Customizing The Grails Spring Security UI Plugin

By Amy Andrychowicz
on May 24, 2011

Security plugin options for Grails have come a long way since I started working with Grails two years ago. Back then we had a few options for security plugins, with the Acegi plugin being the most popular. Last year, securing a Grails application became much easier with the release of the Spring Security Core plugin. Today we have more options than ever with the release of several Spring Security plugins that have been added as extensions to the core plugin (http://grails.org/plugin/spring-security-core). Grails developers can choose to add any of these optional plugins to customize the security of their application as much as they need (or want) to.

One of those plugins is the Spring Security UI plugin. This plugin provides user interfaces and basic functionality for user, role and requestmap management, user registration and forgot password functionality. The best part is that it's fully customizable for your web application. In true Grails style, it is trivial to install and customize this plugin for your application.

Note: Since this is an extension of the spring-security-core plugin, you should already have the core plugin installed and running in your project before installing the UI plugin.

You can download the Spring Security UI plugin and find the documentation here…

http://grails.org/plugin/spring-security-ui

Install the plugin by running the following command…

```
grails install-plugin spring-security-ui
```

Most of the controllers and views included with this plugin are intended to be part of a backend admin functionality. The registration and forgot password screens are meant to be exposed to all users.

For the purpose of this article, I'll focus on customizing the forgot password screens and functionality in an existing application. Note: The forgot password functionality uses the Mail plugin to send emails (http://grails.org/plugin/mail), so it should also be installed and configured in your application as well.

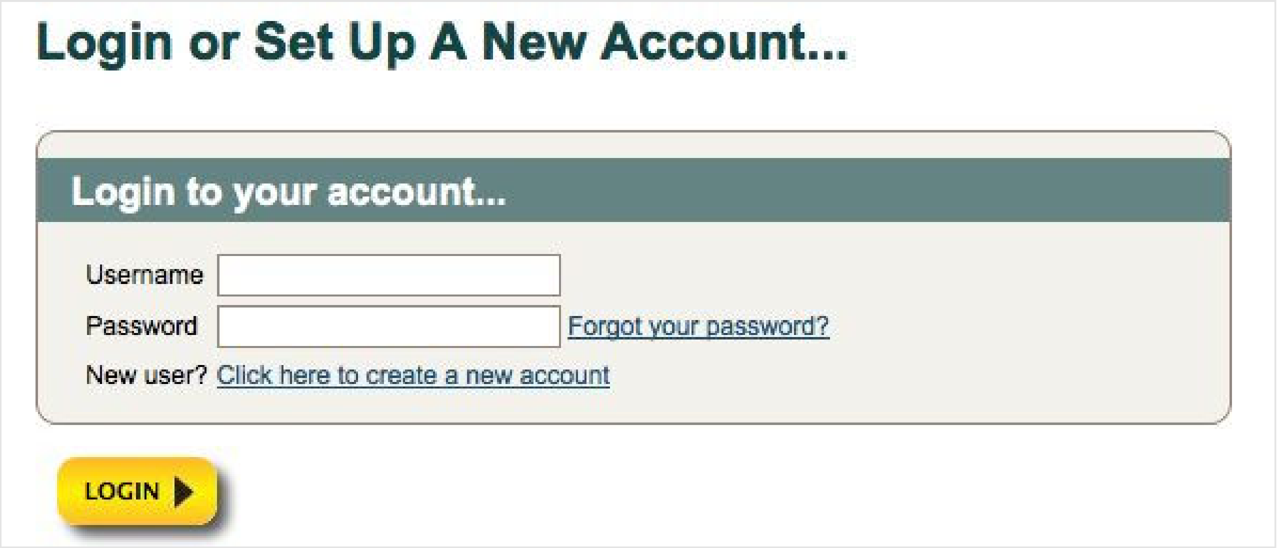
The Spring Security UI plugin provides an option to add a new login screen to your application which will replace the Spring Security Core plugin's default login.gsp. This login screen will include links to the user registration and forgot password screens that were created during the installation of the UI plugin. If you've already customized the core security login.gsp, you can add these links to your application manually instead.

