Spring Security Facebook Plugin - Reference Documentation

Igor Artamonov

Version 0.18

Table of Contents

Chapter 1. Introduction

1.1. Requirements

- Grails 3.0
- spring-security-core plugin 3.0.0

1.2. Links

- Sources - https://github.com/splix/grails-spring-security-facebook
- Sample App - https://github.com/splix/grails-facebook-authentication-example
- if you found a bug - please submit it into https://github.com/splix/grails-spring-security-facebook/issues
- other questions please ask at StackOverflow - http://stackoverflow.com/questions/tagged/grails

Chapter 2. Install plugin

Basically it's just adding dependency into `build.gradle`:

```
dependencies {

    compile 'org.grails.plugins:spring-security-facebook:0.18'

}
```

Follow Configuration and Basic Usage sections for next steps.

2.1. Upgrade Notes

2.1.1. Upgrading from version 0.9

Since version 0.10 plugin have started to use Server Side authentication by default, instead of Client Side authentication (based on Facebook JS SDK) that was default implementation for version 0.9 and earlier.

If you want to continue using Client Side authentication, you should add following configuration into `application.yml`:

```
grails:
    plugin:
        springsecurity:
            facebook:
                filter:
                    type:
                        - transparent
                        - cookieDirect
```

2.1.2. Upgrading from version 0.15.x

Version 0.16 requires a Spring Security Core 2.0, you need to upgrade to this version of core plugin.

2.1.3. Upgrade from version 0.17.x

Version 0.18 is based on Grails 3.0 and Spring Security Core 3.0

Chapter 3. Usage

3.1. Example app

You can take a look at Example Application, it's very basic app, that have only one page, with 'Facebook Connect' button. Just clone it, put your FB App credentials, and play with it.

3.1.1. Add Facebook Authentication into your existing application

Initial plugin config

Domain Class

Create domain class for your facebook user:

```
class FacebookUser {
  Long uid
  String accessToken
  Date accessTokenExpires

  static belongsTo = [user: User] //connected to main Spring Security domain

  static constraints = {
    uid unique: true
  }
}
```

At `conf/application.yml` setup full name (including package name, if used) of just created Facebook user domain, like:

```
grails:
    plugin:
        springsecurity:
            facebook:
                domain:
                    classname: 'FacebookUser'
```

Add FB App credentials

You should create a Facebook App and copy App ID and Secret:

!create_app.png!

into `conf/application.yml`:

```
grails:
    plugin:
        springsecurity:
            facebook:
                appId: 12345678900000
                secret: 76c2279743c99da3715e3d00f29a1234
```

PS it's just example, you should use your own `appId` and `secret`.

Add Facebook Connect button

There is special taglib (`<facebookAuth:`) that can be used at your view (any GSP page) to add Facebook Connect button.

Following code displays connect button for not authorized user, or show a welcome message to logged in user:

```
<sec:ifNotGranted roles="ROLE_USER">
  <facebookAuth:connect />
</sec:ifNotGranted>
<sec:ifAllGranted roles="ROLE_USER">
  Welcome <sec:username/>! (<g:link uri="/j_spring_security_logout">Logout</g:link>)
</sec:ifAllGranted>
```

Run

That's it! Run your application, and test that everything is working.

```
grails run-app
```

3.2. Filters

3.2.1. How it works

Plugin is based on Spring Security and uses web filters for authorization, for more details see Spring Security docs

Available filters

There are 4 types of filter:

- FacebookAuthRedirectFilter - server-side authorization (used by default)
- FacebookAuthCookieTransparentFilter - automatic client-side authorization
- FacebookAuthCookieDirectFilter - manual client-side authorization
- FacebookAuthJsonFilter - for external clients (like Android/iOS app)

Server-Side authentication (FacebookAuthRedirectFilter)

It's a standard Login for Server-side Apps. After clicking on 'connect button' user gets redirected to special Facebook page, for authentication, and then redirected back to your app.

3.2.2. Client-Side authentication

Transparent cookie based authorization (FacebookAuthCookieTransparentFilter)

Based on Facebook Javascript SDK authorization. On client side it makes Facebook authorization and put Facebook Cookie (it's made by Facebook Javascript, you don't need anything special)

After successful authorization on client side, the browser should reload current page. Or open any other page.

This filter will **process each request**, and if it sees valid Facebook cookie, it makes authorization for current user. If it's a new user, it creates a new one for application, with provided Facebook credentials.

