个人资料

这家伙很懒，什么都没有留下.....

等级:

加入时间:2011-12-05 22:36

发送消息

操作中心

- 登录
- 注册

所有文章

- JDK基础
- html5
- javascript
- 我的JavaEE
- 我的博客
- 数据结构

最新发表>>

- jQuery实现的视差滚动教程
- java Annotation注解实现原理
- 开发人员怎样在技术面试中脱颖而出
- 几何变换详解
- 移动端html5重力感应参数分析

最新评论>>

上面的代码找不到ResourceDetailsBuilder这个类
上面已经是源码了哟，只是没有JAR包而已，你下载一下吧
大大共享下源码可以么？
是内存中的权限缓存，这样设置的权限可以即时生效
refresh 刷新是指的什么刷新？

spring Security3.1安全框架配置教程

发表于2012-03-03 18:39 | 阅读 754 | 2人对此综合评价

前端时间一直在研究spring Security3.1这个框架,感觉用着还算不错,包括本站也是使用是了这个框架正好利用这个机会学习一下.特地给大家分享一下,希望对大家有所帮助.对这个框架的介绍就不用多说了,网上都有相关的资料.我要介绍的是如何利用数据库来存储所有的权限,这样的话就直接可以通过在数据库中增加记录来扩展权限,不用每次都要在XML中配置权限.

为了测试更加方便我用的是haldb数据库,比较简单.

首先到spring官网把最新版的spring Security安全框架的压缩文件下载下来:

http://www.springsource.org/spring-community-download

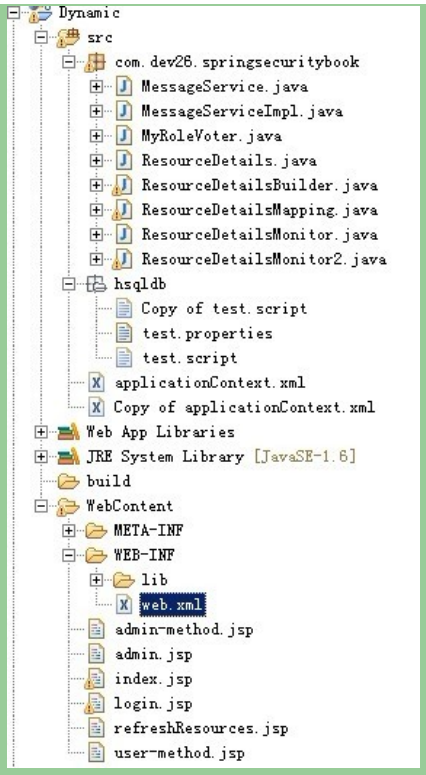
然后在Eclipse中新一个Web工程,在web.xml中加入以下权限过滤器来控制来拦截所有访问

```
1  <filter>
2      <filter-name>springSecurityFilterChain</filter-name>
3      <filter-
4  class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
5  </filter>
6  <filter-mapping>
7      <filter-name>springSecurityFilterChain</filter-name>
8      <url-pattern>/*</url-pattern>
   </filter-mapping>
```

然后在listener中引入spring 的bean配置文件:

```
1    <context-param>
2        <param-name>contextConfigLocation</param-name>
3        <param-value>classpath:applicationContext*.xml</param-value>
4    </context-param>
5  <listener>
6        <listener-
7  class>org.springframework.web.context.ContextLoaderListener</listener-class>
     </listener>
```

整工程截图如下:



以下是hqldb的初始化脚本,也不用建库只要把hqldb.jar放到工程中则会自动执行.这就是工程的test.script的脚本sql

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE MEMORY TABLE RESC(ID BIGINT GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT
NULL,NAME VARCHAR(50),RES_TYPE VARCHAR(50),RES_STRING VARCHAR(200),PRIORITY
INTEGER,DESCN VARCHAR(200),CONSTRAINT PK_RESC PRIMARY KEY(ID))
CREATE MEMORY TABLE ROLE(ID BIGINT GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT
NULL,NAME VARCHAR(50),DESCN VARCHAR(200),CONSTRAINT PK_ROLE PRIMARY KEY(ID))
CREATE MEMORY TABLE USER(ID BIGINT GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT
NULL,USERNAME VARCHAR(50),PASSWORD VARCHAR(50),STATUS INTEGER,DESCN
VARCHAR(200),CONSTRAINT PK_USER PRIMARY KEY(ID))
CREATE MEMORY TABLE RESC_ROLE(RESC_ID BIGINT NOT NULL,ROLE_ID BIGINT NOT
NULL,CONSTRAINT PK_RESC_ROLE PRIMARY KEY(RESC_ID,ROLE_ID),CONSTRAINT FK_RESC_ROLE_RESC
FOREIGN KEY(RESC_ID) REFERENCES RESC(ID),CONSTRAINT FK_RESC_ROLE_ROLE FOREIGN
KEY(ROLE_ID) REFERENCES ROLE(ID))
CREATE MEMORY TABLE USER_ROLE(USER_ID BIGINT NOT NULL,ROLE_ID BIGINT NOT
NULL,CONSTRAINT PK_USER_ROLE PRIMARY KEY(USER_ID,ROLE_ID),CONSTRAINT FK_USER_ROLE_USER
FOREIGN KEY(USER_ID) REFERENCES USER(ID),CONSTRAINT FK_USER_ROLE_ROLE FOREIGN
KEY(ROLE_ID) REFERENCES ROLE(ID))
ALTER TABLE RESC ALTER COLUMN ID RESTART WITH 3
ALTER TABLE ROLE ALTER COLUMN ID RESTART WITH 3
ALTER TABLE USER ALTER COLUMN ID RESTART WITH 3
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 10
SET SCHEMA PUBLIC
INSERT INTO RESC VALUES(1,'','URL','/admin.jsp',1,'')
INSERT INTO RESC VALUES(2,'','URL','/**',2,'')
INSERT INFO RESC
VALUES(3,'','METHOD','com.dev26.springsecuritybook.MessageService.adminMessage',3,'');
INSERT INTO ROLE VALUES(1,'ROLE_ADMIN','\u7ba1\u7406\u5458\u89d2\u8272')
INSERT INTO ROLE VALUES(2,'ROLE_USER','\u7528\u6237\u89d2\u8272')
INSERT INTO USER VALUES(1,'admin','admin',1,'\u7ba1\u7406\u5458')
INSERT INTO USER VALUES(2,'user','user',1,'\u7528\u6237')
INSERT INTO RESC_ROLE VALUES(1,1)
INSERT INTO RESC_ROLE VALUES(2,1)
INSERT INTO RESC_ROLE VALUES(2,2)
INSERT INTO RESC_ROLE VALUES(3,1)
INSERT INTO USER_ROLE VALUES(1,1)
INSERT INTO USER_ROLE VALUES(1,2)
INSERT INTO USER_ROLE VALUES(2,2)
```

这些是基本配置文件,然后咱在分析一下spring Security的运行基制,然后在在看另一个applicationContext.xml中的配置就比较容易了.

Spring Security是一个基于角色的权限框架(Role Based Access Control,RBAC),它通过一系列可配置的组件构建了基于Spring IOC组件装配模式的安全框架.系统中对用于分配角色,而

角色由被授于相应的权限(增加、浏览、删除、更新..),在校验的时候首先要确定用户所属地的角色，然后根据角色来判断当前用户有哪些权限。这只是Spring Security框架的部分功能，其实它比较丰富的配置和扩展比如:(用户组(Group)的配置、ACL权限配置、CAS集成等)。

首先要配置的是一个org.springframework.web.filter.DelegatingFilterProxy的过滤器在webm.xml中这是Spring Security的一个入口，它内部有一个过滤器链。学过Acegi的朋友可能比较了解，这个例子就不深究了。

以下是Secutirity本应的配置，在applicationContext.xml中

