A (re)Introduction to Spring Security

Agenda

- Before Spring Security: Acegi security
- Introducing Spring Security
- View layer security
- What's coming in Spring Security 3

SPRING-LOADED

Before Spring Security There was...

Acegi Security for Spring

- Created by Ben Alex in 2003
  - 1.0 released in March 2004
- Applies security rules using Servlet Filters and Spring AOP
- Extremely powerful and flexible

Spring-Loaded

What Acegi Offered

- Declarative Security
  - Keeps security details out of your code
- Authentication and Authorization
  - Against virtually any user store
- Support for anonymous sessions, concurrent sessions, remember-me, channel-enforcement, and much more
- Spring-based, but can be used for non-Spring web frameworks

SPRING-LOADED

The Downside of Acegi

"Every time you use Acegi...A fairy dies."

– Daniel Deiphouse

http://netzooid.com/blog/2007/12/03/every-time-you-use-acegi/

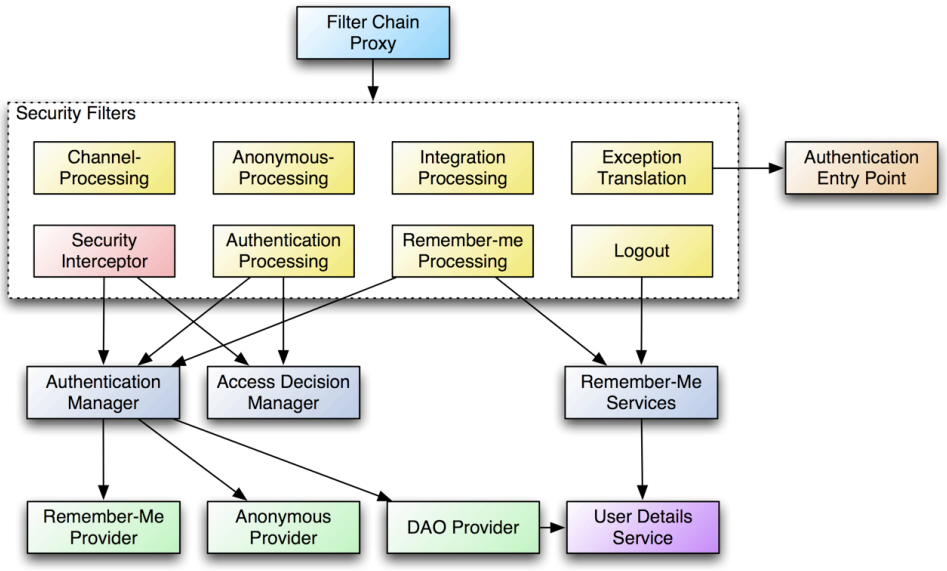
SPRING-LOADED

Example Acegi Config

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <bean id="filterChainProxy"
      class="org.acegisecurity.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /**=channelProcessingFilter,httpSessionIntegrationFilter,
        logoutFilter,authenticationProcessingFilter,rememberMeProcessingFilter,
        anonymousProcessingFilter,exceptionTranslationFilter,filterSecurityInterceptor
      </value>
    </property>
  </bean>
  <bean id="authenticationProcessingFilter"
      class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
    <property name="authenticationManager" ref="authenticationManager"/>
    <property name="authenticationFailureUrl" value="/login.htm?login_error=1" />
    <property name="defaultTargetUrl" value="/" />
    <property name="filterProcessesUrl" value="/j_acegi_security_check" />
    <property name="rememberMeServices" ref="rememberMeServices" />
  </bean>
  <bean id="authenticationManager"
      class="org.acegisecurity.providers.ProviderManager">
    <property name="providers">
      <list>
        <ref bean="daoAuthenticationProvider" />
        <ref bean="anonymousAuthenticationProvider" />
        <ref bean="rememberMeAuthenticationProvider" />
      </list>
    </property>
  </bean>
  <bean id="daoAuthenticationProvider"
      class="org.acegisecurity.providers.dao.DaoAuthenticationProvider">
    <property name="userDetailsService"
        ref="userDetailsService" />
  </bean>
  <bean id="userDetailsService"
      class="org.acegisecurity.userdetails.jdbc.JdbcDaoImpl">
    <property name="dataSource" ref="dataSource" />
    <property name="usersByUsernameQuery"
        value="SELECT email as username, password, 'true' FROM Motorist WHERE email=?" />
    <property name="authoritiesByUsernameQuery"
        value="SELECT email as username, privilege FROM Motorist_Privileges mp, Motorist m WHERE
mp.motorist_id = m.id AND m.email=?" />
  </bean>
  <bean id="authenticationEntryPoint"
      class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilterEntryPoint">
    <property name="loginFormUrl" value="/login.htm" />
    <property name="forceHttps" value="true" />
  </bean>
  <bean id="accessDecisionManager"
      class="org.acegisecurity.vote.UnanimousBased">
    <property name="allowIfAllAbstainDecisions" value="false" />
    <property name="decisionVoters">
      <list>
        <bean class="org.acegisecurity.vote.RoleVoter" />
      </list>
    </property>
  </bean>
  <bean id="exceptionTranslationFilter"
      class="org.acegisecurity.ui.ExceptionTranslationFilter">
    <property name="authenticationEntryPoint"
        ref="authenticationEntryPoint" />
    <property name="accessDeniedHandler">
      <bean class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
        <property name="errorPage" value="/error.htm" />
