**Spring Security 3.0 安全权限管理手册**

参考文献：

1、[www.family168.com中的spring](http://www.family168.com中的spring) security权限管理手册。

2、spring security3.0权限管理手册

3、spring的相关资料。

本文档内容仅仅作为公司权限管理资料用，对于企业来说，权限管理将是系统中的非常重要的一个模块，权限的设计也是参考相关资料进行整理和补充。系统将通过数据库进行管理用户权限。

**权限管理搭建要的问题：**

**1、区分Authentication（验证）与 Authorization（授权）**

验证

这个用户是谁？

用户身份可靠吗？

授权

某用户A是否可以访问资源R

某用户A是否可以执行M操作

某用户A是否可以对资源R执行M操作

**2、SS中的验证特点**

支持多种验证方式

支持多种加密格式

支持组件的扩展和替换

可以本地化输出信息

**3、SS中的授权特点**

支持多种仲裁方式

支持组件的扩展和替换

支持对页面访问、方法访问、对象访问的授权。

**4、SS核心安全实现**

Web安全

通过配置Servlet Filter激活SS中的过滤器链

实现Session一致性验证

实现免登陆验证（Remember-Me验证）

提供一系列标签库进行页面元素的安全控制

方法安全

通过AOP模式实现安全代理

Web安全与方法安全均可以使用表达式语言定义访问规则

**5、配置SS**

配置Web.xml，应用安全过滤器

配置Spring，验证与授权部分

在web页面中获取用户身份

在web页面中应用安全标签库

实现方法级安全

**6、配置web.xml**

**7、Spring配置文件中设置命名空间**

**8、通过数据库验证用户身份**

**9、完善web页面验证规则**

**10、自定义验证配置**

**11、本地化消息输出（国际化）**

根据公司项目的开发要求和集合spring security3.0功能，公司将通过数据库进行对用户身份验证和授权，系统将建立5个基础表进行对权利的管理。

**第一部分 数据库设计**

**1、表设计**

表1：用户表（pub\_users）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | User\_Id | Vchar(32) | 用户id | PK |
| 2 | user\_account | Vchar(30) | 登陆用户名(登陆号) |  |
| 3 | User\_name | Vchar(40) | 用户姓名 |  |
| 4 | user\_Password | Vchar(100) | 用户密码 |  |
| 5 | Enabled | Int | 是否被禁用 | 0禁用1正常 |
| 6 | isSys | Int | 是否是超级用户 | 0非1是 |
| 7 | user\_DESc | Vchar(100) | 描述 |  |
| 说明：pub\_users表中的登录名和密码用来控制用户的登录。 | | | | |

表2：权限表（pub\_authorities）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | authority\_Id | Vchar(32) | 权限id | PK |
| 2 | Authority\_name | Vchar(40) | 权限名称 |  |
| 3 | Authority\_DESc | Vchar(100) | 权限描述 |  |
| 4 | Enabled | Int | 是否被禁用 | 0禁用1正常 |
| 5 | isSys | Int | 是否是超级权限 | 0非1是 |
| 说明：pub\_authorities表中描述的是系统拥有哪些权限，如果要详细分类，可以将一个url定义一个权限，那样就能对所有资源进行管理。 | | | | |

表3：角色表（pub\_roles）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | role\_Id | Vchar(32) | 角色id | PK |
| 2 | role\_name | Vchar(100) | 角色名称 |  |
| 3 | role\_DESc | Vchar(100) | 角色描述 |  |
| 4 | Enabled | Int | 是否被禁用 | 0禁用1正常 |
| 5 | isSys | Int | 是否是超级权限 | 0非1是 |
| 说明：pub\_roles表中描述的是系统按用户分类或按照功能模块分类，将系统进行整合归类管理。 | | | | |

表4：资源表（pub\_resources）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | resource\_Id | Vchar(32) | 资源id | PK |
| 2 | resource\_name | Vchar(100) | 资源名称 |  |
| 3 | resource \_type | Vchar(40) | 资源类型 | url、method |
| 4 | priority | int | 资源优先权 | 即排序 |
| 5 | resource \_string | Vchar(200) | 资源链接 |  |
| 6 | resource\_DESc | Vchar(100) | 资源描述 |  |
| 7 | Enabled | Int | 是否被禁用 | 0禁用1正常 |
| 8 | isSys | Int | 是否是超级权限 | 0非1是 |
| 说明：pub\_roles表中描述的是系统需要保护的资源及（url或方法）。 | | | | |

