Demystifying Spring Security in Grails

Burt Beckwith

Spring Security History

- Initial work was started by Ben Alex in 2003 and was officially created as a SourceForge project in 2004, then known as Acegi Security
- Quickly became the de-facto security framework for Spring and became a Spring subproject around 2005
- Version 1.0 was released in 2006
- In 2007 it was renamed Spring Security, and version 2.0 was released in 2008

Some Definitions

Principal

- An application user, although not necessarily a person, i.e. could be an external system

Authentication

- The process of verifying that a requesting user is a valid user

Authorization

- The process of validating that a principal has permission to perform an action

Some Definitions

Credential

- Information used to verify a principal's identity, typically a password

Authority

- Information associated with an authenticated user to indicate a permission, role, etc.

Other Options

The framework formerly known as JSecurity

- A fine security framework, significant overlap with Spring Security's features

Use Spring Security directly

Roll your own

- Saw a post on some dude's blog, just a simple filter, way simpler than Acegi!
- Dangerous: any time you think you're being slick with security you put your reputation and your users' data at risk

Use a proven framework

Any time you think you're being slick with security you put your reputation and your users' data at risk

Spring Security Overview

- Configurable filter chain registered in web.xml
- Authenticators
  - DAO, LDAP, OpenID, more
- URL, method, and object security
- Interface-based, very pluggable

Grails Acegi Plugin

- Originally written by Tsuyoshi Yamamoto, based on work done for Acegi on Grails in 2006
- Became the Acegi Plugin in 2007
- I joined the team in April 2008, contributed fixes and rewrote to use Spring Security 2
- Current version is 0.5.2, 0.6 being tested for release
- Works in all versions of Grails
- Docs and tutorials at http://grails.org/plugin/acegi

Plugin Features

- Convention over configuration, with centralized configuration in SecurityConfig.groovy
- Highly configurable and customizable
- Optional basic CRUD user interface
- Password encryption
- “Remember me” cookie
- IP Address <=> URL restrictions
- Convenient event handlers
- Run as other user

Plugin Features

Multiple authentication providers

- Form-based
- HTTP Basic
- OpenID
- Facebook Connect
- Kerberos
- NTLM
- LDAP
- External (Pre-auth)
- SSO (CAS)
- Browser certificate

Plugin Features

- Registers Spring Security beans in application context, filters in web.xml
- AuthenticateService
  - IfAllGranted(), encodePassword(), isLoggedIn(), etc.
- AuthorizeTagLib
  - <g:ifAllGranted/>, <g:ifNotGranted/>, <g:isLoggedIn/>, etc.

Plugin Features

- Primary focus of the plugin is URL authentication and authorization
- Work is currently being done to expose Spring Security ACL support in Grails

URL Security

- Three declaration styles
  - Annotations
  - Requestmap (stored in database, configurable at runtime)
  - SecurityConfig string (old style)

Installation and Setup

`grails install-plugin acegi`

- Installs the latest version of the plugin

`grails create-auth-domains User Role Requestmap`

- Creates SecurityConfig.groovy, 3 domain classes, and Login & Logout controller

`grails generate-manager` (optional)

- Creates CRUD controllers and GSPs for the domain classes

`grails generate-registration` (optional)

- Creates UI for user registration

Form Login Demo

BASIC Auth Demo

LDAP Demo

Security Filters

ChannelProcessingFilter

- Optional
- Redirects HTTP requests to HTTPS and vice-versa based on URL rules

ConcurrentSessionFilter

- Not (yet) implemented in plugin
- Restricts the number of concurrent logins per user

Security Filters

HttpSessionContextIntegrationFilter

- uses session Authentication to populate SecurityContext with an Authentication in SecurityContextHolder

LogoutFilter

- intercepts calls to *j_spring_security_logout*, delegates to list of LogoutHandlers to perform logout work, redirects to configurable post-logout url

LogoutFilter LogoutHandlers

SecurityContextLogoutHandler

- invalidates the HTTP session (configurable) and clears the SecurityContextHolder