      </bean>
    </property>
  </bean>
  <bean id="httpSessionIntegrationFilter"
      class="org.acegisecurity.context.HttpSessionContextIntegrationFilter">
    <property name="forceEagerSessionCreation" value="true" />
  </bean>
  <bean id="filterSecurityInterceptor"
      class="org.acegisecurity.intercept.web.FilterSecurityInterceptor">
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="accessDecisionManager" ref="accessDecisionManager" />
    <property name="objectDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /booger.htm=ROLE_BODGER
      </value>
    </property>
  </bean>
  <bean id="anonymousProcessingFilter"
      class="org.acegisecurity.providers.anonymous.AnonymousProcessingFilter">
    <property name="key" value="foobar" />
    <property name="userAttribute" value="anonymousUser,ROLE_ANONYMOUS" />
  </bean>
  <bean id="anonymousAuthenticationProvider"
      class="org.acegisecurity.providers.anonymous.AnonymousAuthenticationProvider">
    <property name="key" value="foobar" />
  </bean>
        <bean id="rememberMeProcessingFilter"
            class="org.acegisecurity.ui.rememberme.RememberMeProcessingFilter">
          <property name="rememberMeServices" ref="rememberMeServices" />
          <property name="authenticationManager" ref="authenticationManager" />
        </bean>
        <bean id="rememberMeServices"
            class="org.acegisecurity.ui.rememberme.TokenBasedRememberMeServices">
          <property name="userDetailsService" ref="userDetailsService" />
          <property name="key" value="roadRantz" />
        </bean>
        <bean id="rememberMeAuthenticationProvider"
            class="org.acegisecurity.providers.rememberme.RememberMeAuthenticationProvider">
          <property name="key" value="roadRantz" />
        </bean>
  <bean id="logoutFilter"
      class="org.acegisecurity.ui.logout.LogoutFilter">
    <constructor-arg value="/home.htm" />
    <constructor-arg>
      <list>
        <ref bean="rememberMeServices"/>
        <bean class="org.acegisecurity.ui.logout.SecurityContextLogoutHandler"/>
      </list>
    </constructor-arg>
  </bean>
  <bean id="channelProcessingFilter"
      class="org.acegisecurity.securechannel.ChannelProcessingFilter">
    <property name="channelDecisionManager" ref="channelDecisionManager" />
    <property name="filterInvocationDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /login.htm=REQUIRES_SECURE_CHANNEL
        /j_acegi_security_check*=REQUIRES_SECURE_CHANNEL
        /**=REQUIRES_INSECURE_CHANNEL
      </value>
    </property>
  </bean>
</beans>
```


SPRING-LOADED

What was in that XML?



E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

SPRING-LOADED

Introducing Spring Security

Solution: Spring Security

- All of the same goodness of Acegi
  - Plus some new stuff
- Provides a new security namespace for Spring
  - Much less XML
- Based on Spring, but can be used with non-Spring applications
- Currently at version 2.0.5
  - Version 3.0.0.RC1 is available

Spring-Loaded

From the home page

"Spring Security is a powerful, flexible security solution for enterprise software, with a particular emphasis on applications that use Spring."

What Spring Security Isn't

- Firewall or proxy server
- OS-level security
- JVM security
- Identity management or single-sign-on
- Protection against cross-site scripting

Features

- Authentication
- Web URL and method authorization
- Channel (HTTP/HTTPS) security
- Domain based security (ACLs)
- Also plays well with other Spring components
  - WSS/WS-Security with Spring-WS
  - Flow authorization with Spring WebFlow
  - Uses Spring 3's SpEL

Key concepts

- Filters
- Authentication
- Repositories
- Web authorization
- Method authorization

DelegatingFilterProxy

In WEB-INF/web.xml:

