The Grails Email Confirmation Plugin

by Marc Palmer

GRAILSROCKS guides

The Grails Email Confirmation Plugin

A Grailsrocks Guide

Marc Palmer

This book is for sale at http://leanpub.com/grails-email-confirmation

This version was published on 2013-01-22

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process.

Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

To learn more about Lean Publishing, go to http://leanpub.com/manifesto.

To learn more about Leanpub, go to http://leanpub.com.

Tweet This Book!

Please help Marc Palmer by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought ”The Grails Email Confirmation Plugin” e-book #grails https://leanpub.com/grails-email-confirmation

The suggested hashtag for this book is #grails.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#grails

Contents

This is a sample

This e-book is an exerpt of the full guide book. Chapters are missing and the chapter numbers may not match that of the full book.

About the author

This short guide book is written by me, Marc Palmer.

I'm a freelancer providing consulting on application usability & user experience in addition to Grails/Groovy development and mentoring for established companies and startups.

I create lots of Open Source Grails stuff under the banner Grailsrocks[1] including a bunch of popular Grails plugins such as the Resources[2] framework, Platform Core[3], Platform UI[4], Feeds[5], Bean-Fields[6] and Cache-Headers[7]. I get excited about using Grails to hide development complexity.

The depth of my contributions to the Grails framework have waxed and waned over the years, but as a Grails developer you are likely using some of my code or ideas every day. From sanitized and reformatted stacktraces to the Resource framework used in Grails 2, there's lots of little bits all over the place.

I am also a founder of the Grails-based startup NoticeLocal[8], and I'm also developing some iOS applications.

You can follow me on Twitter as @wangjammer5 for my no-holds-barred views on Development, Apple, Android and more, or as @grails_rocks for just the Grails bits. More detailed articles are posted on my blog at www.anyware.co.uk[9].

Please do e-mail me with suggestions and corrections for this book at marc@grailsrocks.com

[1] http://grailsrocks.com
[2] http://grck.it/grails-resources-plugin
[3] http://grails.org/plugin/platform-core
[4] http://grails.org/plugin/platform-ui
[5] http://grails.org/plugin/feeds
[6] http://grails.org/plugin/bean-fields
[7] http://grails.org/plugin/cache-headers
[8] http://noticelocal.com
[9] http://www.anyware.co.uk

Foreword

First, I would like to thank you for buying this guide to one of my Grails plugins.

If you didn't in fact buy it, please consider going to LeanPub[10] to pay for it or one of my other titles. I do not get paid for working on the many Grailsrocks[11] plugins that I have been developing for over five years.

I would love to spend more time on them making them better, and at the moment selling e-book guides looks like a possible way of sustaining my Grails plugin development. The contracting work that results from having a high profile in the Grails community is great, but it only serves to do the exact opposite by preventing me actively working on the plugins. As a result, I have more than my share of outstanding JIRA issues across all the plugins - far more than I will ever have time to tackle unless I can fund the time.

It would be fair to say that I have created too many plugins to be able to maintain them reasonably myself in spare time and that this is my own stupid fault. I think this sometimes too. The trouble is that I need almost all of these plugins myself for my own applications and client work, and frankly even flawed solutions are better than no solutions at all.

A word on the Grails plugin community

I firmly believe that the quality of the plugin ecosystem is pivotal to the continued success of Grails. I love using Grails to develop web applications and that is why I feel so strongly about finding ways to improve the quality of plugins. Plugins are Grails' biggest advantage over many of the myriad other web application frameworks out there.

I am pretty adamant that Grails plugins should:

1. Work out of the box without any mandatory config or code generation scripts
2. Add significant value rather than just wrapping existing Java libraries
3. Be *Grailsy*

If plugins don't meet these criteria, the developer experience is poor and not befitting the *"Grails way"*.

Assumptions about the reader

I have not included full instructions in this book for creating the necessary Grails artefacts. This won't be a step-by-step guide for writing a complete application.

[10] http://grck.it/grailsrocks-books
[11] http://grailsrocks.com

The assumption is that the reader is a reasonably proficient Grails developer who knows that when I suggest they create a controller with a given name, they don't have to worry about how to achieve this or at least are happy to Google it. They will also know that they need to create view GSPs for the actions etc.

1 Introduction

Sending e-mail to users so that they can confirm their identity is a common feature of web-applications. Even in today's world of OAuth authentication with third party services such as Facebook and Twitter, you will often need to verify that the user owns the email address they have entered.