To replace the existing login.gsp file in your application with the new one provided by the Spring Security UI plugin, run…

grails s2ui-override auth
For our application, we already have a customized login screen, so I added the links to the existing screen manually. The code to add a link for the forgot password functionality is…

<g:link controller=”register” action=”forgotPassword”>Forgot your password?</g:link>

Here is our customized login screen with the forgot password link…



Login or Set Up A New Account...

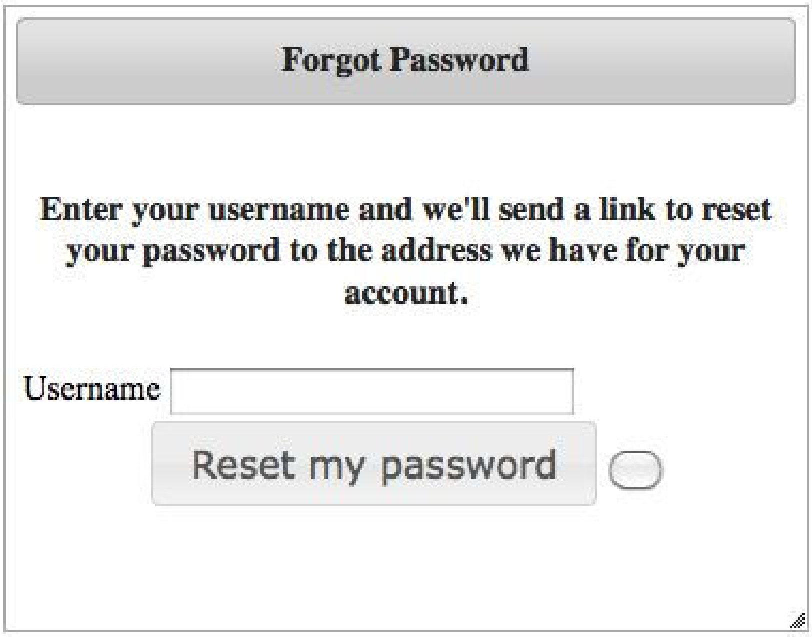
Login to your account...

Username

Password Forgot your password?

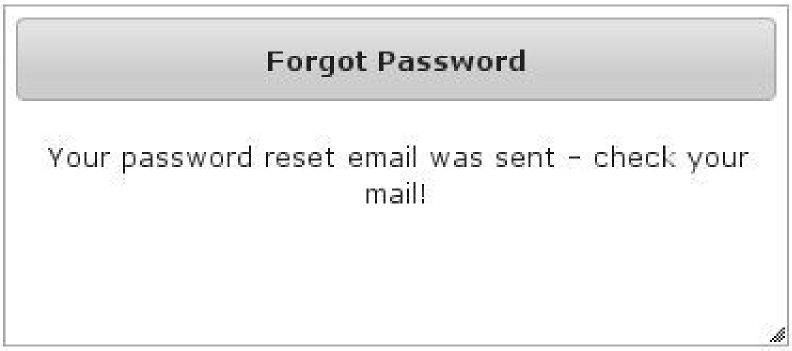
New user? Click here to create a new account

Clicking on this new link will bring you to the default functionality (included with the Spring Security UI Plugin installation) for forgot password. The default views you get with this plugin are what you'd expect to get from Grails scaffolding. If everything is setup correctly in your Spring Security Core plugin and Mail plugin installations, these screens should just work. They're not pretty, but they are functional.