以上四个表是权限管理的基础表（用户表、权限表、角色表、资源表）。

表5：用户角色连接表(pub\_users\_roles)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | Id | Indetity | Id主键 | PK |
| 2 | user\_Id | Vchar(32) | 用户id |  |
| 3 | role\_id | Vchar(32) | 角色id |  |
| 说明：用来管理用户和角色的关系。 | | | | |

表6：角色权限连接表(pub\_roles\_authorities)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | Id | Indetity | Id主键 | PK |
| 2 | role \_Id | Vchar(32) | 角色id |  |
| 3 | authority\_Id | Vchar(32) | 权限id |  |
| 说明：用来管理角色和权限的关系。 | | | | |

表7：权限资源连接表(pub\_authorities\_resources)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **字段** | **类型** | **含义** | **备注** |
| 1 | Id | Indetity | Id主键 | PK |
| 2 | authority\_Id | Vchar(32) | 权限id |  |
| 3 | resource\_Id | Vchar(32) | 资源id |  |
| 说明：用来管理角色和权限的关系。 | | | | |

**2、建表语句如下（数据库采用MS SQL 2000）：**

create table pub\_users(

user\_id varchar(32),

user\_account varchar(30),

user\_name varchar(40),

user\_password varchar(100),

user\_desc varchar(100),

enabled int,

issys int

);

alter table pub\_users add constraint pk\_pub\_users primary key(user\_id);

create table pub\_authorities(

authority\_id varchar(32),

authority\_name varchar(40),

authority\_desc varchar(100),

enabled int,

issys int

);

alter table pub\_authorities add constraint pk\_pub\_authorities primary key(authority\_id);

create table pub\_roles(

role\_id varchar(32),

role\_name varchar(40),

role\_desc varchar(100),

enabled int,

issys int

);

alter table pub\_roles add constraint pk\_pub\_roles primary key(role\_id);

create table pub\_resources(

resource\_id varchar(32),

resource\_name varchar(100),

resource\_desc varchar(100),

resource\_type varchar(40),

resource\_string varchar(200),

priority int,

enabled int,

issys int

);

alter table pub\_resources add constraint pk\_pub\_resources primary key(resource\_id);

create table pub\_users\_roles(

id numeric(12,0) IDENTITY NOT NULL,

user\_id varchar(32),

role\_id varchar(32),

enabled int

);

alter table pub\_users\_roles add constraint pk\_pub\_users\_roles primary key(id);

alter table pub\_users\_roles add constraint fk\_users\_roles\_users foreign key(user\_id) references pub\_users(user\_id);

alter table pub\_users\_roles add constraint fk\_users\_roles\_roles foreign key(role\_id) references pub\_roles(role\_id);

create table pub\_roles\_authorities(

id numeric(12,0) IDENTITY NOT NULL,

role\_id varchar(32),

authority\_id varchar(32),

enabled int

);

alter table pub\_roles\_authorities add constraint pk\_pub\_roles\_authorities primary key(id);

alter table pub\_roles\_authorities add constraint fk\_pub\_roles\_authorities\_authorities foreign key(authority\_id) references pub\_authorities(authority\_id);

alter table pub\_roles\_authorities add constraint fk\_pub\_roles\_authorities\_roles foreign key(role\_id) references pub\_roles(role\_id);