While some product managers may not realise this, asking people to enter their email address twice and checking they match is pretty pointless. *You still don't know they will receive e-mail sent to that address.*

Now, you could be forgiven for thinking that e-mail confirmation is just about account sign up. In reality web applications often perform actions that may need to be double checked with the user, and an email with a click-through link can be useful for this. Furthermore, if you think of e-mail confirmations as generic click through links, you can use them for other e-mail related tasks such as one-time offers, subscription opt-ins, or event targeted opt-outs that can work without the user having to log in to your site.

The Email Confirmation plugin makes it really easy to do all of these things in a Grails application or plugin.

This release of the book is written for Email Confirmation version 2.0.7 and higher.

What exactly does the plugin do for me?

The Email Confirmation plugin for Grails supports all of these confirmation scenarios by providing:

- A single service method to send a confirmation mail
- Unlimited discrete confirmation requests per user
- Single-use obscured links
- Per-confirmation application data
- Events for confirmation callbacks
- Automatic culling of stale confirmations
- Redirection of the user to any URL after they have confirmed
- Fully customisable e-mail templates
- Testing hooks for getting confirmation URLs without parsing the sent emails

Later in the book we'll see how to use all these features by implementing a simple Grails application for subscribing to a data feed.

It is worth repeating that all of these features are available to plugins as well as applications. As of Email Confirmation 2.0 the use of Platform Core Events API for callbacks means that application and plugins do not have problems co-existing, opening up many possibilities for new plugins that use confirmations.

Installing the plugin

The phrase "install the plugin" is become less appropriate these days with Grails as most people are using `BuildConfig.groovy` to specify their dependencies. Nevertheless you should still be able to use both methods of installing the plugin in Grails versions 1.3.7 through to 2.2.

To install the plugin the recommended way, edit `BuildConfig.groovy` and edit your `plugins` section to include the Email Confirmation plugin:

Adding the plugin to your BuildConfig.groovy

```
plugins {
    //  other existing dependencies
    runtime ":email-confirmation:2.0.8"
}
```

Important: this is the latest version of the plugin at the time of writing. You should check the Grails plugin portal page[1] for the number of the latest release.

Alternatively you can try the old-fashioned way of installing the latest release:

```
> grails install-plugin email-confirmation
```

Note however that you may have more dependency pain this way.

Email Confirmation 2.0 and higher, to which this book is applicable, has a transitive dependency on Platform Core[2] so do not be surprised to see this being pulled into your application. It is of course a great plugin that I heartily recommend you take advantage of if you have not already done so - you can watch me talking about it[3]. Email Confirmation also depends on the Grails Mail plugin for mail sending.

The example application

Through this book we'll work with a single example application. If you want to follow with working examples you'll need to create the application and call it **Subomatic**.

All this simple application will do is take form submissions containing email addresses, store them in a database and update them when the email address is known.

We'll add files to this as we go, and you'll run the app and see the effect of the changes.

[1] http://grails.org/plugin/email-confirmation
[2] http://grailsrocks.github.com/grails-platform-core
[3] http://grck.it/platform-core-talk

First off, as per the previous section, you'll need to add the `email-confirmation` plugin to your `BuildConfig.groovy`.

Next, we need to create a controller called `SubscriptionController` in the `subomatic` package:

Creating the controller

```
package subomatic

class SubscriptionController {
    def index = {
    }

    def subscribe = {
    }
}
```

We'll also need to create a form in a GSP for the `index` action to take peoples' details and submit them to the `subscribe` controller.

Creating the subscribe form

```
<html>
<body>
    <h1>Please enter your e-mail address to subscribe with Subomatic</h1>

    <g:form action="subscribe">
        <label for="email">Your e-mail address:</label>
        <input name="email"/>
        <input type="submit" value="Go Subomatic!"/>
    </g:form>
</body>
</html>
```

Let's also create a simple domain class to store subscriptions, for use later on.

Creating the domain class

```
package subomatic

class Subscription {
    String email
    boolean optedIn

    static constraints = {
        email(email:true)
    }
}
```

With all that out of the way we're ready to start.

2 Concepts and flow

Before we start looking at code and using the plugin, we'll quickly go through the concepts surrounding confirmations, in order to fully understand what the plugin is doing for us.

If you don't care about these issues, you can skip to the next chapter and start coding. Should you do this, you may get some suprises later because you didn't consider all of the details of this non-trivial user experience flow. It's not a lot of reading, and you have been warned.

First of all, we'll look at the requirements and how Email Confirmation meets them. Aside from the basic process of sending an e-mail to a user containing a link, there are some nuances to the process.