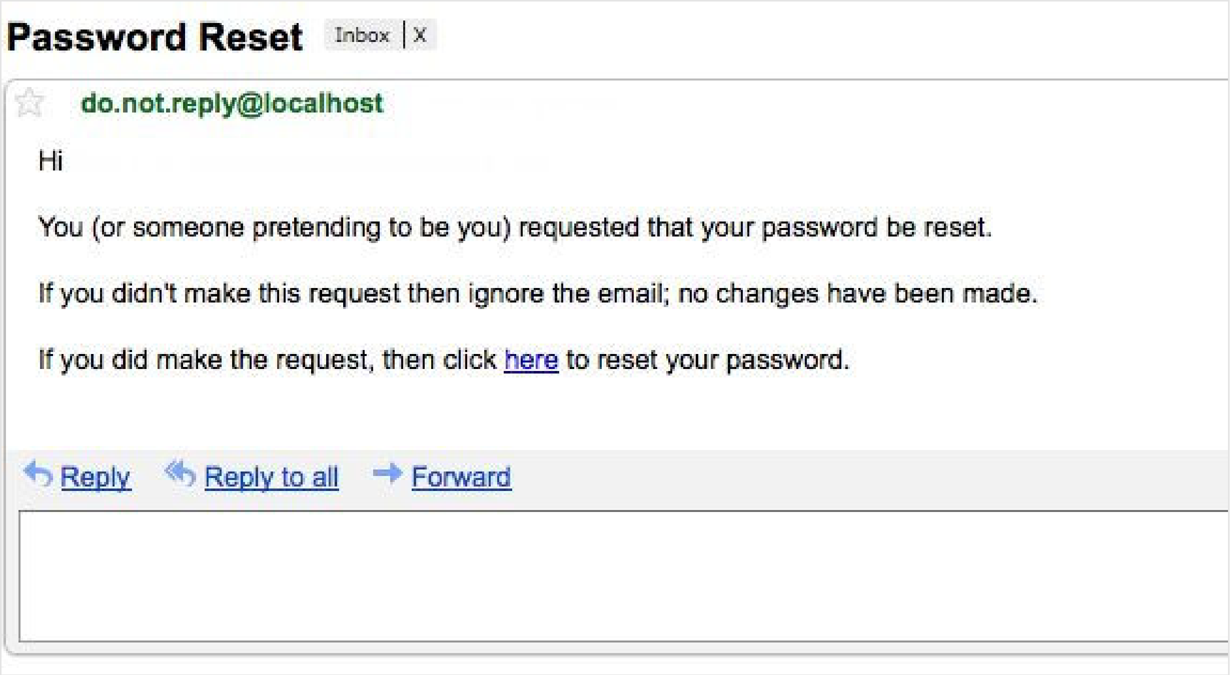
Forgot Password

Enter your username and we'll send a link to reset your password to the address we have for your account.



Forgot Password

Your password reset email was sent - check your mail!



Password Reset

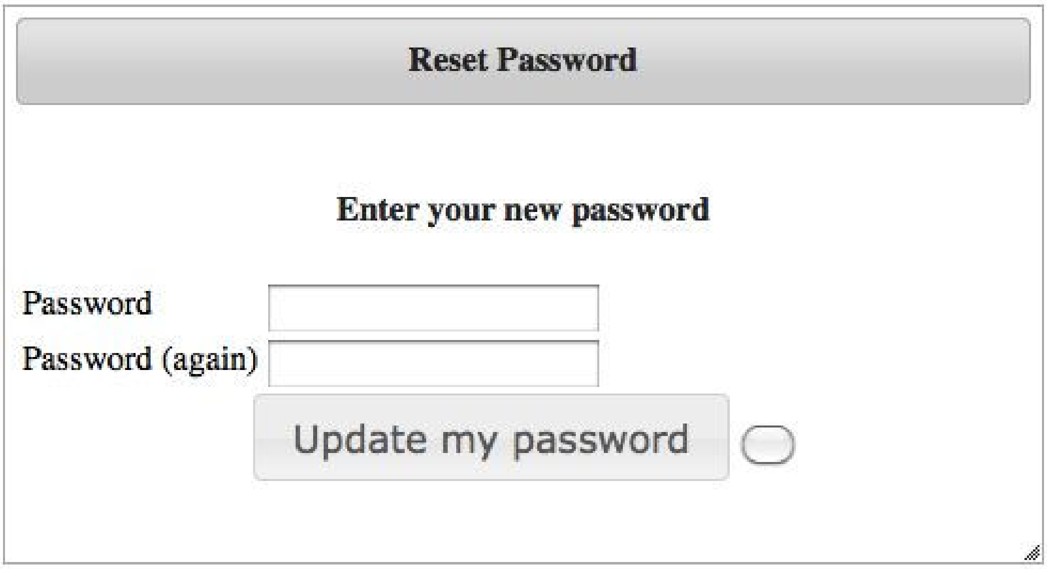
do.not.reply@localhost

Hi

You (or someone pretending to be you) requested that your password be reset.