create table pub\_authorities\_resources(

id numeric(12,0) IDENTITY NOT NULL,

authority\_id varchar(32),

resource\_id varchar(32),

enabled int

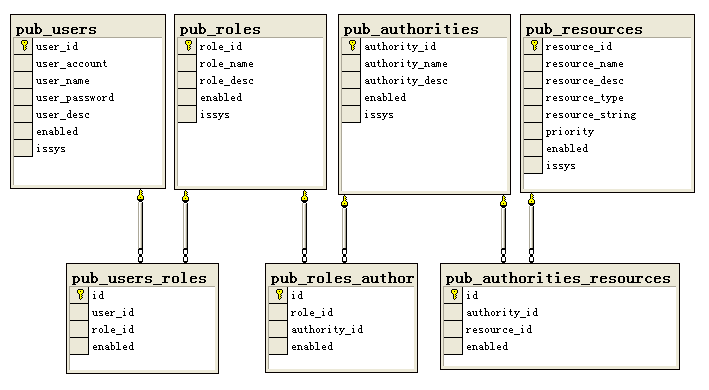
);

alter table pub\_authorities\_resources add constraint pk\_pub\_authorities\_resources primary key(id);

alter table pub\_authorities\_resources add constraint fk\_pub\_authorities\_resources\_authorities foreign key(authority\_id) references pub\_authorities(authority\_id);

alter table pub\_authorities\_resources add constraint fk\_pub\_authorities\_resources\_resources foreign key(resource\_id) references pub\_resources(resource\_id);

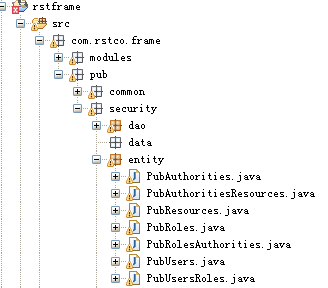
**3、E-R图如下：**



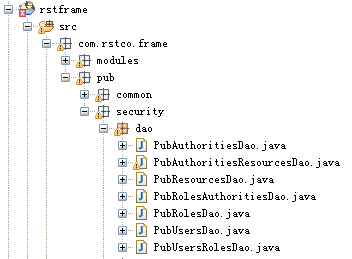
**第二部分 WEB数据库整合**

**提示：相关代码请参考项目模块**

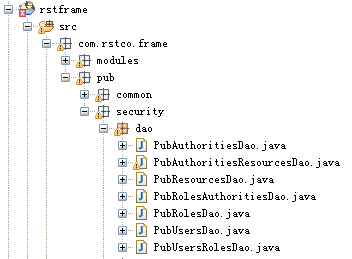
**1、将数据库表结构和Hibernate建立映射，本系统采用annotation进行对数据库进行零配置处理（请参考hibernate映射），如图。**



**2、建立权限的Dao层。**

****

**3、建立权限的Service层**

****

**4、配置web.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*

*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<display-name>rstframe</display-name>

<context-param>

<param-name>webAppRootKey</param-name>

<param-value>rstframe.root</param-value>

</context-param>

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:log4j.properties</param-value>

</context-param>

<context-param>

<param-name>log4jRefreshInterval</param-name>

<param-value>60000</param-value>

</context-param>

<!-- Spring ApplicationContext配置文件的路径,可使用通配符,多个路径用,号分隔

此参数用于后面的Spring Context Loader -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath\*:/applicationContext.xml,

classpath\*:/applicationContext-rstframe.xml

</param-value>

</context-param>

<!-- Character Encoding filter -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>

org.springframework.web.filter.CharacterEncodingFilter

</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- SpringSide's Hibernate Open Session In View filter-->

<filter>

<filter-name>hibernateOpenSessionInViewFilter</filter-name>

<filter-class>

com.rstco.frame.modules.orm.hibernate.OpenSessionInViewFilter

</filter-class>

<init-param>

<param-name>excludeSuffixs</param-name>

<param-value>js,css,jpg,gif</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>hibernateOpenSessionInViewFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- SpringSecurity filter-->

<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>

<!-- Struts2 filter, actionPackages -->

<filter>

<filter-name>struts2Filter</filter-name>

<filter-class>

org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<listener>

<listener-class>

org.springframework.web.util.Log4jConfigListener

</listener-class>

</listener>

<!-- Spring 刷新Introspector防止内存泄露 -->

<listener>

<listener-class>

org.springframework.web.util.IntrospectorCleanupListener

</listener-class>

</listener>

<!-- 防止多人登陆 ,控制一个用户只能登录一次，不能在其他地方重新登录-->

<listener>

<listener-class>

org.springframework.security.web.session.HttpSessionEventPublisher

</listener-class>

</listener>

<!-- session超时定义,单位为分钟 -->

<session-config>

<session-timeout>20</session-timeout>

</session-config>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<!-- error page -->