```
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>

<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Proxies requests to a bean with ID "springSecurityFilterChain"

Authentication

- Several choices
  - Form
  - Basic
  - LDAP
  - Kerberos
  - Container (eg. Tomcat)
  - JAAS
  - JA-SIG CAS
  - OpenID
  - SiteMinder
  - Atlassian Crowd
  - OpenID
  - X.509
  - Digest

Simpler Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
   xmlns:beans="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
      http://www.springframework.org/schema/security
      http://www.springframework.org/schema/security/spring-security-2.0.xsd">

   <http auto-config="true">
      <intercept-url pattern="/addRant.htm" access="ROLE_MOTORIST" />
      <intercept-url pattern="/home.htm" requires-channel="http" />
      <intercept-url pattern="/login.htm" requires-channel="https" />
      <form-login login-page="/login.htm" />
   </http>

   <authentication-provider user-service-ref="userService" />

   <jdbc-user-service id="userService" data-source-ref="dataSource" />
</beans:beans>
```

SPRING-LOADED

<http>: The magic element

- The central configuration element for web security
- <intercept-url> declares a page to be secured (and how it should be secured)
- <form-login> refers to a login page
- The auto-config attribute automatically configures support HTTP Basic authentication, Logout, Remember-Me, and Anonymous sessions
  - In fact, it also automatically creates a login page for you

SPRING-LOADED

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

More on <http>

- May also contain...
  - <access-denied-handler>
  - <anonymous>
  - <concurrency-control>
  - <form-login>
  - <http-basic>
  - <intercept-url>
  - <logout>
  - <openid-login>
  - <port-mappings>
  - <remember-me>
  - <session-management>
  - <x509>

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

Even more on <http>

- Has these attributes
  - servlet-api-provision
  - path-type
  - lowercase-comparisons
  - realm
  - entry-point-ref
  - access-decision-manager-ref
  - access-denied-page
  - once-per-request
  - create-session

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

<AUTHENTICATION-PROVIDER>

- Declares an authentication provider
- Refers to a user details service
  - Optionally may contain a user details service:

```
<authentication-provider>
  <jdbc-user-service data-source-ref="dataSource" />
</authentication-provider>
```

- Declare as many providers as you need

SPRING-LOADED

ABOUT <JDBC-USER-SERVICE>

- Defaults to specific SQL
  - User details:
    - SELECT username,password,enabled FROM users WHERE username=?
  - User privileges:
    - SELECT username,authority FROM authorities WHERE username=?
- Can be overridden...

```
<authentication-provider>
  <jdbc-user-service data-source-ref="dataSource"
      users-by-username-query=
         "select username, password, true FROM spitter WHERE  username=?"
      authorities-by-username-query=
         "select username,authority FROM spitter_privileges WHERE username=?" />
</authentication-provider>
```

SPRING-LOADED

Securing methods

- Two ways...
  - Intercept methods

```
<beans:bean id="userService" class="com.habuma.user.UserAdminServiceImpl"/>
  <intercept-methods access-decision-manager-ref="accessDecisionManager">
    <protect method="addUser" access="ROLE_ADMIN"/>
  </intercept-methods>
</beans:bean>
```

  - Annotation-driven
    - Using @Secured