If you didn't make this request then ignore the email; no changes have been made.

If you did make the request, then click here to reset your password.



Unless you are planning on using all the default screens and functionality in your application, you will want to do some level of customization. The Spring Security UI plugin makes it very easy to customize any part of this plugin by providing a *s2ui-override* script. Running the *s2ui-override* script for the appropriate *type* will add the necessary controller (if applicable) and views to your application for you to use to override the default functionality.

The controller that is created by this script will be empty and extend the corresponding plugin controller. The GSPs that are created by this script are copies of the corresponding plugin GSPs. (see the plugin documentation for a list of all the *s2ui-override* script *types* available)

The general syntax for running this script is…

grails s2ui-override <type> <controller-package>
To customize the forgot password functionality, we need to run the *s2ui-override* script with the *register* type and give it the location where it should create the RegisterController. So, for example…

grails s2ui-override register com.yourapp.register
After running the script, the screens should look and function exactly the same, but you'll notice some new files have been added to your application…

- controllers/com/yourapp/register/RegisterController.groovy
- views/register/forgotPassword.gsp
- views/register/index.gsp
- views/register/resetPassword.gsp
- views/layouts/register.gsp

That’s it, now you can customize away by overriding the actions and methods in <plugin-location>grails-app/controllers/grails/plugins/springsecurity/ui/RegisterController as you need to. (Just like overriding methods in Java; you only override the methods you want to customize, the others will continue to use the parent functionality) Look at the plugins version of the RegisterController to see what the default code looks like, and what actions and methods you can override.

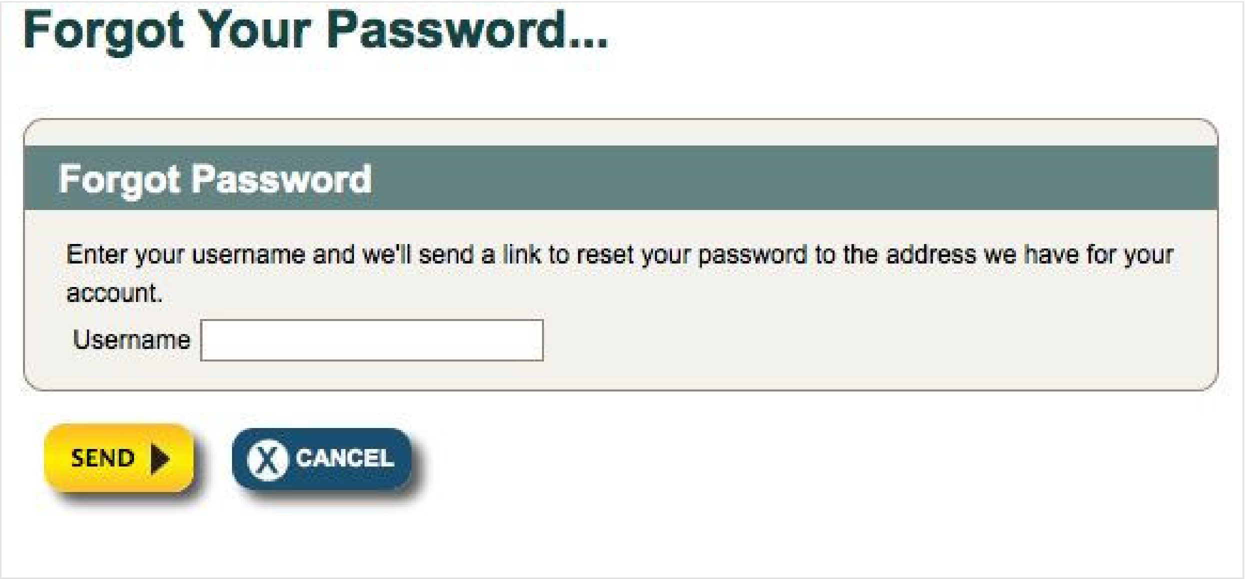
You could override any of these actions to turn off functionality, or redirect to another controller in your application. For the forgot password functionality, you’ll want to look at the forgotPassword and resetPassword actions. For the purpose of our application, I needed to make a few tweaks to these actions. Mainly small changes to get the code to work with our domain. I also modified the code to customize the password validation and the error handling.