```
<?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-3.1.xsd">

    <http  auto-config="true">

    </http>
    <!--通过SQL语句关联查询当前用户所拥有的权限  -->
<authentication-manager>
        <authentication-provider>
            <jdbc-user-service data-source-ref="dataSource"
                users-by-username-query="select username,password,status as enabled
                                        from user
                                       where username=?"
                authorities-by-username-query="select u.username,r.name as authority
                                               from user u
                                               join user_role ur
                                                 on u.id=ur.user_id
                                               join role r
                                                 on r.id=ur.role_id
                                              where u.username=?"  />
        </authentication-provider>
    </authentication-manager>

    <global-method-security />

<!--更新权限缓存,不用每次重启服务  -->
    <beans:bean id="resourceDetailsMonitor"
        class="com.dev26.springsecuritybook.ResourceDetailsMonitor2"
        autowire="byType">
        <beans:property name="dataSource" ref="dataSource" />
        <beans:property name="accessDecisionManager" ref="accessDecisionManager" />
    </beans:bean>
<!--数据源  -->
    <beans:bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <beans:property name="driverClassName" value="org.hsqldb.jdbcDriver" />
        <beans:property name="url" value="jdbc:hsqldb:res:/hsqldb/test" />
        <beans:property name="username" value="sa" />
        <beans:property name="password" value="" />
    </beans:bean>
    <!--决策管理器  -->
    <beans:bean id="accessDecisionManager"
        class="org.springframework.security.access.vote.AffirmativeBased">
        <beans:property name="allowIfAllAbstainDecisions"
            value="false" />
        <beans:property name="decisionVoters">
            <beans:list>
            <beans:bean class="com.dev26.springsecuritybook.MyRoleVoter" />
                <beans:bean
                    class="org.springframework.security.access.vote.AuthenticatedVoter"
/>
            </beans:list>
        </beans:property>
    </beans:bean>
    <beans:bean id="messageService"
        class="com.dev26.springsecuritybook.MessageServiceImpl" />
</beans:beans>
```

权限缓存刷新类代码ResourceDetailsMonitor2

```
package com.dev26.springsecuritybook;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import javax.sql.DataSource;

import org.springframework.beans.factory.InitializingBean;
import org.springframework.security.access.AccessDecisionManager;
import org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor;
import org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource;
import org.springframework.security.access.method.MethodSecurityMetadataSource;
import
org.springframework.security.web.access.intercept.FilterInvocationSecurityMetadataSource;
import org.springframework.security.web.access.intercept.FilterSecurityInterceptor;

import com.dev26.springsecuritybook.ResourceDetailsBuilder;

public class ResourceDetailsMonitor2 implements InitializingBean {

       private String queryUrl = "select re.res_string,r.name" +
       "                        from role r" +
       "                        join resc_role rr" +
       "                          on r.id=rr.role_id" +
       "                        join resc re" +
       "                          on re.id=rr.resc_id" +
       "                       where re.res_type='URL'" +
       "                    order by re.priority"; private String queryMethod =
       "select re.res_string,r.name" + "                        from role r" +
       "                        join resc_role rr" +
       "                          on r.id=rr.role_id" +
       "                        join resc re" +
       "                          on re.id=rr.resc_id" +
       "                       where re.res_type='METHOD'" +
       "                    order by re.priority";

     private DataSource dataSource;
     private FilterSecurityInterceptor filterSecurityInterceptor;
     private org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource
delegatingMethodDefinitionSource;
     private AccessDecisionManager accessDecisionManager;
     private ResourceDetailsBuilder resourceDetailsBuilder;
     private MethodSecurityInterceptor methodSecurityInterceptor;
     private Collection hasMethodAttribute;
     private Collection hasUrlAttribut;

     public void setQueryUrl(String queryUrl) {
         this.queryUrl = queryUrl;
     }