It's per-request authorization. That means that this filter will try to authorize user on each page request.

Manual cookie based authentication (FacebookAuthCookieDirectFilter)

Based on Facebook Javascript SDK authorization. On client side it makes Facebook authorization and put Facebook Cookie (it's made by Facebook Javascript, you don't need anything special)

Same as FacebookAuthCookieTransparentFilter, it parse Facebook cookie, but only for specified url. Like username/password filter from spring-security-core or similar. After successful authorization it can redirect user to specified url.

JSON or Android/iOS/desktop authorization (FacebookAuthJsonFilter)

Client should send Access Token or Signed Request as parameter, and will get JSON response with user details.

See [filter docs|guide:3.5 Json Authentication]

3.3. Filter Type configuration

You can use config parameter `grails.plugin.springsecurity.facebook.filter.type` to configure which filters you want to use in your application.

```
It's not a Spring Security configuration, not a configuration for Spring filters. Just
a
extra configuration, that used only by this plugin.
```

By default it uses only one 'redirect' filter:

```
grails:
    plugin:
        springsecurity:
            facebook:
                filter:
                    type: redirect
```

You can use more that one filter at same time:

```
grails:
    plugin:
        springsecurity:
            facebook:
                filter:
                    type:
                        - transparent
                        - cookieDirect
```

Value types: * `redirect` - use standard server side authorization * `transparent` - use transparent cookie based authorization * `cookieDirect` - use manual cookie based authorization * `json` - use JSON authorization

3.4. Service Side

It's the `FacebookAuthRedirectFilter`, enabled by default.

It's preferred and a standard Login for Server-side Apps. After clicking on 'connect button' user gets redirected to special Facebook page, for authentication, and then redirected back to your app.

User going to see Facebook Authentication screen only at the first time. Next time user will be redirected back from Facebook to your application immediately.

3.4.1. How to process failed login

When user declines Facebook Authentication (click Cancel, for example), you'll '401 Authentication Failed' by default. It's default configuration of Spring Security failure handler, but for most cases it's not what you really want.

To handle this situation you have to create your own Failure Handler, a bean implementing `org.springframework.security.web.authentication.AuthenticationFailureHandler`. If you just need to show a page (a GSP view), you can use standard `SimpleUrlAuthenticationFailureHandler`, that could redirect failed authentication to specified URL.

For example you can create bean at `resources.groovy`:

```
import
org.springframework.security.web.authentication.SimpleUrlAuthenticationFailureHandler

// Place your Spring DSL code here
beans = {

    redirectFailureHandlerExample(SimpleUrlAuthenticationFailureHandler) {
        defaultFailureUrl = '/failed' //redirect to this URL when authentication fails
    }

}
```

and setup this bean for 'redirect' filter at `application.yml`:

```
grails:
    plugin:
        springsecurity:
            facebook:
                filter:
                    redirect:
                        failureHandler: redirectFailureHandlerExample
```

Same way for configuring Success Handler.

3.5. Client Side Authentication

Based on Facebook Javascript SDK authorization. Useful when you need to use FB Javascript SDK on client side.

And there are two ways:

- try authenticate user on each request, to any page - it's `transparent` filter
- authenticate only when user redirected to specified page, like a standard username/password authentication - it's `cookieDirect` filter

3.5.1. Transparent filter

If you're using first way (`transparent` filter), your user will be automatically authenticated whenever he has Facebook cookie. Btw, don't forget that you should **reload** current page after you have successfully authenticated user on client side. Like:

```
<facebookAuth:init>
   FB.Event.subscribe('auth.login', function() {
      window.location.reload();
   });
</facebookAuth:init>
```

To logout user, simply call `FB.logout()` (using Javascript) on client side.

3.5.2. CookieDirect filter

If you're using second way (`cookieDirect` filter), you could configure URL that will be used for authentication at `application.yml`:

```
grails:
    plugin:
        springsecurity:
            facebook:
                filter:
                    processUrl: '/j_spring_security_facebook_check' # it's default
value
```

And after authorization redirect user to `/j_spring_security_facebook_check`, like:

```
<facebookAuth:init>
   FB.Event.subscribe('auth.login', function() {
      window.location.href = '/j_spring_security_facebook_check'
   });
</facebookAuth:init>

<g:javascript>
   $('#fbloginbutton').click(function() {
       FB.login();
   });
</g:javascript>
```

3.6. Json Authentication

Filter 'FacebookAuthJsonFilter' accepts Facebook Access Token or Signed Request as parameter, and responds with JSON to authorization requests. It's useful if you an external client for your Grails application, it could be Android or iOS application, or Desktop application, or just AJAX client.

