter org.codehaus.groovy.grails.plugins.springsecurity.WebExpressionVoter
--

## Authentication Providers

The Authentication Providers menu item displays your registered AuthenticationProviders. Typic 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/messages`. You can override any of these values by putting an override in your application's `grails-app/i18n/messages`.

## Config.groovy attributes

There are a few configuration options specified in `DefaultUiSecurityConfig.groovy` in `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 them are 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` plugin. This makes the code simpler (for example in controllers where you create users or update user passwords) but presents a problem for plugins like this one since it's not possible to reliably determine if the domain class is explicitly calling `springSecurityService.encodePassword()`.

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

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

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

## Password Verification

By default the registration controller has rather strict requirements for valid passwords; they must be between 8 and 64 characters long, contain at least one letter, at least one number, and at least one symbol from `!"#$%^&".` You can customize these rules with

Property	Default Value
<code>grails.plugin.springsecurity.ui.password.minLength</code>	8
<code>grails.plugin.springsecurity.ui.password.maxLength</code>	64
<code>grails.plugin.springsecurity.ui.password.validationRegex</code>	<code>"^(?=.*[0-9])(?=.*[a-zA-Z])(?=.*[!@#\$%^&amp;]).*\$"</code>