     public void setQueryMethod(String queryMethod) {
         this.queryMethod = queryMethod;
     }

     public void setDataSource(DataSource dataSource) {
         this.dataSource = dataSource;
     }

     public FilterSecurityInterceptor getFilterSecurityInterceptor() {
         return filterSecurityInterceptor;
     }

     public void setFilterSecurityInterceptor(
             FilterSecurityInterceptor filterSecurityInterceptor) {
         this.filterSecurityInterceptor = filterSecurityInterceptor;
     }

     public org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource
getDelegatingMethodDefinitionSource() {
         return delegatingMethodDefinitionSource;
     }

     public void setDelegatingMethodDefinitionSource(
             org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource
delegatingMethodDefinitionSource) {
         this.delegatingMethodDefinitionSource = delegatingMethodDefinitionSource;
     }

     public ResourceDetailsBuilder getResourceDetailsBuilder() {
         return resourceDetailsBuilder;
     }

     public void setResourceDetailsBuilder(
             ResourceDetailsBuilder resourceDetailsBuilder) {
```

```
86              this.resourceDetailsBuilder = resourceDetailsBuilder;
87          }
88
89          public String getQueryUrl() {
90              return queryUrl;
91          }
92
93          public String getQueryMethod() {
94              return queryMethod;
95          }
96
97          public DataSource getDataSource() {
98              return dataSource;
99          }
100
101         public void afterPropertiesSet() {
102             resourceDetailsBuilder = new ResourceDetailsBuilder(dataSource);
103             filterSecurityInterceptor
104                     .setAccessDecisionManager(accessDecisionManager);
105             this.hasMethodAttribute = delegatingMethodDefinitionSource
106                     .getAllConfigAttributes();
107             this.hasUrlAttribut = filterSecurityInterceptor
108                     .getSecurityMetadataSource().getAllConfigAttributes();
109             refresh();
110         }
111
112         public void refresh() {
113             if (filterSecurityInterceptor != null) {
114                 FilterInvocationSecurityMetadataSource source = resourceDetailsBuilder
115                         .createUrlSource(queryUrl);
116                 source.getAllConfigAttributes().addAll(hasUrlAttribut);
117                 filterSecurityInterceptor.setSecurityMetadataSource(source);
118             }
119             if (delegatingMethodDefinitionSource != null) {
120                 MethodSecurityMetadataSource source = resourceDetailsBuilder
121                         .createMethodSource(queryMethod);
122                 delegatingMethodDefinitionSource.getMethodSecurityMetadataSources()
123                         .clear();
124                 delegatingMethodDefinitionSource.getMethodSecurityMetadataSources()
125                         .add(source);
126                 List<MethodSecurityMetadataSource> list = new
127 ArrayList<MethodSecurityMetadataSource>();
128                 source.getAllConfigAttributes().addAll(hasMethodAttribute);
129                 list.add(source);
130                 delegatingMethodDefinitionSource = new DelegatingMethodSecurityMetadataSource(
131                         list);
132                 ;
133                 methodSecurityInterceptor
134                         .setSecurityMetadataSource(delegatingMethodDefinitionSource);
135                 try {
136                     // methodSecurityInterceptor.afterPropertiesSet();
137                 } catch (Exception e) {
138                     // TODO Auto-generated catch block
139                     e.printStackTrace();
140                 }
141             }
142
143             System.out.println("-------------------------refresh");
144         }
145
146         public AccessDecisionManager getAccessDecisionManager() {
147             return accessDecisionManager;
148         }
149
150         public void setAccessDecisionManager(
151                 AccessDecisionManager accessDecisionManager) {
152             this.accessDecisionManager = accessDecisionManager;
153         }
154
155         public MethodSecurityInterceptor getMethodSecurityInterceptor() {
156             return methodSecurityInterceptor;
157         }
158
159         public void setMethodSecurityInterceptor(
                    MethodSecurityInterceptor methodSecurityInterceptor) {
                this.methodSecurityInterceptor = methodSecurityInterceptor;
            }

