导航

BlogJava
首页
新随笔
联系
XML聚合
管理

< 2012年3月 >

日	一	二	三	四	五	六
26	27	28	29	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

公告

Sparta-紫杉

PMP（项目管理专业人士）；
历任Java程序开发、系统分析、部门经理之职；
目前专注于J2EE相关技术研究及应用、SSH2框架研究与应用。

请通过QQ22086526 或SpartaYew@163.com与我联系！

常用链接

我的随笔
我的评论
我的参与
最新评论

留言簿

给我留言
查看公开留言
查看私人留言

随笔分类

Database(rss)
Java(5)(rss)

春天的故事－Spring Security3十五日研究

sparta-紫杉 2011-4-2 22:00

前言

南朝《述异记》中记载，晋王质上山砍柴，见二童子下棋，未看完，斧柄已烂，下山回村，闻同代人都去世了，自己还未变老。
因此发出“山中方一日，世上几千年” 的慨叹。原文寥寥几笔，读来却发人深省。

另有宋朝周敦颐在《暮春即事》中也有诗云：双双瓦雀行书案，点点杨花入砚池。闲坐小窗读周易，不知春去几多时。

上述古文或古诗中对于时间的论述最符合我现在的感受。已经整整十五日，对于Spring Serurity3的研究终于可以告一个段落了。

感觉这过往的十五日仿佛一瞬间而过，我沉浸在此中，一种强烈的求知愿望使我乐此不疲。到今天为止，终于将一种版本调通，可以正常使用了。
再回头看时，楼下的小公园里已经开放了一丛丛黄色的花朵，整齐的柳树丛林里，已被春风剪出嫩绿的风景。
再回头看自己其间的经历，有几多所得做为铺垫，所有的痛苦和疲惫已经烟消云散了。
欣喜之余整理一下，一来梳理一下自己的认知，进一步深入的理解；二来可以记录下来作为参考。

使用Spring Security3的四种方法概述

那么在Spring Security3的使用中，有4种方法：

一种是全部利用配置文件，将用户、权限、资源(url)硬编码在xml文件中，已经实现过，并经过验证；

二种是用户和权限用数据库存储，而资源(url)和权限的对应采用硬编码配置，目前这种方式已经实现，并经过验证。

三种是细分角色和权限，并将用户、角色、权限和资源均采用数据库存储，并且自定义过滤器，代替原有的FilterSecurityInterceptor过滤器，
并分别实现AccessDecisionManager、InvocationSecurityMetadataSourceService和UserDetailsService，并在配置文件中进行相应配置。
目前这种方式已经实现，并经过验证。

四是修改spring security的源代码，主要是修改InvocationSecurityMetadataSourceService和UserDetailsService两个类。
前者是将配置文件或数据库中存储的资源(url)提取出来加工成为url和权限列表的Map供Security使用，后者提取用户名和权限组成一个完整的(UserDetails)User对象，该对象可以提供用户的详细信息供AuthentationManager进行认证与授权使用。
该方法理论上可行，但是比较暴力，也没有时间实现，未验证，以后再研究。

说明一下，我目前调通的环境为： java1.6 + struts2.1.6 + spring3.0.1 + hibernate3.3.1 + spring security3.0.2 + oracle9i + weblogic10.3，
顺便提一下，目前（2011-4-2）serutity的最新版本为3.1，比较稳定的版本为3.0.5和2.0.6。

当然在进行spring security3的下面4种方法介绍之前，先假定SSH2的环境已经配置完毕，进入正常开发的过程，并且已经导入
spring security3.0.2的5个jar包，分别为：
spring-security-acl-3.0.2.RELEASE.jar
spring-security-config-3.0.2.RELEASE.jar
spring-security-core-3.0.2.RELEASE.jar

JavaScript(2)(rss)
Manager(3)(rss)
software engineering(rss)
SSH2(9)(rss)
心湖揽翠(2)(rss)

随笔档案

2011年5月 (21)

搜索

搜索

最新评论 XML

1. re: 春天的故事－Spring Security3十五日研究
正在学习这块！辛苦了lz
2628329@qq.com
--cno
2. re: 春天的故事－Spring Security3十五日研究
求Demo楼主。。。651125352@qq.com
--jamf
3. re: 春天的故事－Spring Security3十五日研究[未登录]
忘记留邮箱了
1140384524
--我
4. re: 春天的故事－Spring Security3十五日研究[未登录]
评论内容较长,点击标题查看
--我
5. re: 春天的故事－Spring Security3十五日研究[未登录]
楼主给我发个jar包吧，老是出错hww_315@163.com,谢谢
--Bruce

阅读排行榜

1. 春天的故事－Spring Security3十五日研究(15321)
2. Struts2中以非迭代方式提取Map中的值(1969)
3. hibernate中提倡持久类实现equals()和hashCode()的原因分析(1564)
4. 365天，730个吻(1438)
5. Struts2的dojo使用与/template/ajax/head.ftl not found.(1157)

评论排行榜

1. 春天的故事－Spring Secu

spring-security-taglibs-3.0.2.RELEASE.jar
spring-security-web-3.0.2.RELEASE.jar
当然还有其他相关的jar包，在此不再赘述。

第一种方法

第一种方法比较简单，可参考Spring Security自带的例子spring-security-samples-tutorial-3.0.2.RELEASE。
这里给出下载网址：http://www.springsource.com/download/community?sid=1087087，不过在下载之前必须填写相应的用户信息，才允许下载。各种版本号的均可以下载。

在spring-security-samples-tutorial-3.0.2.RELEASE的例子里，硬编码的配置请参见applicationContext-security.xml文件中的内容。
里面配置了用户名、经过MD5加密后的密码密文、相关的权限，以及与权相对应的访问资源（URL）。还有对于Session超时时的处理。
特别是因为版本号为3.0.2，因此还增加了对表达式的配置演示，具体内容请参见该例子。

当然你最好运行起该例子来，感受一下，你可以直接将下载下来的解压缩后的文件夹中找到spring-security-samples-tutorial-3.0.2.RELEASE.war文件，然后拷贝到Tomcat的安装目录下的\webapps文件夹下，然后运行Tomcat的服务器，服务器在启动过程中，会自动解开该war文件，在IE内输入http://localhost:8080/webapps/spring-security-samples-tutorial-3.0.2.RELEASE 就可以运行该系统了。在此不再赘述。

第二种方法

第二种方法的代码如下：

使用到的两个表，用户表和权限表的SQL语句。将用户和权限以数据库进行存储。

```
create table USERS(
  USERNAME   VARCHAR2(50) not null,
  PASSWORD   VARCHAR2(50) not null,
  ENABLED    NUMBER(1) not null,
  USERNAMECN VARCHAR2(50),
  primary key( username )
)

create table AUTHORITIES(
  USERNAME  VARCHAR2(50) not null,
  AUTHORITY VARCHAR2(50) not null
)
```

-- 外键使用户和权限相联。

```
Create/Recreate primary, unique and foreign key constraints
alter table AUTHORITIES
add constraint FK_AUTHORITIES_USERS foreign key (USERNAME)
references USERS (USERNAME);
```