JSON filter just returns an object with user details, nothing else. For authentication of other requests, you still have to use different filter. If you have a RESTful client, take a look at spring-security-oauth2-provider plugin

How it works:

```
> GET /j_spring_security_facebook_json?access_token=<ACCESS_TOKEN>
```

For successful authorization you'll get:

```
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
<
{
  "authenticated": true,
  "uid": 12345612345,   # Facebook User Id
  "roles":[
     "ROLE_FACEBOOK",
     "ROLE_USER"
  ],
  "username": "facebook_12345612345", # Grails Application User Id/Username
  "enabled": true # Grails Application User status
}
```

For unsuccessful:

```
< HTTP/1.1 401 Unauthorized
< Content-Type: application/json;charset=UTF-8
<
{
  "authenticated": false,
  "message": "Expired token" # Authentication Failure reason
}
```

3.6.1. How to extend JSON response

The plugin going to call `Map onJsonSuccess(Map input, FacebookAuthToken token)` or `Map onJsonFailure(Map input, AuthenticationException exception)` methods of FacebookAuthService (if exists).

There you can update `input` data with any other values, introduce new fields/keys, or even return your own structure. This structure will be transformed to JSON and sent to client.

Chapter 4. Configuration

4.1. Basic Configuration

Make sure that you have installed and configured spring-security-core plugin before this step.

Most default configuration will look like:

```
grails:
    plugin:
        springsecurity:
            facebook:
                domain:
                    classname: '<your FacebookUser domain>'
                secret: '<Facebook secret for your app>'
                appId: '<Facebooks's app ID>'
```

When you have valid configuration you can put Facebook Connect button in you GSP:

```
<facebookAuth:connect permissions="email,user_about_me"/>
```

You don't need to add anything else.

4.2. Configure Facebook App

Name	Default Value
grails.plugin.springsecurity.facebook.secret	must be specified
grails.plugin.springsecurity.facebook.appId	must be specified
grails.plugin.springsecurity.facebook.apiVersion	not set

- `apiVersion` - Facebook API version (e.g., "v2.2"). If not set is used unversioned Facebook API by default.

Name	Default Value
grails.plugin.springsecurity.facebook.permissions	['email']

For a list of all possible permissions see https://developers.facebook.com/docs/reference/login/#permissions

4.3. Configure Domains

Name	Default Value	Values
grails.plugin.springsecurity.facebook.domain.classname	'FacebookUser'	
grails.plugin.springsecurity.facebook.domain.appUserConnectionPropertyName	'user'	

- `domain.classname` - name of your domain class, used to store Facebook User details (uid, access_token, etc). Could be same as configured for Spring Security Core, or a own domain, just for Facebook User details.

If you're using own domain for Facebook user (a joined to main User) you should specify `domain.appUserConnectionPropertyName`: it's how your domain class is related to main (used by Spring Security Core) user domain. It's the name of the property, usually defined as `static belongsTo = [user: User]` in your Facebook User domain class.

4.3.1. User creation/initialization

Name	Default Value
grails.plugin.springsecurity.facebook.autoCreate.roles	['ROLE_USER', 'ROLE_FACEBOOK']

List of roles for user created by the plugin.

4.4. Configure Login Button

4.4.1. Button configuration

Name	Default Value
grails.plugin.springsecurity.facebook.taglib.button.text	'Login with Facebook'

4.4.2. Button for Server Side authentication (default)

Standard `<img ⋯ />` will be used for button, with following extra configuration options:

Name	Default Value
grails.plugin.springsecurity.facebook.taglib.button.img	an url for image distributed with plugin

- `img` - url of a default image to use for button

4.4.3. Button for Client Side authentication

At this case a HTML button, provided by Facebook JS SDK, will be user.