    }
```

扩展决策机制，如果用户有多个角色，只要有一个决色拥有权限则通过校验.MyRoleVoter

```
package com.dev26.springsecuritybook;

import java.util.Collection;

import org.springframework.security.access.AccessDecisionVoter;
import org.springframework.security.access.ConfigAttribute;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;

public class MyRoleVoter implements AccessDecisionVoter<Object> {
    private String rolePrefix = "ROLE_";

    public String getRolePrefix() {
        return rolePrefix;
    }

    public void setRolePrefix(String rolePrefix) {
        this.rolePrefix = rolePrefix;
    }

    public boolean supports(ConfigAttribute attribute) {
        if ((attribute.getAttribute() != null)
                && attribute.getAttribute().startsWith(getRolePrefix())) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * This implementation supports any type of class, because it does not
query
     * the presented secure object.
     *
     * @param clazz
     *            the secure object
     *
     * @return always <code>true</code>
     */
    public boolean supports(Class<?> clazz) {
        return true;
    }

    public int vote(Authentication authentication, Object object,
            Collection<ConfigAttribute> attributes) {
        int result = ACCESS_ABSTAIN;
        Collection<? extends GrantedAuthority> authorities =
extractAuthorities(authentication);
        for (ConfigAttribute attribute : attributes) {
            if (this.supports(attribute)) {
                result = ACCESS_DENIED;
                // Attempt to find a matching granted authority
                for (GrantedAuthority authority : authorities) {
                    String[] roles = attribute.getAttribute().split(",");
                    for (String string : roles) {
                        if (string.equals(authority.getAuthority())) {
                            return ACCESS_GRANTED;
                        }
                    }

                }
            }
        }
        return result;
    }

    Collection<? extends GrantedAuthority> extractAuthorities(
            Authentication authentication) {
        return authentication.getAuthorities();
    }

}
```

以下是业务类，被保护的方法已经在上面的SQL脚本中进行了初始化：

```
package com.dev26.springsecuritybook;

public class MessageServiceImpl implements MessageService {
    public String adminMessage() {
        return "admin message";
    }

```

```
    public String adminDate() {
        return "admin " + System.currentTimeMillis();
    }

