Spring Security

[Spring]

- http://static.springsource.org/spring-security/site/docs/3.0.x/reference/springsecurity.html
- http://static.springsource.org/spring-security/site/petclinic-tutorial.html
- http://journal.mycom.co.jp/articles/2010/03/25/spring3/index.html

モジュール

内容	モジュール
Core	spring-security-core.jar
Web	spring-security-web.jar
Config	spring-security-config.jar
LDAP	spring-security-ldap.jar
ACL	spring-security-acl.jar
CAS	spring-security-cas-client.jar
OpenID	spring-security-openid.jar

POM 例

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

Namespace

Spring 設定ファイル

■ security namespace を利用するには、spring-security-config.jar がクラスパスにある必要がある

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:security="http://www.springframework.org/schema/security"
  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-3.0.xsd
          http://www.springframework.org/schema/security
          http://www.springframework.org/schema/security/spring-security-3.0.3.xsd">
    ...
</beans>
```

■ beans ではなく、security をデフォルト namespace とする場合

```
<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-3.0.xsd
           http://www.springframework.org/schema/security
           http://www.springframework.org/schema/security/spring-security-3.0.3.xsd">
    ...
</beans:beans>
```

■最小限の HTTP セキュリティ

```
 <http auto-config='true'>
   <intercept-url pattern="/**" access="ROLE_USER" />
 </http>
```

Web.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/security.xml /WEB-INF/spring/root-context.xml</param-value>
</context-param>

<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>
```

Database

JDBC

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName" value="org.hsqldb.jdbcDriver"/>
  <property name="url" value="jdbc:hsqldb:hsql://localhost:9001"/>
  <property name="username" value="sa"/>
  <property name="password" value=""/>
</bean>

                        <bean                                                    id="userDetailsService"
class="org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

スキーマ

・http://static.springsource.org/spring-security/site/docs/3.0.x/reference/appendix-schema.html

■ User スキーマ

```
 create table users(
     username varchar_ignorecase(50) not null primary key,
     password varchar_ignorecase(50) not null,
     enabled boolean not null);


 create table authorities (
     username varchar_ignorecase(50) not null,
     authority varchar_ignorecase(50) not null,
     constraint fk_authorities_users foreign key(username) references users(username));
     create unique index ix_auth_username on authorities (username,authority);
```

■グループ権限

```
create table groups (
  id bigint generated by default as identity(start with 0) primary key,
  group_name varchar_ignorecase(50) not null);


create table group_authorities (
  group_id bigint not null,
  authority varchar(50) not null,
  constraint fk_group_authorities_group foreign key(group_id) references groups(id));


create table group_members (
  id bigint generated by default as identity(start with 0) primary key,
  username varchar(50) not null,
  group_id bigint not null,
  constraint fk_group_members_group foreign key(group_id) references groups(id));
```

■ Persistent Login (Remember-Me) スキーマ

```
create table persistent_logins (
  username varchar(64) not null,
  series varchar(64) primary key,
  token varchar(64) not null,
  last_used timestamp not null);
```

コアコンポーネント

・http://static.springsource.org/spring-security/site/docs/3.0.x/reference/technical-overview.html#core-components

SecurityContextHolder、SecurityContext

■概要

・最も基本的なオブジェクト
・アプリケーションの存在しているセキュリティコンテキストの詳細を格納
・セキュリティコンテキストはアプリケーションで使用しているセキュリティプリンシパル (アクセス権の付与対象主体) の詳細を含んでいる

■戦略

スレッドローカル

・SecurityContextHolder.MODE_THREADLOCAL がデフォルト
・デフォルトでは SecurityContextHolder は ThreadLocal にこれら詳細を格納する
・これはセキュリティコンテキストは同じ実行スレッドにおいて常にメソッドに対して有効ということを意味する
・ThreadLocal を利用することは、現在の主体のリクエストが完了すればクリアされるため極めて安全

グローバル

・例えば、Swing クライアントは JVM のすべてのスレッドを必要とするかもしれない。スタンドアロンアプリケーションでは、SecurityContextHolder.MODE_GLOBAL 戦略 を利用できる。

継承可能なスレッドローカル

・セキュアなスレッドから発生したスレッドに同じセキュリティを仮定する場合、SecurityContextHolder.MODE_INHERITABLETHREADLOCAL を利用するとよい