<error-page>

<exception-type>java.lang.Throwable</exception-type>

<location>/common/500.jsp</location>

</error-page>

<error-page>

<error-code>500</error-code>

<location>/common/500.jsp</location>

</error-page>

<error-page>

<error-code>404</error-code>

<location>/common/404.jsp</location>

</error-page>

<error-page>

<error-code>403</error-code>

<location>/common/403.jsp</location>

</error-page>

<jsp-config>

<taglib>

<taglib-uri>/WEB-INF/struts-menu-el.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/struts-menu-el.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/struts-menu.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/struts-menu.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/c.tld</taglib-uri>

<taglib-location>/WEB-INF/tlds/c.tld</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/fmt.tld</taglib-uri>

<taglib-location>/WEB-INF/tlds/fmt.tld</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/fn.tld</taglib-uri>

<taglib-location>/WEB-INF/tlds/fn.tld</taglib-location>

</taglib>

<!--loushang tld-->

<taglib>

<taglib-uri>/WEB-INF/web-date.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-date.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-flex.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-flex.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-graph.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-graph.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-grid.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-grid.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-html.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-html.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-list.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-list.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-loushang.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-loushang.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-menu.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-menu.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-multitab.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-multitab.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-seltree.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-seltree.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-tab.tld</taglib-uri>

<taglib-location>/WEB-INF/tlds/web-tab.tld</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-tree.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-tree.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-widgets.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-widgets.tld

</taglib-location>

</taglib>

<taglib>

<taglib-uri>/WEB-INF/web-i18n.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/web-i18n.tld

</taglib-location>

</taglib>

<!-- loushang end -->

<taglib>

<taglib-uri>/WEB-INF/gystudio.tld</taglib-uri>

<taglib-location>

/WEB-INF/tlds/gystudio.tld

</taglib-location>

</taglib>

</jsp-config>

<mime-mapping>

<extension>rar</extension>

<mime-type>application/rar</mime-type>

</mime-mapping>

</web-app>

**5、配置spring security3.0中的xml文件**

**文件名：applicationContext-security.xml**

<?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-3.0.xsd*

*http://www.springframework.org/schema/security*

*http://www.springframework.org/schema/security/spring-security-3.0.xsd"*>

<beans:description>SpringSecurity安全配置</beans:description>

<!-- http安全配置 -->

<http auto-config=*"true"*>

<intercept-url pattern=*"/css/\*\*"* filters=*"none"* />

<intercept-url pattern=*"/images/\*\*"* filters=*"none"* />

<intercept-url pattern=*"/js/\*\*"* filters=*"none"* />

<intercept-url pattern=*"/login.jsp"* filters=*"none"* />

<!--

<intercept-url pattern="/index.jsp" access="ROLE\_USER"/>

<intercept-url pattern="/main.jsp" access="ROLE\_ADAMIN"/>

-->

<form-login login-page=*"/login.jsp"* default-target-url=*"/index.jsp"*

authentication-failure-url=*"/login.jsp?error=1"* />

<!-- 尝试访问没有权限的页面时跳转的页面 -->

<access-denied-handler error-page=*"/common/403.jsp"*/>

<logout logout-success-url=*"/login.jsp"* />

<session-management>

<concurrency-control max-sessions=*"1"* error-if-maximum-exceeded=*"true"* />

</session-management>

<!-- 增加一个filter，这点与Acegi是不一样的，不能修改默认的filter了，

这个filter位于FILTER\_SECURITY\_INTERCEPTOR之前 -->

<custom-filter ref=*"myFilter"* before=*"FILTER\_SECURITY\_INTERCEPTOR"*/>

</http>

<!-- 一个自定义的filter，必须包含authenticationManager,accessDecisionManager,securityMetadataSource三个属性，

我们的所有控制将在这三个类中实现，解释详见具体配置 -->

<beans:bean id=*"myFilter"* class=*"com.rstco.frame.pub.security.interceptor.MyFilterSecurityInterceptor"*>