    public String userMessage() {
        return "user message";
    }

    public String userDate() {
        return "user " + System.currentTimeMillis();
    }
}
```

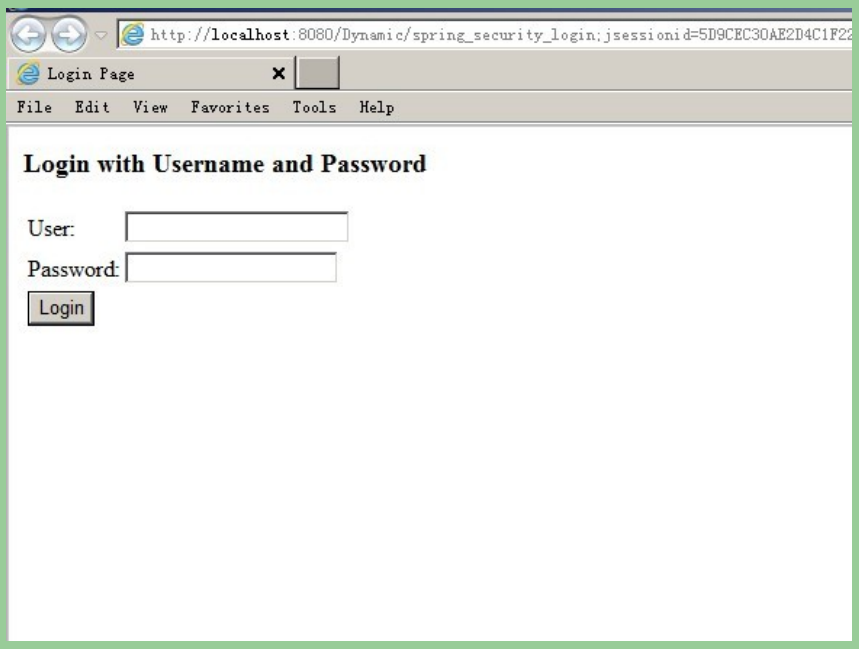
让我来验证一下，首先要编写一个登录页面:login.jsp

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page pageEncoding="UTF-8" %>

<html>
  <head>
    <title>Login</title>
  </head>

  <body onload="document.f.j_username.focus();">
    <h1>Login</h1>

    <p>Valid users:
    <p>
    <p>username <b>rod</b>, password <b>koala</b>
    <p>username <b>dianne</b>, password <b>emu</b>
    <p>username <b>scott</b>, password <b>wombat</b>
    <p>username <b>peter</b>, password <b>opal</b> (user disabled)
    <p>username <b>bill</b>, password <b>wombat</b>
    <p>username <b>bob</b>, password <b>wombat</b>
    <p>username <b>jane</b>, password <b>wombat</b>
    <p>

    <p>Locale is: <%= request.getLocale() %></p>
    <%-- this form-login-page form is also used as the
         form-error-page to ask for a login again.
         --%>
    <c:if test="${not empty param.login_error}">
      <font color="red">
        Your login attempt was not successful, try again.<br/><br/>
        Reason: <c:out value="${SPRING_SECURITY_LAST_EXCEPTION.message}"/>.
      </font>
    </c:if>

    <form name="f" action="<c:url value='j_spring_security_check'/>"
method="POST">
      <table>
        <tr><td>User:</td><td><input type='text' name='j_username'
value='<c:if test="${not empty param.login_error}"><c:out
value="${SPRING_SECURITY_LAST_USERNAME}"/></c:if>'/></td></tr>
        <tr><td>Password:</td><td><input type='password'
name='j_password'></td></tr>
        <tr><td><input type="checkbox"
name="_spring_security_remember_me"></td><td>Don't ask for my password for
two weeks</td></tr>

        <tr><td colspan='2'><input name="submit" type="submit"></td></tr>
        <tr><td colspan='2'><input name="reset" type="reset"></td></tr>
      </table>

    </form>

  </body>
</html>
```

如果通过校验则进入idnex.jsp

```
<%@ taglib prefix="security"
uri="http://www.springframework.org/security/tags" %>
<div>username :<security:authentication
property="principal.username"/></div>
<hr>
<a href="admin.jsp">admin.jsp</a>
|
<a href="admin-method.jsp">admin-method.jsp</a>
|
<a href="user-method.jsp">user-method.jsp</a>
|
<a href="j_spring_security_logout">logout</a>
```

如果你是用普通户登录的，如果进行admin-method.jsp则会被拦截,user-method.jsp则可以访问，如果有物是admin的角色由都可以进行访问.



如果点击admin.jsp

评分

请登录后再对此文章评分

列出所有评论

viano发表于2012-05-18 13:50

refresh 刷新是指的什么刷新？

undead发表于2012-05-23 12:16

是内存中的权限缓存，这样设置的权限可以即时生效

samng508发表于2012-08-01 09:50

大大共享下源码可以么？

undead发表于2012-08-01 13:52

上面已经是源码了哟，只是没有JAR包而已，你下载一下吧

samng508发表于2012-08-02 15:01

上面的代码找不到ResourceDetailsBuilder这个类

共**5**项，**1**页： 上一页 1 下一页

发表您的评论

您尚未登录，无法发表评论

立刻登录

为什么要登录后才能发表评论？

我们发现许多网站的大多数恶意评论（人身攻击，广告等）都是以匿名形式发布的，这样难以使一个技术社区形成一个和谐的讨论氛围。我们鼓励网友提出中肯的评论，留下您的身份也使得作者更容易与您交流。