Name	Default Value
grails.plugin.springsecurity.facebook.taglib.language	'en_US'

- `language` - language for Facebook Javascript SDK. You could also pass this option as a `lang` attribute for `:connect` or `:init` tags

4.5. Configure Plugin

Name	Default Value
grails.plugin.springsecurity.facebook.autoCreate.enabled	true
grails.plugin.springsecurity.facebook.autoCreate.roles	['ROLE_USER', 'ROLE_FACEBOOK']

- `autoCreate.enabled` - enable/disabled automatic creation of Application User for a new Facebook user (when FB user first time authenticates)
- `autoCreate.roles` - list of roles to set to a newly created user (if enabled)

Name	Default Value
grails.plugin.springsecurity.facebook.host	''

Set a hostname of current app, could be used when user logged out, but FB didn't clear all cookies for domain. Note: it's host name, not url. Like `example.com`

4.6. Configure Authentication Types

Name	Default Value
grails.plugin.springsecurity.facebook.filter.processUrl	'/j_spring_security_facebook_check'
grails.plugin.springsecurity.facebook.filter.type	'redirect'

- `type` - type of authentication, can be `transparent`, `cookieDirect`, `redirect` or `json`.

You can specify list of filters as a list `['redirect', 'json']` or comma-separated string:

```
grails.plugin.springsecurity.facebook.filter.type='redirect,json'
```

4.6.1. Configuration for REDIRECT filter

Name	Default Value
grails.plugin.springsecurity.facebook.filter.redirect.processUrl	not set
grails.plugin.springsecurity.facebook.filter.redirect.redirectFromUrl	'/j_spring_security_facebook_redirect'
grails.plugin.springsecurity.facebook.filter.redirect.failureHandler	not set
grails.plugin.springsecurity.facebook.filter.redirect.successHandler	not set

- `redirectFromUrl` - it's the url that will redirect user to special Facebook Authentication URL. You can put link to/redirect user to `redirectFromUrl` when you want to use Facebook authentication. This url is used by default `<facebook:connect />` button.
- if `filter.redirect.processUrl` is not then, then default `filter.processUrl` will be used
- `failureHandler` - a name of the bean that implements [AuthenticationFailureHandler|http://static.springsource.org/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/web/authentication/AuthenticationFailureHandler.html]
- `successHandler` - a name of the bean that implements [AuthenticationSuccessHandler|http://static.springsource.org/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/web/authentication/AuthenticationSuccessHandler.html]

4.6.2. Configuration for TRANSPARENT filter

NA

4.6.3. Configuration for COOKIEDIRECT filter

Name	Default Value
grails.plugin.springsecurity.facebook.filter.cookieDirect.processUrl	not set
grails.plugin.springsecurity.facebook.filter.cookieDirect.failureHandler	not set
grails.plugin.springsecurity.facebook.filter.cookieDirect.successHandler	not set

- if `filter.cookieDirect.processUrl` is not set, then default `filter.processUrl` will be used
- `failureHandler` - a name of the bean that implements [AuthenticationFailureHandler|http://static.springsource.org/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/web/authentication/Authenticatio

nFailureHandler.html]

- `successHandler` - a name of the bean that implements [AuthenticationSuccessHandler|http://static.springsource.org/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/web/authentication/AuthenticationSuccessHandler.html]

4.6.4. Configuration for JSON filter

Name	Default Value
grails.plugin.springsecurity.facebook.filter.json.processUrl	'/j_spring_security_facebook_json'
grails.plugin.springsecurity.facebook.filter.json.type	'json'
grails.plugin.springsecurity.facebook.filter.json.methods	['POST']

- `type` could be `json` (default) or `jsonp`
- `methods` - allowed HTTP methods. Notice that it's used only for JSON, for JSONP it will be forced to 'GET'

Chapter 5. Customization

5.1. How it works

If you need to add some specific logic to default plugin behaviour you have to create your own service called `FacebookAuthService`. Plugin will check for know methods of this service, and if they're exist - use them instead of own.

It's some kind of extending an abstract class. You don't need to create all methods, just what you need.

Used objects:

- <FacebookUser> - domain class for your facebook user. It's your own class, can have other name, it's just a example
- <Person> - general user, used by Spring Security. It's your own class, can have other name, it's just a example
- FacebookAuthToken - token provided by plugin

<FacebookUser> and <Person> can be same object, or can be two different object (with a relation), depends on your architecture.

5.2. Default Implementation

Please, take a look at sources of [DefaultFacebookAuthDao|https://github.com/splix/grails-spring-security-facebook/blob/master/src/groovy/com/the6hours/grails/springsecurity/facebook/DefaultFacebookAuthDao.groovy] to understand how it works, and which methods you can use for customization

5.2.1. List of possible methods:

void onCreate(<FacebookUser> user, FacebookAuthToken token)

Called after user was created by plugin, just before saving into database. You can fill user object with some extra values.