可插入几条数据做为试验，首先插入用户：

```
insert into users (USERNAME, PASSWORD, ENABLED, USERNAMECN, ROWID)
values ('lxb', 'c7d3f4c857bc8c145d6e5d40c1bf23d9', 1, '登录用户', 'AAAHmhAALAAAAAOAAA');

insert into users (USERNAME, PASSWORD, ENABLED, USERNAMECN, ROWID)
values ('admin', 'ceb4f32325eda6142bd65215f4c0f371', 1, '系统管理员', 'AAAHmhAALAAAAAPAAA');

insert into users (USERNAME, PASSWORD, ENABLED, USERNAMECN, ROWID)
values ('user', '47a733d60998c719cf3526ae7d106d13', 1, '普通用户', 'AAAHmhAALAAAAAPAAB');
```

再插入角色：

```
insert into authorities (USERNAME, AUTHORITY, ROWID)
values ('admin', 'ROLE_PLATFORMADMIN', 'AAAHmjAALAAAAAgAAA');

insert into authorities (USERNAME, AUTHORITY, ROWID)
values ('admin', 'ROLE_SYSADMIN', 'AAAHmjAALAAAAAgAAB');

insert into authorities (USERNAME, AUTHORITY, ROWID)
values ('lxb', 'ROLE_LOGIN', 'AAAHmjAALAAAAAeAAA');

insert into authorities (USERNAME, AUTHORITY, ROWID)
values ('lxb', 'ROLE_LOGINTOWELCOME', 'AAAHmjAALAAAAAeAAB');

insert into authorities (USERNAME, AUTHORITY, ROWID)
values ('user', 'ROLE_USER', 'AAAHmjAALAAAAAgAAC');
```

第二种方法之密码加密

可能要有人要问，用户表里面的密码是如何取得的呢？这个密码是通过MD5进行加密过的，并且以用户名做为了盐值，最后就成为32位数字这个样子，这个你可以参见下面applicationContext-Security.xml中的password-encoder和salt-source的配置就会明白。

那么在spring security3中是如何加密的呢？当我们设置了pawwrod-encoder和salt-source之后，Spring Security3会根据配置，采用相匹配的加密算法（比如设置了MD5加密算法）再加上salt-source进行加密，形成32位数字的密文。

比如用户名为yew,密码为yew1234，盐值为用户名yew。那么最后加密的明文为“yew1234{yew}”，密文就为“8fe2657d1599dba8e78a7a0bda8651bb”。

我们在试验过程中，通常喜欢先将几个常用的用户及密码插入数据库进行试验，这种情况下如何得到该用户的密码密文呢？

不妨试试我这个办法，假设，用户名为user，密码明文为user369，而且在配置文件里面设置了以MD5作为加密算法，并以用户名做为盐值。

那么你可以首先将各个信息组合成待加密的密码明文， 应是 密码明文 + { + 盐值 + }， 那么很明显，上述user的密码明文应当是：

user369{user}

拿上述的字串拷贝到 http://www.51240.com/md5jiami/ 网页上的输入框里，点击加密按钮，下面即可生成32位数字的密码密文。

哈哈，屡试不爽啊。这个方法要谨慎使用，一般人我不告诉他。

第二种方法之相关配置

将权限及资源(URL或Action)的关系配置在xml文件中，并且配置与Spring Security3相关的其他配置：

1、applicationContext-Security.xml代码：

```
<b:beans xmlns="http://www.springframework.org/schema/security"
 xmlns:b="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">

 <http auto-config="true" access-denied-page="/accessDenied.jsp">
  <!-- 不要过滤图片等静态资源，其中**代表可以跨越目录，*不可以跨越目录。 -->
  <intercept-url pattern="/**/*.jpg" filters="none" />
  <intercept-url pattern="/**/*.png" filters="none" />
  <intercept-url pattern="/**/*.gif" filters="none" />
  <intercept-url pattern="/**/*.css" filters="none" />
  <intercept-url pattern="/**/*.js" filters="none" />
  <!-- 登录页面和忘记密码页面不过滤 -->
  <intercept-url pattern="/login.jsp" filters="none" />
  <intercept-url pattern="/jsp/forgotpassword.jsp"   filters="none" />

   <!-- 下面是对Action配置。表示具有访问/unitsManager资源的用户必须具有ROLE_PLATFORMADMIN的权限。
                当用户登录时，SS3将用户的所有权限从数据库中提取出来，形成列表。 当用户访问该资源时，SS3将
                登录用户的权限列表提出来跟下面配置的权限进行比对，若有，则允许访问，若没有，则给出AccessDeniedException。-->
  <intercept-url pattern="/unitsManager"   access="ROLE_PLATFORMADMIN" />
  <intercept-url pattern="/usersManager"  access="ROLE_PLATFORMADMIN" />

  <intercept-url pattern="/horizontalQuery"  access="ROLE_PLATFORMADMIN" />

  <intercept-url pattern="/verticalQuery"    access="ROLE_PLATFORMADMIN" />

  <form-login login-page="/login.jsp"  authentication-failure-url="/login.jsp?error=true"   default-target-url="/index.jsp" />

  <!-- "记住我"功能，采用持久化策略（将用户的登录信息存放在数据库表中） -->
  <remember-me data-source-ref="dataSource" />

  <!-- 检测失效的sessionId,超时时定位到另外一个URL -->
  <session-management invalid-session-url="/sessionTimeout.jsp" />

 </http>

 <!-- 注意能够为authentication-manager 设置alias别名 -->
 <authentication-manager alias="authenticationManager">
     <authentication-provider user-service-ref="userDetailsManager">
         <password-encoder ref="passwordEncoder">
             <!-- 用户名做为盐值 -->
             <salt-source user-property="username" />
         </password-encoder>
```

```
    </authentication-provider>
 </authentication-manager>

</b:beans>
```

2、applicationContext.service.xml:

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:util="http://www.springframework.org/schema/util"
 xmlns:jee="http://www.springframework.org/schema/jee"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:context="http://www.springframework.org/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
   http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
   http://www.springframework.org/schema/jee
   http://www.springframework.org/schema/jee/spring-jee-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd
   http://www.springframework.org/schema/util
   http://www.springframework.org/schema/util/spring-util-3.0.xsd">

 <!-- 定义上下文返回的消息的国际化。 -->
 <bean id="messageSource"
  class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
  <property name="basename"
   value="classpath:org/springframework/security/messages_zh_CN"/>
 </bean>

 <!--  事件监听:实现了 ApplicationListener监听接口，包括AuthenticationCredentialsNotFoundEvent 事件，
  AuthorizationFailureEvent事件，AuthorizedEvent事件， PublicInvocationEvent事件 -->
 <bean  class="org.springframework.security.authentication.event.LoggerListener" />

 <!-- 用户的密码加密或解密 -->
 <bean id="passwordEncoder"
  class="org.springframework.security.authentication.encoding.Md5PasswordEncoder" />


 <!-- 用户详细信息管理：数据源、用户缓存、启用用户组功能。 -->
 <bean id="userDetailsManager"
  class="org.springframework.security.provisioning.JdbcUserDetailsManager">
```