As for the views, you could either modify the GSP files to style them for your application, or you could strip out all of the default code and replace it with your own, custom code. The UI plugin also provides a layout and a stylesheet. The layout.gsp is copied into your application, which you can modify to fit your application design. The CSS is file is in the plugin directory and is located at <plugin-location>web-app/css/spring-security-ui.css. You can override any of these styles in your applications CSS file.

All of the text on the screens is also customizable by overriding the default message.properties file (located at <plugin-location>grails-app/i18n/messages.properties) in your applications grails-app/i18n/messages.properties file. To customize the email message text, you would override the values in <plugin-location>grails-app/conf/DefaultUiSecurityConfig.groovy by adding those values to your application’s grails-app/conf/Config.groovy file.

One other thing to note, if you only want to override the forgot password functionality, you can remove the views/register/index.gsp from your application. In my case, I also removed the views/layouts/register.gsp file because we already have a layout defined in our application.

Here are the final screens in our application after I completed my customization of the forgot password screens in the Spring Security UI plugin…

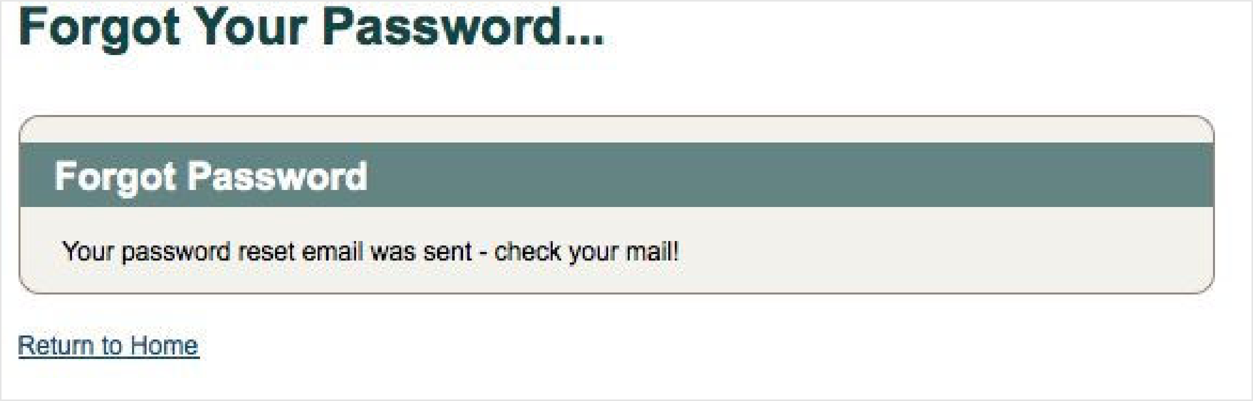


Forgot Your Password...

Forgot Password

Enter your username and we'll send a link to reset your password to the address we have for your account.