Not called if you have implemented method @create(..)@

Where:

- user - your domain for Facebook User
- token - com.the6hours.grails.springsecurity.facebook.FacebookAuthToken

void afterCreate(<FacebookUser> user, FacebookAuthToken token)

Called after user was created by plugin, and when it's already saved into database.

Not called if you have implemented method @create(..)@

Where:

- user - your domain for Facebook User
- token - com.the6hours.grails.springsecurity.facebook.FacebookAuthToken

<FacebookUser> findUser(long uid)

Called when facebook user is authenticated (on every request), must return existing instance for specified facebook uid, if exits. If doesn't - return null

Where:

- uid - facebook user id

<FacebookUser> create(FacebookAuthToken token)

Called when we have a new facebook user, called on first login to create all required data structures.

Where:

- token - com.the6hours.grails.springsecurity.facebook.FacebookAuthToken

Notice, that if you have such method, all other method for user creation will no be called:

- createAppUser
- prepopulateAppUser
- onCreate
- afterCreate
- createRoles

<Person> createAppUser(<FacebookUser> user, FacebookAuthToken token)

Called when we have a new facebook user, called on first login to create main app User domain (when we store Facebook User details in different domain).

Not called if you have implemented method @create(..)@

Where:

- user - your domain for Facebook User
- token - com.the6hours.grails.springsecurity.facebook.FacebookAuthToken

void createRoles(<FacebookUser> user)

Called when we have a new facebook user, called on first login to create roles list for new user

Where:

- user - your domain for Facebook User

def getPrincipal(<FacebookUser> user)

Must return object to store in security context for specified facebook user (can return itself)

Where:

- user - your domain for Facebook User

<FacebookUser> getFacebookUser(<Person> person)

Must return instance of your domain object for facebook user for specified person (if it's not a same object)

Where:

- person - your domain for <Person>

Collection<GrantedAuthority> getRoles(<Person> user)

Must return roles list for specified user

Where:

- user - your domain for Facebook User

void prepopulateAppUser(<Person> person, FacebookAuthToken token)

Must return roles list for specified facebook user

Where:

- person - your domain for <Person>
- token - com.the6hours.grails.springsecurity.facebook.FacebookAuthToken

Chapter 6. How To

6.1. How To

6.1.1. How to get user full name and/or email?

Main goal of the plugin is to make authorization. All other usage of Facebook API should be done by using additional library, Spring Social for example.

First of all: you need 'email' permission on connect `<facebookAuth:connect permissions="email"/>`

Add Spring Social lib into your classpath, by adding following dependencies into your `build.gradle`:

```
compile 'org.springframework.social:spring-social-facebook:2.0.3.RELEASE'
```

and then you can use Facebook API. For example you can fetch user email and full name on user creation step:

```
def facebook = new FacebookTemplate(token.accessToken.accessToken)
def fbProfile = facebook.userOperations().userProfile
String email = fbProfile.email
String name = fbProfile.name
```

See documentations for Spring Social Facebook: http://docs.spring.io/spring-social-facebook/docs/2.0.3.RELEASE/reference/htmlsingle/#retrieving-a-user-s-profile-data

Chapter 7. Common Issues

7.1. Debug

7.1.1. Enable logging

If you have troubles with plugin, please enabled logging, so you can see what's happening. Add to `logback.groovy` following:

```
logger("com.the6hours", DEBUG, ["CONSOLE"])
```

7.2. Issues

7.2.1. Client side authentication don't work on dev server

Make sure that you're using a real domain name for your application. Not a `localhost`, because Facebook can't setup cookie for localhost, and avoid .local domains as well.

You can make a fake domain like myapp.dev, by putting into `/etc/hosts` the following line:

```
127.0.0.1  myapp.dev
```

If you already have line starting with `127.0.0.1`, just add your `myapp.dev` at the end of the line.

See more details about hosts file, and location of the file for different operation systems see: http://en.wikipedia.org/wiki/Hosts_(file)

After that, you should configure your Grails app to use this domain, by adding following line into `conf/application.yml`:

```
grails:
    serverURL: "http://myapp.dev:8080/${appName}"
```

Of course, you need to use this domain only for development, so put this configuration into `development` environment config:

```
environments:
    development:
        grails:
            serverURL: "http://myapp.dev:8080/${appName}"
```

7.2.2. Logout doesn’t work for Transparent filter

"transparent" filter always authorize request that contains FB cookie. If you need to log out current user when using this type of filter, you need to log out user on client side, call `FB.logout()` (will logout user from Facebook as well) and reload the page.