<beans:property name=*"authenticationManager"*

ref=*"authenticationManager"* />

<beans:property name=*"accessDecisionManager"*

ref=*"myAccessDecisionManagerBean"* />

<beans:property name=*"securityMetadataSource"*

ref=*"mySecurityMetadataSource"* />

</beans:bean>

<!-- 验证配置 ， 认证管理器，实现用户认证的入口，主要实现UserDetailsService接口即可 -->

<authentication-manager alias=*"authenticationManager"*>

<authentication-provider user-service-ref=*"userDetailsService"*>

<!--

<s:password-encoder hash="sha" />

-->

</authentication-provider>

</authentication-manager>

<!-- 项目实现的用户查询服务,将用户信息查询出来 -->

<beans:bean id=*"userDetailsService"* class=*"com.rstco.frame.pub.security.support.MyUserDetailService"* />

<!-- 访问决策器，决定某个用户具有的角色，是否有足够的权限去访问某个资源 -->

<beans:bean id=*"myAccessDecisionManagerBean"*

class=*"com.rstco.frame.pub.security.support.MyAccessDecisionManager"*>

</beans:bean>

<!-- 资源源数据定义，将所有的资源和权限对应关系建立起来，即定义某一资源可以被哪些角色访问 -->

<beans:bean id=*"mySecurityMetadataSource"*

class=*"com.rstco.frame.pub.security.support.MyInvocationSecurityMetadataSourceService"*>

</beans:bean>

<!-- 定义国际化 -->

<beans:bean id=*"messageSource"*

class=*"org.springframework.context.support.ReloadableResourceBundleMessageSource"*>

<beans:property name=*"basename"*

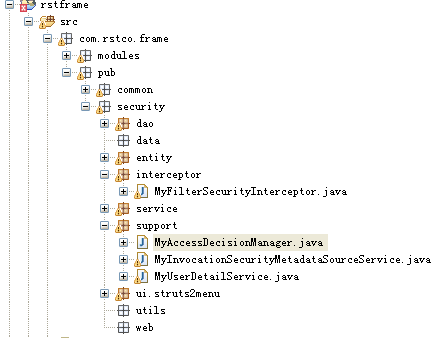
value=*"classpath:org/springframework/security/messages\_zh\_CN"*/>

</beans:bean>

</beans:beans>

**第三部分 SS3.0的实现**

**这是项目的主体部分：**

****

**这四个类说明如下。**

1. 用来获得用户验证信息（MyUserDetailService）

代码如下：

package com.rstco.frame.pub.security.support;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.dao.DataAccessException;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

import com.rstco.frame.pub.security.dao.PubAuthoritiesResourcesDao;

import com.rstco.frame.pub.security.dao.PubUsersDao;

import com.rstco.frame.pub.security.entity.PubAuthorities;

import com.rstco.frame.pub.security.entity.PubAuthoritiesResources;

//你就可以从数据库中读入用户的密码，角色信息，是否锁定，账号是否过期

@Service

public class MyUserDetailService implements UserDetailsService {

@Autowired

private PubUsersDao pubUsersDao;

@Autowired

private PubAuthoritiesResourcesDao pubAuthoritiesResourcesDao;

public UserDetails loadUserByUsername(String username)

throws UsernameNotFoundException, DataAccessException {

Collection<GrantedAuthority> auths=new ArrayList<GrantedAuthority>();

//取得用户的权限

List<PubAuthorities> auth=pubUsersDao.findAuthByUserName(username);

String password=null;

//取得用户的密码

password=pubUsersDao.findUserByname(username).get(0).getUserPassword();

List<PubAuthoritiesResources> aaa=pubAuthoritiesResourcesDao.getAll();

User user = new User(username,

password, true, true, true, true, auths);

return user;

}

}

1. 最核心的地方，就是提供某个资源对应的权限定义，取得所有角色（auth）的对应资源数据（MyInvocationSecurityMetadataSourceService）