Username

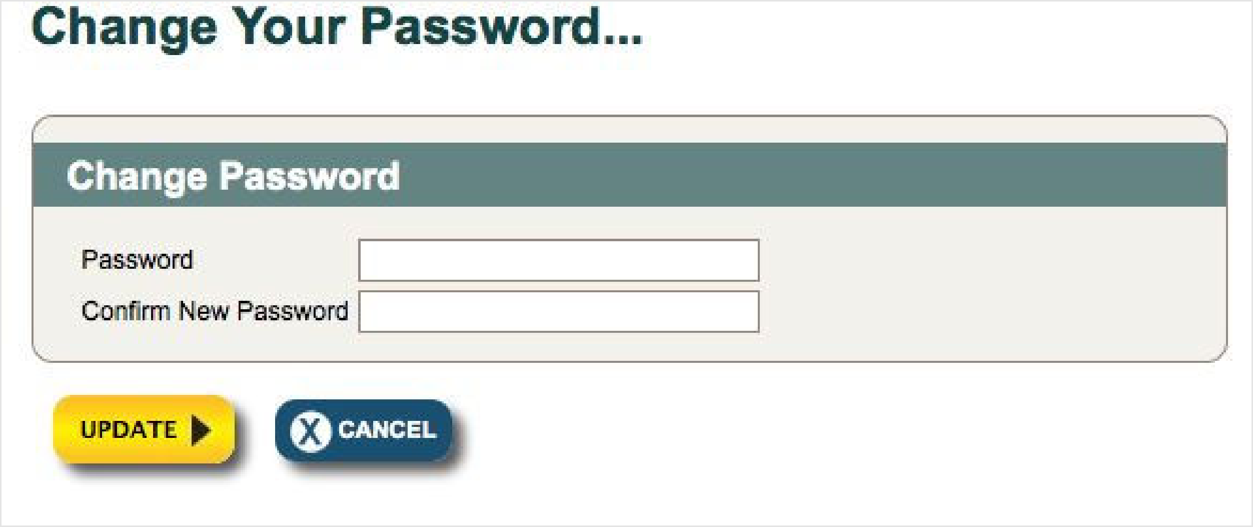


Forgot Your Password...

Forgot Password

Your password reset email was sent - check your mail!

Return to Home



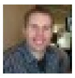
Change Your Password...

Change Password

Password

Confirm New Password

One thought on “Customizing The Grails Spring Security UI Plugin”



Bobby Warner says:

May 24, 2011 at 1:33 pm

Great article!!

Reply

Amy Andrychowicz says:

May 26, 2011 at 6:50 am

Thanks Bobby!

Reply

Pingback: An Army of Solipsists » Blog Archive » This Week in Grails (2011-21)

Sergey Ponomarev says:

May 21, 2013 at 7:12 am

Hey! Images are broken 😕

Reply

Phil Hartley says:

May 21, 2013 at 3:09 pm

Folks, the images are not loading for this page. It's a useful article that I refer other to all the time. Any chance you could put the images back, please?

Reply

Ehren Seim says:

May 23, 2013 at 12:50 pm

This has been fixed.

Reply

Pingback: Instructions needed to integrate spring-security-ui into my grails application | Technology & Programming

Biniam says:
March 17, 2014 at 6:32 am

Thank you. Great article! Posting the code would be much better too.

Reply

Eduardo says:
March 17, 2014 at 8:58 am

Thanks so much for your tutorial, just what I need for my project 😃

Reply

Ganesh says:
December 2, 2014 at 11:01 am

Great Article.

Reply

rik says:
January 14, 2015 at 3:04 pm

Very informative.

Spring Security UI is missing a stand alone 'change password' function. Perhaps at some point you could include the usual, but missing functionality in a tutorial.

Reply

Mark Katerberg says:
April 29, 2015 at 9:13 am

This article fantastic. Thanks for taking the time to help us all! Is there a chance you can get the images back? They are throwing 404s.

Thanks!

Reply

Ryan Andrews says:

July 23, 2015 at 6:08 pm

On thing I cannot figure out is how to give a different message when the user types in a username that is not in the DB. It just goes back to the main page with the form on it with no special message. I dont see anything in the docs about this. I see in the code there is the mailSent variable. Are there any others? How can I see these?

Reply

Chris Betts says:

February 16, 2016 at 4:31 pm

What a great article – saved me a whole heap of time, and still relevant in 2016. Thanks!

Reply

Leave a Reply

Your email address will not be published. Required fields are marked

Name*

Email*

Comment*

Post Comment

ABOUT US

Over the past 19 years, we have an impressive history of successful enterprise software development projects using the best software developers with the best tools and technologies.

CONTACT

Minneapolis, MN (HQ)
(612) 746-1580
Map »

Omaha, NE
(402) 740-5742
Map »

Chicago, IL
(630) 446-0674
Map »

CONTACT FORM

First Name *

Last Name *

Email *

Your Message *

Submit

RECENT POSTS

Setting Up the Raspberry Pi 3 for Home Assistant
by Igor Shults on April 12, 2016

Managing Your Github Org with Terraform
by Jacob Severson on March 23, 2016

Resolving Codenarc Compilation Warnings
by Jennifer Strater on March 16, 2016

Using secondary datasources in Grails 3
by Mike Hostetler on March 9, 2016

Unit Testing URI-Based Grails Filters
by Igor Shults on March 1, 2016

An Introduction to Kotlin
by Mike Plummer on February 23, 2016