```
  <property name="dataSource" ref="dataSource" />
  <property name="userCache" ref="userCache" />
 </bean>

 <bean id="userCache"
  class="org.springframework.security.core.userdetails.cache.EhCacheBasedUserCache">
  <property name="cache" ref="userEhCache" />
 </bean>



 <bean id="userEhCache" class="org.springframework.cache.ehcache.EhCacheFactoryBean">
  <property name="cacheName" value="userCache" />
  <property name="cacheManager" ref="cacheManager" />
 </bean>

 <!-- 缓存用户管理 -->
 <bean id="cacheManager"
  class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean" />


 <!-- spring security自带的与权限有关的数据读写Jdbc模板 -->
 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource" />
 </bean>

</beans>
```

3、web.xml：

```
<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">


 <!-- 设置log4j存放Log文件位置（通过spring统一进行管理） -->
 <context-param>
  <param-name>webAppRootKey</param-name>
  <param-value>log.root</param-value>
 </context-param>

 <!-- 加载log4j的配置文件 -->
 <context-param>
  <param-name>log4jConfigLocation</param-name>
  <param-value>classpath:/log4j.properties</param-value>
 </context-param>
```

```
<!--Spring默认刷新Log4j配置文件的间隔,单位为millisecond-->
<context-param>
 <param-name>log4jRefreshInterval</param-name>
 <param-value>60000</param-value>
</context-param>

<!--Spring用于log4j初始化的监听器-->
<listener>
 <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
</listener>

<!--
 加载Spring XML配置文件，Spring安全配置及各类资源文件，暂不加
 /WEB-INF/applicationContext-security.xml,
-->
<context-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>
        /WEB-INF/applicationContext*.xml,
        classpath*:applicationContext.xml
      </param-value>
</context-param>

<!--spring监听器的配置，用于在启动Web容器时，自动装配ApplicationContext的配置信息-->
<listener>
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!-- 使用Spring中的过滤器解决在请求和应答中的中文乱码问题 -->
<filter>
 <filter-name>characterEncodingFilter</filter-name>
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
 <init-param>
  <param-name>encoding</param-name>
  <param-value>gbk</param-value>
 </init-param>
 <init-param>
  <!--强制转换编码(request和response均适用) -->
  <param-name>ForceEncoding</param-name>
  <param-value>true</param-value>
 </init-param>
</filter>

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

```
</filter-mapping>


<!-- Spring Secutiry3.0.2的过滤器链配置 -->
<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>


  <!-- 配置Struts2的FilterDispathcer的Filter -->
  <filter>
     <filter-name>struts2</filter-name>
     <filter-class>
      org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
     </filter-class>
  </filter>

<!-- struts2用以处理用户Web请求的路径模式-->
  <filter-mapping>
     <filter-name>struts2</filter-name>
     <url-pattern>/*</url-pattern>
  </filter-mapping>



<!-- 避免乱码问题 -->
<filter>
     <filter-name>struts-cleanup</filter-name>
     <filter-class>
        org.apache.struts2.dispatcher.ActionContextCleanUp
     </filter-class>
  </filter>

  <filter-mapping>
     <filter-name>struts-cleanup</filter-name>
     <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!-- Spring刷新Interceptor防止内存泄漏 -->
  <listener>
   <listener-class>
```

```
    org.springframework.web.util.IntrospectorCleanupListener
    </listener-class>
   </listener>


  <!-- 设置session 超时时间为20分钟 -->
  <session-config>
   <session-timeout>20</session-timeout>
  </session-config>

  <!--系统欢迎页面-->
  <welcome-file-list>
   <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>

 </web-app>
```

令人欣喜的是，整个Security配置过程中，除了建立数据库和编写配置文件之外，不需要编写任何的代码。怎么样？ 有点意思吧!

第二种方法中遇见的问题

当然，首次使用Spring serutiry，在整合的过程中，我还是遇见了不少问题，当然有些问题比如找不到类呀，包呀，和框架的整合呀等问题不作为谈论的重点。主要还是探讨Spring Security的配置和注意事项的问题。

我在其中碰到的对我印象最深的问题是，当完全配置好之后，重启Web服务器，却发现Spring Security不能拦截任何的URL了，这使我感到惊诧，因为在去年时，我已经将该框架搭建完成，在当时正是使用的该种方法，并且在试验是否能够拦截jsp文件时进行了确认是没有问题的。

接下来我又整理了一下applicationContext-security.xml的文件才发现， 除了不需要进行检测的图片及登录页面之外，没有对任何的资源和权限之间的对应关系进行配置，参见下面的代码：

```
 <http auto-config="true" access-denied-page="/accessDenied.jsp">
  <!-- 不要过滤图片等静态资源，其中**代表可以跨越目录，*不可以跨越目录。 -->
  <intercept-url pattern="/**/*.jpg" filters="none" />
  <intercept-url pattern="/**/*.png" filters="none" />
  <intercept-url pattern="/**/*.gif" filters="none" />
  <intercept-url pattern="/**/*.css" filters="none" />
  <intercept-url pattern="/**/*.js" filters="none" />
  <!-- 登录页面和忘记密码页面不过滤 -->
  <intercept-url pattern="/login.jsp" filters="none" />
  <intercept-url pattern="/jsp/forgotpassword.jsp" filters="none" />

        <!-- 下面是对Struts2的Action请求时的配置。注意在前面加/，否则不会被SS3进行拦截验证。
            表示具有访问/unitsManager资源的用户必须具有ROLE_PLATFORMADMIN的权限。
            当用户登录时，SS3将用户的所有权限从数据库中提取出来，形成列表。 当用户访问该资源时，
            SS3将登录用户的权限列表提出来跟下面配置的权限进行比对，若有，则允许访问，若没有，
            则给出AccessDeniedException。
  <intercept-url pattern="/unitsManager"  access="ROLE_PLATFORMADMIN" />
  <intercept-url pattern="/usersManager"  access="ROLE_PLATFORMADMIN" />
```

```
    <intercept-url pattern="/horizontalQuery" access="ROLE_PLATFORMADMIN" />
    <intercept-url pattern="/verticalQuery"  access="ROLE_PLATFORMADMIN" />   -->

    <form-login login-page="/login.jsp"
     authentication-failure-url="/login.jsp?error=true"
     default-target-url="/index.jsp" />

    <!-- "记住我"功能，采用持久化策略（将用户的登录信息存放在数据库表中） -->
    <remember-me data-source-ref="dataSource" />

    <!-- 检测失效的sessionId,超时时定位到另外一个URL -->
    <session-management invalid-session-url="/sessionTimeout.jsp" />

  </http>
```

这样一来，spring security3就会认为根本不需要对任何的URL或Action进行检测(注意上面代码中被注释掉的4条配置)。 哈哈，当时这个问题深深动摇了我对Spring security的信心，花费了这么多天的精力，却是这样的结果，当时就在考虑是否有更好的替代品。有点崩溃啊。 还好，深深地求知欲和征服欲让我坚持下来了。