モードの変更方法

1. システムプロパティで変更
2.SecurityContextHolder の静的メソッドをコールすることで変更

Authentication

現在ユーザーについての情報を取得する

・SecurityContextHolder の内部に、アプリケーションのセキュリティプリンシパル (認証主体) を保存している
・Spring Security は この情報を表現する Authentiaction オブジェクトを利用する
・通常 Authentication オブジェクトを自分自身で生成する必要はない

■認証ユーザーの取得

・アプリケーションのどこからでも、現在認証されているユーザーを取得することができる

例

```
Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {
  String username = ((UserDetails)principal).getUsername();
} else {
  String username = principal.toString();
}
```

・getContext() は、SecurityContext のインスタンスを返す
・SecurityContext はスレッドローカルなストレージオブジェクト
・ほとんどの認証メカニズムではセキュリティプリンシパル (認証主体) として UserDetail のインスタンスを返す。

UserDetailsService

・Authentication オブジェクトから セキュリティプリンシパル (認証主体) を取得することができる
・セキュリティプリンシパル (認証主体) は単なるオブジェクト。ほとんどの場合、UserDetails オブジェクトにキャストできる
・UserDetails は Spring Security の中心的なインターフェースで、セキュリティプリンシパル (認証主体) を表現しており、アプリケーション使用にあわせて拡張可能
・UserDetails を 自分自身のユーザーデータベースと、Spring Security が内部で必要とする SecurityContentHolder のアダプターと考えることができる
・自分自身のユーザーデータベースからなんらかを表現したものであると言うことは、UserDetail を アプリケーションが提供するオリジナルオブジェクト (ビジネス仕様に基づいたメソッド、例えば getEmail(),getEmployeeNumber などをもつ) にキャストできるということである。

■どのようにオリジナル UserDetails オブジェクトを提供するか

・UserDetails を返す、UserDetailsService とよばれる特別なインターフェースがある
・Spring Security にユーザー情報を取り込む、最も一般的なやり方

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

・認証の成功により、UserDetails は、SecurityContextHoder に保存される Authentication オブジェクトをビルドするのに使われる
・Spring Security は、いくつかの UserDetailServiice の実装を提供している
  ・ImMemoryDaoImpl
  ・JdbcDaoImpl

GrantedAuthority

・Authentication は getAutorities() という重要なメソッドも提供している。
・このメソッドは、GrantedAuthority オブジェクトの配列を提供する
・GrantedAuthority は、セキュリティプリンシパル (認証主体) に与えられた権限であり、通常このような権限はロールと呼ばれる
・通常 GrantedAuthority オブジェクトは、アプリケーション単位の許可であり、ドメインオブジェクトとしては設計されない

まとめ

・SecurityContentHolder は SecurityContext へのアクセスを提供する
・SecurityContent は Authentication を保持し、セキュリティ情報へのリクエストを可能にする
・Authentication は Supring Security での セキュリティプリンシパル (認証主体) を表現している
・GrantedAuthority は セキュリティプリンシパル (認証主体) にアプリケーション単位に付与された許可をリフレクトする
・UserDetails は Authentication をアプリケーションの DAO や他のセキュリティデータソースからビルドするに必要な情報を提供する
・UserDetailServiice は、文字列ベースのユーザー名 (もしくは認証 ID など) が通過したときに、UserDetails を生成する

Tips

Controller から認証ユーザーを取得

・http://stackoverflow.com/questions/248562/when-using-spring-security-what-is-the-proper-way-to-obtain-current-username-i

```
@RequestMapping(method = RequestMethod.GET)
public ModelAndView showResults(final HttpServletRequest request, Principal principal) {
       final String currentUser = principal.getName();
       ...
}
```

JSP から認証ユーザーを取得

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

<security:authentication property="principal.username" />
```

Links

■ JDBC 認証プロバイダのクエリを変更できそう

・http://myblog.shriharisc.com/2010/09/27/customizing-jdbc-authentication-provider-in-spring-security/

■リクエスト元の IP アドレスやユーザーエージェントでアクセス制御できそう

・http://static.springsource.org/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/web/authentication/DelegatingAuthenticationEntryPoint.html