Links must be obscured and non-guessable

In many use cases it would be a catastrophe if your confirmation links used a numbered id such as `/confirm/45435`. If you don't know why this is then you're obviously pure of heart. When creating web applications with any kind of security or privacy issues, you actually have to try to put yourself into the mind of somebody a little more evil than yourself.

Take a scenario where you use e-mail confirmations to verify and unlock somebody's account, automatically logging them in to your site after they confirm[1]. If you used a numeric URL for confirmations it would be trivial to write a script that hits your server thousands of times to confirm all pending confirmations, perhaps revealing e-mail address data in the page responses which would be valuable information. It would also mean people could tell roughly how many customers you have.

To this end, the Email Confirmation plugin generates a unique code and encodes it in URL-safe characters. This is stored in the database with the pending confirmation information and used to retrieve that information when the link is clicked.

This means that it would take a lot of CPU and network time to hit a real confirmation using a script.

Single use tokens

The confirmation page that the user is taken to must record the fact they used the confirmation token so that it cannot be re-used. This is important because if a user forgot that they had already clicked the link, they might do so again in the future and this might confuse your application. You'd have to write a lot of defensive code to ensure actions were not performed twice.

[1] The detail of your user flows, particularly around security sensitive actions like sign up and password reset, is critically important. You must spend a lot of time considering the scenarios and "attack vectors" to make sure you do not create a system that has inherent flaws that can be exploited. This, for example, is why most sites that confirm your e-mail address on sign-up do not automatically log you in straight after. This would allow anybody who "sniffs" the network to use your confirmation URL before you do, then log in and change your password or worse.

In addition, it means that bad people cannot profit from packet sniffing or hacking e-mail accounts and using old confirmation e-mails that have not been removed from the user's mail account.

So when the user has successfully confirmed, the pending confirmation is automatically removed from the database.

Tell the user what is happening

The user must see a meaningful web page when they have confirmed. This is a user experience consideration. Now that the user has confirmed, they should be taken to a relevant page in your application. Perhaps this would be their user profile page, or a welcome page.

Whatever it is, this page needs to come from your application. The plugin provides a default page for this, but this is only to make sure it all works out of the box. What you do is return parameters for a redirect to occur after they have confirmed, and the plugin will send the user there for you.

This is great because it means you can change what happens in response to a confirmation depending on the particular user and confirmation type that you received.

Invalid confirmation links must be handled gracefully

Whatever you do this *will* happen sometimes. Confirmation links can be vulnerable to a few problems that cause them to be invalid. Invalid means they are not found in the pending confirmations database.

The possible causes are:

1. The user already redeemed the link previously, or someone else did by stealing their URL
2. The user's e-mail client wrapped the link (contained in a plain text e-mail) in a strange way and failed to send the browser to the full URL
3. The user decided to copy the URL out of a plain text mail and paste it into the browser, and missed the last few characters[2]

So when these things happen, you need to give the user some content that explains some of these issues and who they should contact and what they should do to remedy the situation.

The plugin will generate an event that your application or plugins can receive to indicate where the user should be taken when this happens.

[2] Don't laugh too hard at this one. I dealt with numerous support requests from an unlucky NoticeLocal user who could not confirm their account. She said the URL was too difficult to type in, and as a result kept getting it wrong. Turns out her AOL e-mail client was not detecting URLs correctly and as a result they were not clickable. Yes, the AOL mail client being used in 2012.

Unused confirmations should be trackable

It is often useful to detect un-redeemed confirmations, as this is important information. It can indicate problems with your user experience, ineffective copy writing in your confirmation e-mails, or a problem with the attractiveness of your product.

The plugin has a background job that culls stale confirmations, removing them from the database to ensure your database doesn't grow to more rows than Gangnam Style has had YouTube views. When it finds each of these stale confirmations it will send your application or plugin an event with the information about the lapsed confirmation.

Multiple concurrent confirmations per user

This may not be intuitive but you do need to allow multiple confirmations per user. This is because they may "lose" the original e-mail and request a confirmation again via your application.

It says "lose" above because often it may appear that a confirmation e-mail did not arrive, but actually it was simply stuck in a mail backlog somewhere or in the "wrong" e-mail folder in the recipient's e-mail client and so on. What can then happen is that the user requests a second confirmation and in the meantime the **first** one is found in their inbox and they click the original link. This is why new confirmation requests can't cancel the previous request. In e-mail perhaps more than any other area of applications we have to remember that we are dealing with humans.

The plugin can't reliably reuse the previous confirmation code again because the metadata your application or plugin provides may not match the previous one's data (for good reasons), and it would add unnecessary load by querying the database every time a new confirmation is requested.