代码如下：

package com.rstco.frame.pub.security.support;

import java.util.ArrayList;

import java.util.Collection;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import javax.servlet.ServletContext;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.security.access.ConfigAttribute;

import org.springframework.security.access.SecurityConfig;

import org.springframework.security.web.FilterInvocation;

import org.springframework.security.web.access.intercept.FilterInvocationSecurityMetadataSource;

import org.springframework.security.web.util.AntUrlPathMatcher;

import org.springframework.security.web.util.UrlMatcher;

import org.springframework.stereotype.Service;

import com.rstco.frame.modules.orm.hibernate.HibernateDao;

import com.rstco.frame.pub.security.dao.PubAuthoritiesResourcesDao;

import com.rstco.frame.pub.security.entity.PubAuthorities;

import com.rstco.frame.pub.security.entity.PubResources;

/\*

\*

\* 最核心的地方，就是提供某个资源对应的权限定义，即getAttributes方法返回的结果。

\* 注意，我例子中使用的是AntUrlPathMatcher这个path matcher来检查URL是否与资源定义匹配，

\* 事实上你还要用正则的方式来匹配，或者自己实现一个matcher。

\*

\* 此类在初始化时，应该取到所有资源及其对应角色的定义

\*

\* 说明：对于方法的spring注入，只能在方法和成员变量里注入，

\* 如果一个类要进行实例化的时候，不能注入对象和操作对象，

\* 所以在构造函数里不能进行操作注入的数据。

\*/

@Service

public class MyInvocationSecurityMetadataSourceService implements

FilterInvocationSecurityMetadataSource {

@Autowired

private PubAuthoritiesResourcesDao pubAuthoritiesResourcesDao;

private UrlMatcher urlMatcher = new AntUrlPathMatcher();

private static Map<String, Collection<ConfigAttribute>> resourceMap = null;

public MyInvocationSecurityMetadataSourceService() {

loadResourceDefine();

}

/\* private void loadResourceDefine() {

resourceMap = new HashMap<String, Collection<ConfigAttribute>>();

Collection<ConfigAttribute> atts = new ArrayList<ConfigAttribute>();

ConfigAttribute ca = new SecurityConfig("ROLE\_ADMIN");

atts.add(ca);

resourceMap.put("/index.jsp", atts);

resourceMap.put("/i.jsp", atts);

}\*/

private void loadResourceDefine() {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

SessionFactory sessionFactory = (SessionFactory)context.getBean("sessionFactory");

Session session = sessionFactory.openSession();

List<String> query=session.createSQLQuery("select authority\_name from pub\_authorities ").list();

resourceMap = new HashMap<String, Collection<ConfigAttribute>>();

Collection<ConfigAttribute> atts = new ArrayList<ConfigAttribute>();

//List<PubAuthorities> auths =session.createQuery(arg0); //pubAuthoritiesResourcesDao.findAuthAll();

for (String auth : query) {

ConfigAttribute ca = new SecurityConfig(auth);// "ROLE\_ADMIN"

// atts.add(ca);

List<String> query1=session.createSQLQuery("select resource\_string " +

"from Pub\_Authorities\_Resources,Pub\_Resources, Pub\_authorities " +

"where Pub\_Authorities\_Resources.resource\_id=Pub\_Resources.resource\_id and " +

" Pub\_Authorities\_Resources.resource\_id=Pub\_authorities.authority\_id and " +

" Authority\_name='"+auth+"'").list();

for (String res : query1) {

String url = res;

// 判断资源文件和权限的对应关系，如果已经存在，要进行增加

if (resourceMap.containsKey(url)) {

Collection<ConfigAttribute> value = resourceMap.get(url);

value.add(ca);

resourceMap.put(url, value);

// "log.jsp","role\_user,role\_admin"

} else {

atts.add(ca);

resourceMap.put(url, atts);

}

resourceMap.put(url, atts);

}

}

}

// According to a URL, Find out permission configuration of this URL.

public Collection<ConfigAttribute> getAttributes(Object object)

throws IllegalArgumentException {

// guess object is a URL.