哈哈，这算不算Spring Security的一个Bug呢？没有任何的权限与资源的配置，就认为登录后的用户具有访问任何资源的权限，说起来有点可怕哈。

当然，当我将上述代码中被注释的4条配置放开后，Spring security奇迹般的恢复了活力。

接下来实现了jsp型URL的拦截之后，我又遇见了不能拦截action的情况，不过经过多次的配置和重启服务试验，终于发现，在配置Action与权限时，一定要在Action的路径前面加“/”斜杠，否则，Spring Security就会对该请求的URL熟视无睹，无视它的存在，即使你在Action的前后加上*号进行匹配也不会起任何作用，哈哈，不禁慨叹Spring Security的牛脾气。

第二种方法BTW

顺便提一下子，Spring Security3需要配置的过滤器是双重的，首先在web.xml中配置一个过滤器代理，参见上述web.xml中的springSecurityFilterChain配置。

我们通常设置过滤的url模式为/*，就是说任何的url访问都要进行过滤，工作量有点大哈。当然我们可以为之设置不同的过滤url模式，比如.action、.do、.jsp等。这样的话，遇到.action或.jsp或.do结尾的url访问，Spring Security就会突然站出来打截，若是其他的访问，Spring Security就会挥一挥手，潇洒地让你路过。

所以说，这个过滤器主要对大的方面进行拦截，一些细小的活儿，还是要交给第二重过滤器。 就是说，这第一重过滤器是个总代理，他威武地管理着一个过滤器链。

那么这第二重过滤器的配置，就是那些所谓的过滤器链，分别包括“记住我”、“登录”、“注销”、“url访问”等的过滤器，这个过滤器依顺序排开，形成一个过滤链条。具体拦截我们明细Url的是一个叫做FilterInterCeptor的伙计，我认为这个家伙是在整个过滤器链条中是最重要的一个，因为我们登录系统之后，要访问的任何资源都必须经得他的同意。 那么这第二重链条就设置在applicationContext-security.xml文件中的<http>元素下面。

什么，你看不到？ 忘记告诉你了，从spring security2开始，就使用了命名空间，若你在<http>中设置了auto="true"，Spring Security就会在服务启动时自动加载

所有的过滤器链，省事了吧！

第三种方法

当然，spring security3毕竟是西方国家的东西，以英文为主，使用习惯和文化的差异共存，况且为了适应大多数Web应用的权限管理，作者将Spring

Security3打造的精简而灵活。精简指Spring Security3对用户和权限的表设计的非常简单，并且没有采用数据库来管理资源（URL）。这样的话，对于我们国人用户来说，是个很大的遗憾，这个遗憾甚至能够影响到我们对安全框架的选型。你想啊，在国内大多数项目中，均设置了比较复杂的权限控制，一般就会涉及到用户、角色、权限、资源4张表，若要加上4张表之间的对应关系表3张，得有7张表才行。

得7张表才行，但是Spring Security3才给我们提供了2张最简洁的表，这足以不能完成国人用户的项目应用。那么在对Spring Security3一无所知的情况下，
我们很容易就会放弃对该安全框架的选型。

还好，Spring Security3提供了灵活的扩展方法。具体应该扩展哪些类呢？ 或者到底Spring Security3工作的流程如何，你不妨参看下面一篇文章，就会获得
一些启示，网址为：http://www.blogjava.net/youxia/archive/2008/12/07/244883.html ， 哈哈，谢谢分享。

还有一个地址很有价值， http://wenku.baidu.com/view/4ec7e324ccbff121dd368364.html ，我就参考着上面的介绍扩展了4个类。

不过我得提一下，原文的作者为了考验你的耐性和自信心，故意在代码里面卖了几点小小的关子，因此若是完全按照作者的原文代码装配起来的权限系统，是不会那么顺利地工作的，天下似乎真是没有不花费力气的午餐！在装配完成后，我也是经过九九八十一难的折磨，在用户、角色、权限、资源的“天下黄河九曲十八弯”里面盘旋迂回，终于到达了成功的彼岸。至此才对Spring Security有了更深层次的理解，更加佩服作者的良苦用心。 哈哈。

并扩展了User类以增加其相关的各类其他信息(如Email，职务，所在单位id等)。

相关的代码如下（包含5个关键类）：

```
/*
 * @(#) MyFilterSecurityInterceptor.java  2011-3-23 上午07:53:03
 *
 * Copyright 2011 by Sparta
 */

package avatar.base.security;

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.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;

/**
```

```
 * 该过滤器的主要作用就是通过spring著名的IoC生成securityMetadataSource。
 * securityMetadataSource相当于本包中自定义的MyInvocationSecurityMetadataSourceService。
 * 该MyInvocationSecurityMetadataSourceService的作用提从数据库提取权限和资源，装配到HashMap中，
 * 供Spring Security使用，用于权限校验。
 * @author sparta 11/3/29
 *
 */

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){
     this.securityMetadataSource = securityMetadataSource;
    }


    public void destroy(){

    }

    public void init( FilterConfig filterconfig ) throws ServletException{

    }


}



/*
 * @(#) MyInvocationSecurityMetadataSourceService.java  2011-3-23 下午02:58:29
 *
 * Copyright 2011 by Sparta
 */

package avatar.base.security;

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 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.core.GrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
```

```
import org.springframework.security.core.userdetails.UserDetails;
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 avatar.base.security.dao.PubAuthoritiesResourcesHome;

/**
 * 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。 此类在初始化时，应该取到所有资源及其对应角色的定义。
 *
 */
@Service
public class MyInvocationSecurityMetadataSourceService implements
    FilterInvocationSecurityMetadataSource {

  @Autowired
  private PubAuthoritiesResourcesHome pubAuthoritiesResourcesHome;

  private UrlMatcher urlMatcher = new AntUrlPathMatcher();

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

  public MyInvocationSecurityMetadataSourceService() {
    loadResourceDefine();
  }

  private void loadResourceDefine() {
    ApplicationContext context = new ClassPathXmlApplicationContext(
        "classpath:applicationContext.xml");

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

    Session session = sessionFactory.openSession();

    String username = "";
    String sql = "";

    // 在Web服务器启动时，提取系统中的所有权限。
    sql = "select authority_name from pub_authorities";

    List<String> query = session.createSQLQuery(sql).list();

    /*
     * 应当是资源为key， 权限为value。 资源通常为url， 权限就是那些以ROLE_为前缀的角色。 一个资源可以由多个权限来访问。
     * sparta
     */
```

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

   for (String auth : query) {
    ConfigAttribute ca = new SecurityConfig(auth);

    List<String> query1 = session
      .createSQLQuery(
        "select b.resource_string "
          + "from Pub_Authorities_Resources a, Pub_Resources b, "
          + "Pub_authorities c where a.resource_id = b.resource_id "
          + "and a.authority_id=c.authority_id and c.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，如果已经存在相关的资源url，则要通过该url为key提取出权限集合，将权限增加到权限集合中。
      * sparta
      */
     if (resourceMap.containsKey(url)) {