Confirmations for different purposes

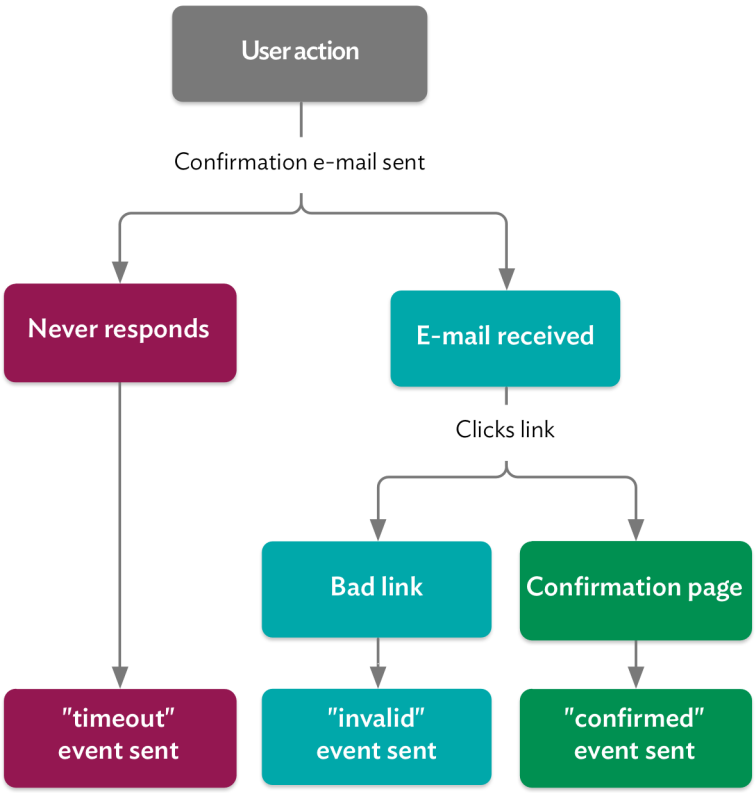
As mentioned earlier, e-mail confirmations can be used for many things. In the most basic cases any sign-up based web application will need user registration confirmation and password reset confirmations. If you are using them for different tasks you don't want to be storing state in your own data to indicate which confirmation is for which task. In fact the plugin is designed to be *opaque* to your application level code. You just make requests for confirmations and receive events back some time later.

As a result, the plugin supports an `id` string that you can provide when requesting a confirmation. It also supports custom event prefixes for callbacks on a *per confirmation* basis so you can get very fine grained control over which code gets the callback for each confirmation. It turns out this is really powerful, and marks version 2.0+ of the plugin from the previous generation in which this was hard if not impossible to achieve. As a result other plugins could not safely hook into confirmation callbacks in the past.

The application-supplied information is stored in the database along with the pending confirmation data.

The flow

Now that we've thought about the various issues, it makes sense to take a look at the flow of interactions that the plugin supports.



Confirmation interaction and event flow

Resources and links

Other documentation

Email Confirmation reference documentation	http://grck.it/email-confirmation-docs
Platform Core documentation	http://grck.it/platform-core-docs
Platform Core cheat sheet PDF	http://grck.it/platform-core-cheat

Grailsrocks plugin news & updates

Twitter	@grails_rocks
Blog	http://grailsrocks.com

Other Grailsrocks plugins

This book is a guide to just one of many Grailsrocks plugins. These may also be useful to you.

Bean Fields Grails Plugin	http://grails.org/plugin/bean-fields
Cache Headers Grails Plugin	http://grails.org/plugin/cache-headers
Cached-Resources Grails Plugin	http://grails.org/plugin/cached-resources
Email-Confirmation Grails Plugin	http://grails.org/plugin/email-confirmation
Feeds Grails Plugin	http://grails.org/plugin/feeds
Functional Test Grails Plugin	http://grails.org/plugin/functional-test
Invitation-Only Grails Plugin	http://grails.org/plugin/invitiation-only
Lamer Filter Grails Plugin	http://grails.org/plugin/lamer-filter
One Time Data Grails Plugin	http://grails.org/plugin/one-time-data
Platform Core Plugin	http://grails.org/plugin/platform-core
Platform UI Plugin	http://grails.org/plugin/platform-ui
Resources framework for Grails	http://grails.org/plugin/resources
Rocks Plugin	http://grails.org/plugin/rocks
Taxonomy Grails Plugin	http://grails.org/plugin/taxonomy
Zipped-Resources Grails Plugin	http://grails.org/plugin/zipped-resources