TokenBasedRememberMeServices

- cancels the remember-me cookie

FacebookLogoutHandler (if using Facebook)

- cancels Facebook cookies

Security Filters

IpAddressFilter

- Optional
- Plugin-only, not part of Spring Security
- define IP patterns for URL patterns to block access by IP
- useful for admin – e.g. restrict to 10.** or 192.168.**

Security Filters

X509PreAuthenticatedProcessingFilter

- Authentication via browser certificate

CasProcessingFilter

- Integration with CAS SSO

AuthenticationProcessingFilter

- intercepts *j_spring_security_check* to process form-based logins

Security Filters

OpenIDAuthenticationProcessingFilter

- intercepts *j_spring_openid_security_check* to handle OpenID logins and manage redirecting to external OpenID authentication provider

FacebookAuthenticationProcessingFilter

- Plugin-only, not part of Spring Security
- intercepts *j_spring_facebook_security_check* to handle Facebook logins and manage redirecting to Facebook for authentication

Security Filters

RememberMeProcessingFilter

- uses a RememberMeServices (typically TokenBasedRememberMeServices) to auto-login from remember-me cookie

AnonymousProcessingFilter

- populates SecurityContextHolder with an AnonymousAuthenticationToken if not authenticated
- useful to guarantee that there's always an auth, avoids null checks

Security Filters

BasicProcessingFilter

- Optional
- supports HTTP BASIC auth
- Often used with remoting

DigestProcessingFilter

- Not (yet) supported in plugin
- Supports Digest authentication, similar to BASIC auth but more secure
- Often used with remoting

Security Filters

SecurityContextHolderAwareRequestFilter

- replaces the Request with a wrapper, typically SavedRequestAwareWrapper
- overrides getRemoteUser() returning the authenticated user's username
- overrides getUserPrincipal() returning the authenticated user's Authentication/Principal
- overrides isUserInRole() checking the authenticated user's Authentication for matching Authority

Security Filters

ExceptionTranslationFilter

- catches AccessDeniedExceptions and AuthenticationExceptions
- allows a complex chain of filters to break out at any point to common handler
- resets any current authentication, creates and registers a SavedRequest, delegates to AuthenticationProcessingFilterEntryPoint (in the plugin a custom Ajax-aware subclass) to redirect to login form

Security Filters

NtlmProcessingFilter

- Optional
- Supports Windows NTLM authentication

FilterSecurityInterceptor

- uses URL <=> Role rules to determine if access is allowed to URLs

SwitchUserProcessingFilter

- Optional
- Allows one user (e.g. an admin) to assume the authentication of another temporarily

Authentication

- ProviderManager has list of AuthenticationProviders that attempt to authenticate from provided credentials in order
- A filter creates an Authentication, e.g.UsernamePasswordAuthenticationToken, FacebookAuthenticationToken, OpenIDAuthenticationToken from the Request
- For form-based logins, DaoAuthenticationProvider is standard, uses database for users and roles

DaoAuthenticationProvider

- Depends primarily on PasswordEncoder and UserDetailsService (GrailsDaoImpl)
- UserDetailsService loads user by username or throws UsernameNotFoundException
- PasswordEncoder checks that encrypted version of supplied cleartext password matches encrypted password from database
- Checks that account isn't expired/locked/disabled

UserDetailsService

- Single-method interface

```
UserDetails loadUserByUsername(String username);
```

- Plugin implementation is GrailsDaoImpl
- Easy to implement your own, see Custom UserDetailsService

Spring Bean Customization

- Two places: resources.groovy or BootStrap.groovy
- Replace beans with custom versions in resources.groovy for most flexibility
- If changes are smaller then tweak properties of existing beans in BootStrap
- Check first that there isn't already a configuration option in DefaultSecurityConfig.groovy

Coming in 0.6+

- Salted passwords
- Session hijacking protection
- Nearly 100% Java for performance
- More configuration hooks
- ACLs
- New modular plugin for Spring 3.0/Spring Security 3.0

Q&A