      Collection<ConfigAttribute> value = resourceMap.get(url);
      value.add(ca);
      resourceMap.put(url, value);
     } else {
      Collection<ConfigAttribute> atts = new ArrayList<ConfigAttribute>();
      atts.add(ca);
      resourceMap.put(url, atts);
     }

    }

   }

  }

  @Override
  public Collection<ConfigAttribute> getAllConfigAttributes() {

   return null;
  }

  // 根据URL，找到相关的权限配置。
  @Override
  public Collection<ConfigAttribute> getAttributes(Object object)
    throws IllegalArgumentException {
```

```
    // object 是一个URL，被用户请求的url。
    String url = ((FilterInvocation) object).getRequestUrl();

        int firstQuestionMarkIndex = url.indexOf("?");

        if (firstQuestionMarkIndex != -1) {
            url = url.substring(0, firstQuestionMarkIndex);
        }

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

    while (ite.hasNext()) {
     String resURL = ite.next();

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

      return resourceMap.get(resURL);
     }
    }

    return null;
}

    @Override
public boolean supports(Class<?> arg0) {

    return true;
}

}

/*
 * @(#) MyUserDetailsService.java  2011-3-23 上午09:04:31
 *
 * Copyright 2011 by Sparta
 */

package avatar.base.security;


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


import javax.sql.DataSource;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
```

```java
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserCache;
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 avatar.base.security.dao.PubAuthoritiesResourcesHome;
import avatar.base.security.dao.PubUsersHome;


/**
 *该类的主要作用是为Spring Security提供一个经过用户认证后的UserDetails。
 *该UserDetails包括用户名、密码、是否可用、是否过期等信息。
 *sparta 11/3/29
 */
@Service
public class MyUserDetailsService implements UserDetailsService {

    @Autowired
    private PubUsersHome pubUsersHome;

    @Autowired
    private PubAuthoritiesResourcesHome pubAuthoritiesResourcesHome;

    @Autowired
    private DataSource dataSource;

    @Autowired
    private UserCache userCache;

    @Override
    public UserDetails loadUserByUsername(String username)
        throws UsernameNotFoundException, DataAccessException {

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


        //得到用户的权限
        auths = pubUsersHome.loadUserAuthoritiesByName( username );

        String password = null;

        //取得用户的密码
        password = pubUsersHome.getPasswordByUsername( username );
```

```
        return new User( username, password, true, "", true, true, true, auths);
    }

    //set PubUsersHome
    public void setPubUsersHome( PubUsersHome pubUsersHome ){
        this.pubUsersHome = pubUsersHome;

    }

    public PubUsersHome getPubUsersHome(){
        return pubUsersHome;
    }


    //set PubAuthoritiesResourcesHome
    public void setPubAuthoritiesResourcesHome( PubAuthoritiesResourcesHome pubAuthoritiesResourcesHome ){
        this.pubAuthoritiesResourcesHome = pubAuthoritiesResourcesHome;

    }

    public PubAuthoritiesResourcesHome getPubAuthoritiesResourcesHome(){
        return pubAuthoritiesResourcesHome;

    }

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

    public DataSource getDataSource(){
        return dataSource;
    }

    //设置用户缓存功能。
      public void setUserCache(UserCache userCache) {
          this.userCache = userCache;
      }

      public UserCache getUserCache(){
       return this.userCache;
      }

}

/*
```

```
 * @(#) MyAccessDecisionManager.java  2011-3-23 下午04:41:12
 *
 * Copyright 2011 by Sparta
 */

package avatar.base.security;

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;

/**
 *AccessdecisionManager在Spring security中是很重要的。
 *
 *在验证部分简略提过了，所有的Authentication实现需要保存在一个GrantedAuthority对象数组中。
 *这就是赋予给主体的权限。 GrantedAuthority对象通过AuthenticationManager
 *保存到 Authentication对象里，然后从AccessDecisionManager读出来，进行授权判断。
 *
 *Spring Security提供了一些拦截器，来控制对安全对象的访问权限，例如方法调用或web请求。
 *一个是否允许执行调用的预调用决定，是由AccessDecisionManager实现的。
 *这个 AccessDecisionManager 被AbstractSecurityInterceptor调用，
 *它用来作最终访问控制的决定。 这个AccessDecisionManager接口包含三个方法:
 *
 void decide(Authentication authentication, Object secureObject,
    List<ConfigAttributeDefinition> config) throws AccessDeniedException;
 boolean supports(ConfigAttribute attribute);
 boolean supports(Class clazz);

 从第一个方法可以看出来，AccessDecisionManager使用方法参数传递所有信息，这好像在认证评估时进行决定。
 特别是，在真实的安全方法期望调用的时候，传递安全Object启用那些参数。
 比如，让我们假设安全对象是一个MethodInvocation。
 很容易为任何Customer参数查询MethodInvocation，
 然后在AccessDecisionManager里实现一些有序的安全逻辑，来确认主体是否允许在那个客户上操作。
 如果访问被拒绝，实现将抛出一个AccessDeniedException异常。

 这个 supports(ConfigAttribute) 方法在启动的时候被
 AbstractSecurityInterceptor调用，来决定AccessDecisionManager
 是否可以执行传递ConfigAttribute。
 supports(Class)方法被安全拦截器实现调用，
 包含安全拦截器将显示的AccessDecisionManager支持安全对象的类型。
```

```
 */
public class MyAccessDecisionManager implements AccessDecisionManager {

 public void decide( Authentication authentication, Object object,
  Collection<ConfigAttribute> configAttributes)
 throws AccessDeniedException, InsufficientAuthenticationException{

 if( configAttributes == null ) {
  return ;
 }

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

 while( ite.hasNext()){

  ConfigAttribute ca = ite.next();
  String needRole = ((SecurityConfig)ca).getAttribute();

  //ga 为用户所被赋予的权限。 needRole 为访问相应的资源应该具有的权限。
  for( GrantedAuthority ga: authentication.getAuthorities()){

   if(needRole.trim().equals(ga.getAuthority().trim())){

    return;
   }

  }

 }

 throw new AccessDeniedException("");

}

public boolean supports( ConfigAttribute attribute ){

 return true;

}

public boolean supports(Class<?> clazz){
 return true;

}


}
```

数据库的SQL及预置数据：

```
prompt PL/SQL Developer import file
prompt Created on 2011年6月1日 by Administrator
set feedback off
set define off
prompt Creating SYS_AUTHORITIES...
create table SYS_AUTHORITIES
(
  AUTHORITY_ID   VARCHAR2(32) not null,
  AUTHORITY_NAME VARCHAR2(40),
  AUTHORITY_DESC VARCHAR2(100),
  ENABLED        NUMBER(1),
  ISSYS          NUMBER(1),
  MODULE         VARCHAR2(4)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
comment on table SYS_AUTHORITIES
  is '权限表';
comment on column SYS_AUTHORITIES.MODULE
  is '所属的子系统，比如平台里面包括10个系统，分别为成本、作业、集输等。';
alter table SYS_AUTHORITIES
  add constraint PK_PUB_AUTHORITIES primary key (AUTHORITY_ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
```