```
<global-method-security secured-annotations="enabled" />
```

    - Using JSR-250 annotations (e.g., @RolesAllowed)

```
<global-method-security jsr250-annotations="enabled" />
```

SPRING-LOADED

JSR-250

```
@DenyAll
public class Bank {
  @RolesAllowed("ROLE_TELLER")
  void deposit(Account account, float amount) {
    //...
  }
}
```

SPRING-LOADED

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

@Secured

```
public class Bank {
  @Secured("ROLE_TELLER")
  void deposit(Account account, float amount) {
    //...
  }
}
```


E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

View layer security

Spring Security JSP tags

- Controls what gets rendered
- Includes...
  - <security:authorize>
  - <security:authentication>
  - <security:accesscontrollist>
- For you Velocity fans...
  - $authz

SPRING-LOADED

JSP tag example

```
<%@ taglib prefix="security"
    uri="http://www.springframework.org/security/tags" %>

...

<security:authorize ifAnyGranted="ROLE_ANONYMOUS">
   <p>Please login:</p>
...
</security:authorize>

<security:authorize ifNoneGranted="ROLE_ANONYMOUS">
<p>Welcome, <security:authentication
                          property="principal.username"/>!</p>
</security:authorize>
```

SPRING-LOADED

What's new in Spring Security 3

Expression-based security

- Uses Spring Expression Language SpEL
- Flexible security rules
- Can be used to define authorization rules for web requests and methods

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

Expression elements

- hasRole(String)
- hasAnyRole(String)
- hasIpAddress("192.168.1.2/24")
- hasPermission(String)
- isAnonymous
- isRememberMe
- isFullyAuthenticated
- authentication
- permitAll, denyAll
- access to method args and return objects

SPRING-LOADED

Expressions & web security

```
<http use-expressions="true">
   <intercept-url pattern="/secure/**"
     access="hasRole('ROLE_SUPERVISOR')
        and hasIpAddress('192.168.1.2')" />
   ...
</http>
```

SPRING-LOADED

E-mail: craig@habuma.com Blog: http://www.springloaded.info Source Code: svn://svn.geekisp.com/SiA svn://svn.geekisp.com/habuma

Pre- and Post- annotations

- Four new annotations...
  - @PreAuthorize – Permits access if expression evaluates to true
  - @PostFilter – Filters a collection return value according to expression evaluation
  - @PreFilter – Filters collection method arguments according to expression evaluation
  - @PostAuthorize – Restricts access to a method's return value

SPRING-LOADED

@PreAuthorize

Allow method access if user has "ROLE_USER" role

```
@PreAuthorize("hasRole('ROLE_USER')")
public void create(Contact contact);
```

Allow method access if user has "admin" permission on the contact object

```
@PreAuthorize("hasPermission(#contact, 'admin')")
public void deletePermission(Contact contact, Sid recipient,
Permission permission);
```

Allow method access if the user has "ROLE_TELLER" role and if the deposit will reconcile overdraft

```
@PreAuthorize("hasRole('ROLE_TELLER') and
    (#account.balance + #amount >= -#account.overdraft)")
void deposit(Account account, double amount) {...}
```

SPRING-LOADED

@PostFilter

Allow access if the user has "ROLE_USER" role. Filter the list to include only those objects for which user has "read" or "admin" permission.

```
@PreAuthorize("hasRole('ROLE_USER')")
@PostFilter("hasPermission(filterObject, 'read') or
             hasPermission(filterObject, 'admin')")
public List getAll();
```

SPRING-LOADED

Restructuring

- Historically, most of Spring Security contained in a single JAR
- Some split packages...not OSGi-friendly
- Spring Security 3, split across ~7 JAR files
- More modular...and OSGi-friendly

SPRING-LOADED

E-MAIL: CRAIG@HABUMA.COM BLOG: HTTP://WWW.SPRINGLOADED.INFO SOURCE CODE: SVN://SVN.GEEKISP.COM/SIA SVN://SVN.GEEKISP.COM/HABUMA

Summary

Final thoughts

- Spring Security picks up where Acegi left off
- Extremely powerful and flexible security framework
- Spring-based, but can be used to secure non-Spring apps

Spring-Loaded