String url = ((FilterInvocation) object).getRequestUrl();

Iterator<String> ite = resourceMap.keySet().iterator();

while (ite.hasNext()) {

String resURL = ite.next();

if (urlMatcher.pathMatchesUrl(url, resURL)) {

return resourceMap.get(resURL);

}

}

return null;

}

public boolean supports(Class<?> clazz) {

return true;

}

public Collection<ConfigAttribute> getAllConfigAttributes() {

return null;

}

}

1. 最重要的是decide方法，如果不存在对该资源的定义，直接放行；否则，如果找到正确的角色，即认为拥有权限，并放行，否则throw new AccessDeniedException("no right");这样，就会进入上面提到的403.jsp页面。(MyAccessDecisionManager)

代码如下：

package com.rstco.frame.pub.security.support;

import java.util.Collection;

import java.util.Iterator;

import org.springframework.security.access.AccessDecisionManager;

import org.springframework.security.access.AccessDeniedException;

import org.springframework.security.access.ConfigAttribute;

import org.springframework.security.access.SecurityConfig;

import org.springframework.security.authentication.InsufficientAuthenticationException;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.GrantedAuthority;

public class MyAccessDecisionManager implements AccessDecisionManager {

//In this method, need to compare authentication with configAttributes.

// 1, A object is a URL, a filter was find permission configuration by this URL, and pass to here.

// 2, Check authentication has attribute in permission configuration (configAttributes)

// 3, If not match corresponding authentication, throw a AccessDeniedException.

public void decide(Authentication authentication, Object object,

Collection<ConfigAttribute> configAttributes)

throws AccessDeniedException, InsufficientAuthenticationException {

if(configAttributes == null){

return ;

}

System.out.println(object.toString()); //object is a URL.

Iterator<ConfigAttribute> ite=configAttributes.iterator();

while(ite.hasNext()){

ConfigAttribute ca=ite.next();

String needRole=((SecurityConfig)ca).getAttribute();

for(GrantedAuthority ga:authentication.getAuthorities()){

if(needRole.equals(ga.getAuthority())){ //ga is user's role.

return;

}

}

}

throw new AccessDeniedException("no right");

}

public boolean supports(ConfigAttribute attribute) {

// TODO Auto-generated method stub

return true;

}

public boolean supports(Class<?> clazz) {

return true;

}

}

1. 这个过滤器要插入到授权之前。最核心的代码就是invoke方法中的InterceptorStatusToken token = super.beforeInvocation(fi);这一句，即在执行doFilter之前，进行权限的检查，而具体的实现已经交给accessDecisionManager了(MyFilterSecurityInterceptor)

代码如下：

package com.rstco.frame.pub.security.interceptor;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.access.AccessDecisionManager;

import org.springframework.security.access.SecurityMetadataSource;

import org.springframework.security.access.intercept.AbstractSecurityInterceptor;

import org.springframework.security.access.intercept.InterceptorStatusToken;

import org.springframework.security.web.FilterInvocation;

import org.springframework.security.web.access.intercept.FilterInvocationSecurityMetadataSource;

public class MyFilterSecurityInterceptor extends AbstractSecurityInterceptor

implements Filter {

private FilterInvocationSecurityMetadataSource securityMetadataSource;

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

FilterInvocation fi = new FilterInvocation(request, response, chain);

invoke(fi);

}

public FilterInvocationSecurityMetadataSource getSecurityMetadataSource() {

return this.securityMetadataSource;

}

public Class<? extends Object> getSecureObjectClass() {

return FilterInvocation.class;

}

public void invoke(FilterInvocation fi) throws IOException,

ServletException {

InterceptorStatusToken token = super.beforeInvocation(fi);

try {

fi.getChain().doFilter(fi.getRequest(), fi.getResponse());

} finally {

super.afterInvocation(token, null);

}

}

@Override

public SecurityMetadataSource obtainSecurityMetadataSource() {

return this.securityMetadataSource;

}

public void setSecurityMetadataSource(

FilterInvocationSecurityMetadataSource securityMetadataSource) {

System.out.println("abc=======================edf");

this.securityMetadataSource = securityMetadataSource;

}

public void destroy() {

// TODO Auto-generated method stub

}

public void init(FilterConfig filterconfig) throws ServletException {

// TODO Auto-generated method stub

}

}

**如有异议，请加qq：89168934，互相学习交流。**