```
prompt Creating SYS_RESOURCES...
create table SYS_RESOURCES
(
  RESOURCE_ID     VARCHAR2(32) not null,
  RESOURCE_NAME   VARCHAR2(100),
  RESOURCE_DESC   VARCHAR2(100),
  RESOURCE_TYPE   VARCHAR2(40),
  RESOURCE_STRING VARCHAR2(200),
  PRIORITY        NUMBER(1),
  ENABLED         NUMBER(1),
  ISSYS           NUMBER(1),
  MODULE          VARCHAR2(4)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
comment on table SYS_RESOURCES
  is '资源表';
comment on column SYS_RESOURCES.PRIORITY
  is '（暂不用，保留）';
comment on column SYS_RESOURCES.MODULE
  is '所属的子系统，比如平台里面包括10个系统，分别为成本、作业、集输等。 （暂不用，保留）';
alter table SYS_RESOURCES
  add constraint PK_PUB_RESOURCES primary key (RESOURCE_ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );

prompt Creating SYS_AUTHORITIES_RESOURCES...
create table SYS_AUTHORITIES_RESOURCES
```

```
(
  ID          NUMBER(13) not null,
  AUTHORITY_ID VARCHAR2(32),
  RESOURCE_ID  VARCHAR2(32),
  ENABLED      NUMBER(1)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
comment on table SYS_AUTHORITIES_RESOURCES
  is '权限资源表';
alter table SYS_AUTHORITIES_RESOURCES
  add constraint PK_PUB_AUTHORITIES_RE primary key (ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
alter table SYS_AUTHORITIES_RESOURCES
  add constraint FK_PUB_AUTHORITIES_RE_AU foreign key (AUTHORITY_ID)
  references SYS_AUTHORITIES (AUTHORITY_ID);
alter table SYS_AUTHORITIES_RESOURCES
  add constraint FK_PUB_AUTHORITIES_RE_RE foreign key (RESOURCE_ID)
  references SYS_RESOURCES (RESOURCE_ID);

prompt Creating SYS_ROLES...
create table SYS_ROLES
(
  ROLE_ID   VARCHAR2(32) not null,
  ROLE_NAME VARCHAR2(40),
  ROLE_DESC VARCHAR2(100),
  ENABLED   NUMBER(1),
  ISSYS     NUMBER(1),
```

```
  MODULE    VARCHAR2(4)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
comment on table SYS_ROLES
  is '角色表';
comment on column SYS_ROLES.MODULE
  is '所属的子系统，比如平台里面包括10个系统，分别为成本、作业、集输等。';
alter table SYS_ROLES
  add constraint PK_PUB_ROLES primary key (ROLE_ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );

prompt Creating SYS_ROLES_AUTHORITIES...
create table SYS_ROLES_AUTHORITIES
(
  ID          NUMBER(13) not null,
  ROLE_ID     VARCHAR2(32),
  AUTHORITY_ID VARCHAR2(32),
  ENABLED     NUMBER(1)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
```

```
    minextents 1
    maxextents unlimited
  );
comment on table SYS_ROLES_AUTHORITIES
  is '角色权限表';
alter table SYS_ROLES_AUTHORITIES
  add constraint PK_PUB_ROLES_AUTHORITY primary key (ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
alter table SYS_ROLES_AUTHORITIES
  add constraint FK_PUB_ROLES_AUTHORITIES_AU foreign key (AUTHORITY_ID)
  references SYS_AUTHORITIES (AUTHORITY_ID);
alter table SYS_ROLES_AUTHORITIES
  add constraint FK_PUB_ROLES_AUTHORITIES_ROLES foreign key (ROLE_ID)
  references SYS_ROLES (ROLE_ID);

prompt Creating SYS_USERS...
create table SYS_USERS
(
  USER_ID       VARCHAR2(32) not null,
  USER_ACCOUNT  VARCHAR2(30),
  USER_NAME     VARCHAR2(40),
  USER_PASSWORD VARCHAR2(100),
  USER_DESC     VARCHAR2(100),
  ENABLED       NUMBER(1),
  ISSYS         NUMBER(1),
  USER_DEPT     VARCHAR2(20),
  USER_DUTY     VARCHAR2(10),
  SUB_SYSTEM    VARCHAR2(30)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
```

```
    minextents 1
    maxextents unlimited
  );
comment on table SYS_USERS
  is '用户表';
comment on column SYS_USERS.USER_PASSWORD
  is '该密码是经加盐值加密的，格式为password{username}。 比如用户的密码为user，用户名为user，那么通过MD5进行加密的串为： user{user}';
comment on column SYS_USERS.ISSYS
  is '是否是超级用户';
comment on column SYS_USERS.USER_DEPT
  is '所在单位';
comment on column SYS_USERS.USER_DUTY
  is '经理或主任';
comment on column SYS_USERS.SUB_SYSTEM
  is '该用户所负责的各子系统，可多个，中间用逗号分隔。(目前暂未用，作为保留字段)';
alter table SYS_USERS
  add constraint PK_PUB_USERS primary key (USER_ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );

prompt Creating SYS_USERS_ROLES...
create table SYS_USERS_ROLES
(
  ID      NUMBER(13) not null,
  USER_ID VARCHAR2(32),
  ROLE_ID VARCHAR2(32),
  ENABLED NUMBER(1)
)
tablespace SCJD
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
```

```
  );
comment on table SYS_USERS_ROLES
  is '用户角色表';
alter table SYS_USERS_ROLES
  add constraint PK_PUB_USERS_ROLES primary key (ID)
  using index
  tablespace SCJD
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
alter table SYS_USERS_ROLES
  add constraint FK_USERS_ROLES_ROLES foreign key (ROLE_ID)
  references SYS_ROLES (ROLE_ID);
alter table SYS_USERS_ROLES
  add constraint FK_USERS_ROLES_USERS foreign key (USER_ID)
  references SYS_USERS (USER_ID);

prompt Disabling triggers for SYS_AUTHORITIES...
alter table SYS_AUTHORITIES disable all triggers;
prompt Disabling triggers for SYS_RESOURCES...
alter table SYS_RESOURCES disable all triggers;
prompt Disabling triggers for SYS_AUTHORITIES_RESOURCES...
alter table SYS_AUTHORITIES_RESOURCES disable all triggers;
prompt Disabling triggers for SYS_ROLES...
alter table SYS_ROLES disable all triggers;
prompt Disabling triggers for SYS_ROLES_AUTHORITIES...
alter table SYS_ROLES_AUTHORITIES disable all triggers;
prompt Disabling triggers for SYS_USERS...
alter table SYS_USERS disable all triggers;
prompt Disabling triggers for SYS_USERS_ROLES...
alter table SYS_USERS_ROLES disable all triggers;
prompt Disabling foreign key constraints for SYS_AUTHORITIES_RESOURCES...
alter table SYS_AUTHORITIES_RESOURCES disable constraint FK_PUB_AUTHORITIES_RE_AU;
alter table SYS_AUTHORITIES_RESOURCES disable constraint FK_PUB_AUTHORITIES_RE_RE;
prompt Disabling foreign key constraints for SYS_ROLES_AUTHORITIES...
alter table SYS_ROLES_AUTHORITIES disable constraint FK_PUB_ROLES_AUTHORITIES_AU;
alter table SYS_ROLES_AUTHORITIES disable constraint FK_PUB_ROLES_AUTHORITIES_ROLES;
```

```
prompt Disabling foreign key constraints for SYS_USERS_ROLES...
alter table SYS_USERS_ROLES disable constraint FK_USERS_ROLES_ROLES;
alter table SYS_USERS_ROLES disable constraint FK_USERS_ROLES_USERS;
prompt Deleting SYS_USERS_ROLES...
delete from SYS_USERS_ROLES;
commit;
prompt Deleting SYS_USERS...
delete from SYS_USERS;
commit;
prompt Deleting SYS_ROLES_AUTHORITIES...
delete from SYS_ROLES_AUTHORITIES;
commit;
prompt Deleting SYS_ROLES...
delete from SYS_ROLES;
commit;
prompt Deleting SYS_AUTHORITIES_RESOURCES...
delete from SYS_AUTHORITIES_RESOURCES;
commit;
prompt Deleting SYS_RESOURCES...
delete from SYS_RESOURCES;
commit;
prompt Deleting SYS_AUTHORITIES...
delete from SYS_AUTHORITIES;
commit;
prompt Loading SYS_AUTHORITIES...
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('1303910437484', 'AUTH_xxx', 'xxx', null, null, '01');
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('AUTH_LOGIN4', 'AUTH_LOGIN', '登录', 1, 0, '01');
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('AUTH_AFTERLOGINWELCOME5', 'AUTH_AFTERLOGINWELCOME', '登录后欢迎界面', 1, 0, '01');
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('AUTH_XTSZ_DEPT1', 'AUTH_XTSZ_DEPT', '单位设置', 1, 0, '01');
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('AUTH_XTSZ_USER2', 'AUTH_XTSZ_USER', '用户设置、横向查询', 1, 0, '01');
insert into SYS_AUTHORITIES (AUTHORITY_ID, AUTHORITY_NAME, AUTHORITY_DESC, ENABLED, ISSYS, MODULE)
values ('AUTH_NODE_MGR3', 'AUTH_NODE_MGR', '节点管理、纵向查询', 1, 0, '01');
commit;
prompt 6 records loaded
prompt Loading SYS_RESOURCES...
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('1303909883031', 'ff', 'ff', 'action', 'b.jsp', null, 1, 0, null);
```

```
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('1303909847687', 'ff1', 'ff1', 'action', 'b.jsp', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('node_mgr3', 'node_mgr', '节点管理', 'url', '/*/*/Tree.jsp', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('login4', 'login', '登录', 'url', '/login.jsp', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('index5', 'index', '登录后欢迎页面', 'url', '/index.jsp', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('resources_mgr', 'resources_mgr', '资源管理', 'action', '/managerResource', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('horizontal_qry6', 'horizontal_qry', '横向查询', 'action', '/horizontalQuery', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('vertical_qry7', 'vertical_qry', '纵向查询', 'action', '/verticalQuery', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('dep_mgr1', 'dep_mgr', '单位管理', 'action', '/UnitsManager', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('user_mgr2', 'user_mgr', '用户管理', 'action', '/managerUser', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('authority_mgr', 'authority_mgr', '权限管理', 'action', '/managerAuthority', null, 1, 0, null);
insert into SYS_RESOURCES (RESOURCE_ID, RESOURCE_NAME, RESOURCE_DESC, RESOURCE_TYPE, RESOURCE_STRING, PRIORITY, ENABLE
D, ISSYS, MODULE)
values ('role_mgr', 'role_mgr', '角色管理', 'action', '/managerRole', null, null, null, null);
commit;
prompt 12 records loaded

prompt Loading SYS_AUTHORITIES_RESOURCES...
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (1, 'AUTH_AFTERLOGINWELCOME5', 'index5', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (2, 'AUTH_LOGIN4', 'login4', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (3, 'AUTH_NODE_MGR3', 'node_mgr3', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (4, 'AUTH_XTSZ_DEPT1', 'dep_mgr1', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
```

```
values (5, 'AUTH_XTSZ_USER2', 'user_mgr2', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (7, 'AUTH_XTSZ_USER2', 'horizontal_qry6', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (8, 'AUTH_XTSZ_DEPT1', 'vertical_qry7', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (12, 'AUTH_XTSZ_USER2', 'role_mgr', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (10, 'AUTH_XTSZ_USER2', 'resources_mgr', 1);
insert into SYS_AUTHORITIES_RESOURCES (ID, AUTHORITY_ID, RESOURCE_ID, ENABLED)
values (11, 'AUTH_XTSZ_USER2', 'authority_mgr', 1);
commit;
prompt 10 records loaded
prompt Loading SYS_ROLES...
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('1303463518765', 'ROLE_dd1', 'dd1', 1, 0, '01');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('1303463949640', 'ROLE_rr1', 'rr1', 1, 0, '02');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_PLATFORMADMIN1', 'ROLE_PLATFORMADMIN', '可管理整个平台的用户、单位设置。', 1, 1, '01');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_USER2', 'ROLE_USER', '普通用户', 1, 0, '01');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_LOGINTOWELCOME4', 'ROLE_LOGINTOWELCOME', '仅登录到欢迎界面！', 1, 0, '01');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_SYSADMIN3', 'ROLE_SYSADMIN', '可管理本系统的用户、单位设置。', 1, 0, '01');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_WORK', 'ROLE_WORK', '作业子系统的角色（试验）', 1, 0, '02');
insert into SYS_ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, ENABLED, ISSYS, MODULE)
values ('ROLE_LOGIN', 'ROLE_LOGIN', '系统登录', 1, 0, '01');
commit;
prompt 8 records loaded
prompt Loading SYS_ROLES_AUTHORITIES...
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1, 'ROLE_LOGINTOWELCOME4', 'AUTH_AFTERLOGINWELCOME5', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (2, 'ROLE_PLATFORMADMIN1', 'AUTH_AFTERLOGINWELCOME5', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (3, 'ROLE_PLATFORMADMIN1', 'AUTH_LOGIN4', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (4, 'ROLE_PLATFORMADMIN1', 'AUTH_NODE_MGR3', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (5, 'ROLE_PLATFORMADMIN1', 'AUTH_XTSZ_DEPT1', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (6, 'ROLE_PLATFORMADMIN1', 'AUTH_XTSZ_USER2', 1);
```

```
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (7, 'ROLE_SYSADMIN3', 'AUTH_XTSZ_DEPT1', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (8, 'ROLE_SYSADMIN3', 'AUTH_XTSZ_USER2', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (9, 'ROLE_USER2', 'AUTH_LOGIN4', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (10, 'ROLE_LOGINTOWELCOME4', 'AUTH_LOGIN4', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (11, 'ROLE_USER2', 'AUTH_AFTERLOGINWELCOME5', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463962718, '1303463949640', 'AUTH_LOGIN4', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463972234, 'ROLE_WORK', 'AUTH_LOGIN4', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463972235, 'ROLE_WORK', 'AUTH_AFTERLOGINWELCOME5', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463972250, 'ROLE_WORK', 'AUTH_XTSZ_DEPT1', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463972251, 'ROLE_WORK', 'AUTH_XTSZ_USER2', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303463972265, 'ROLE_WORK', 'AUTH_NODE_MGR3', 1);
insert into SYS_ROLES_AUTHORITIES (ID, ROLE_ID, AUTHORITY_ID, ENABLED)
values (1303287600015, 'ROLE_LOGIN', 'AUTH_LOGIN4', 1);
commit;
prompt 18 records loaded
prompt Loading SYS_USERS...
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('1304494573750', 'lxb', 'lxb', 'c7d3f4c857bc8c145d6e5d40c1bf23d9', null, 1, 0, '10011001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('1304490737406', 'lxb', 'lxb', 'c7d3f4c857bc8c145d6e5d40c1bf23d9', null, 1, 0, '10011001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('1304574079546', 'ddd', 'ddd', '0a4f6a961276619f7f91356bcba5a746', null, 0, 0, null, null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('1304573363921', 'lxb', '卢小兵', '09eb37d219cfa835db40e5ab587f7082', '普通仅登录到欢迎界面！', 0, 0, '1001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('1304573484515', 'lll', 'lll', '47acedc22cef8c3762c21a435e262d67', null, 1, 0, '1001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY,
SUB_SYSTEM)
values ('admin1', 'admin', '系统管理员', 'ceb4f32325eda6142bd65215f4c0f371', '超级系统管理员', 1, 1, '1001', null, '01');
```

```
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY, SUB_SYSTEM)
values ('user2', 'user', '普通用户', '47a733d60998c719cf3526ae7d106d13', '普通用户', 1, 0, '1001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY, SUB_SYSTEM)
values ('sysUser3', 'sysUser', '系统设置维护', '8f0295328c34f8eedc2362e9f4a10b7e', '系统设置用户', 1, 0, '1001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY, SUB_SYSTEM)
values ('lxb4', 'lxb', '卢小兵', 'c7d3f4c857bc8c145d6e5d40c1bf23d9', '普通仅登录到欢迎界面！', 1, 0, '1001', null, '01');
insert into SYS_USERS (USER_ID, USER_ACCOUNT, USER_NAME, USER_PASSWORD, USER_DESC, ENABLED, ISSYS, USER_DEPT, USER_DUTY, SUB_SYSTEM)
values ('1304566319625', 'lxb5', 'lx5', '1abe40ed6d0da1c834586e8ecef61fe7', null, 0, 0, '10011001', null, '01');
commit;
prompt 10 records loaded
prompt Loading SYS_USERS_ROLES...
insert into SYS_USERS_ROLES (ID, USER_ID, ROLE_ID, ENABLED)
values (1, 'admin1', 'ROLE_PLATFORMADMIN1', 1);
insert into SYS_USERS_ROLES (ID, USER_ID, ROLE_ID, ENABLED)
values (2, 'sysUser3', 'ROLE_SYSADMIN3', 1);
insert into SYS_USERS_ROLES (ID, USER_ID, ROLE_ID, ENABLED)
values (3, 'user2', 'ROLE_USER2', 1);
insert into SYS_USERS_ROLES (ID, USER_ID, ROLE_ID, ENABLED)
values (4, 'lxb4', 'ROLE_LOGINTOWELCOME4', 1);
insert into SYS_USERS_ROLES (ID, USER_ID, ROLE_ID, ENABLED)
values (5, '1304573484515', '1303463518765', null);
commit;
prompt 5 records loaded
prompt Enabling foreign key constraints for SYS_AUTHORITIES_RESOURCES...
alter table SYS_AUTHORITIES_RESOURCES enable constraint FK_PUB_AUTHORITIES_RE_AU;
alter table SYS_AUTHORITIES_RESOURCES enable constraint FK_PUB_AUTHORITIES_RE_RE;
prompt Enabling foreign key constraints for SYS_ROLES_AUTHORITIES...
alter table SYS_ROLES_AUTHORITIES enable constraint FK_PUB_ROLES_AUTHORITIES_AU;
alter table SYS_ROLES_AUTHORITIES enable constraint FK_PUB_ROLES_AUTHORITIES_ROLES;
prompt Enabling foreign key constraints for SYS_USERS_ROLES...
alter table SYS_USERS_ROLES enable constraint FK_USERS_ROLES_ROLES;
alter table SYS_USERS_ROLES enable constraint FK_USERS_ROLES_USERS;
prompt Enabling triggers for SYS_AUTHORITIES...
alter table SYS_AUTHORITIES enable all triggers;
prompt Enabling triggers for SYS_RESOURCES...
alter table SYS_RESOURCES enable all triggers;
prompt Enabling triggers for SYS_AUTHORITIES_RESOURCES...
alter table SYS_AUTHORITIES_RESOURCES enable all triggers;
prompt Enabling triggers for SYS_ROLES...
```

```
alter table SYS_ROLES enable all triggers;
prompt Enabling triggers for SYS_ROLES_AUTHORITIES...
alter table SYS_ROLES_AUTHORITIES enable all triggers;
prompt Enabling triggers for SYS_USERS...
alter table SYS_USERS enable all triggers;
prompt Enabling triggers for SYS_USERS_ROLES...
alter table SYS_USERS_ROLES enable all triggers;
set feedback on
set define on
prompt Done.
```

相关配置文件:

web.xml与第一种方法同。

applicationContext-security.xml:

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

<b:beans xmlns="http://www.springframework.org/schema/security"
 xmlns:b="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">


 <http auto-config="true" access-denied-page="/accessDenied.jsp">
  <!-- 不要过滤图片等静态资源 -->
  <intercept-url pattern="/**/*.jpg" filters="none" />
  <intercept-url pattern="/**/*.png" filters="none" />
  <intercept-url pattern="/**/*.gif" filters="none" />
  <intercept-url pattern="/**/*.css" filters="none" />
  <intercept-url pattern="/**/*.js" filters="none" />
  <!-- 登录页面和忘记密码页面不过滤  -->
  <intercept-url pattern="/login.jsp" filters="none" />
  <intercept-url pattern="/jsp/forgotpassword.jsp"
   filters="none" />

  <form-login login-page="/login.jsp"
   authentication-failure-url="/login.jsp?error=true"
   default-target-url="/index.jsp" />

  <!-- "记住我"功能，采用持久化策略（将用户的登录信息存放在数据库表中） -->
```

```
    <remember-me data-source-ref="dataSource" />

    <!-- 检测失效的sessionId,超时时定位到另外一个URL -->
    <session-management invalid-session-url="/sessionTimeout.jsp" />


    <!-- 增加一个自定义的filter，放在FILTER_SECURITY_INTERCEPTOR之前，
    实现用户、角色、权限、资源的数据库管理。 -->
    <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。 -->
<b:bean id="myFilter"
 class="avatar.base.security.MyFilterSecurityInterceptor">
 <b:property name="authenticationManager"
  ref="authenticationManager"/>
 <b:property name="accessDecisionManager"
  ref="myAccessDecisionManager"/>
 <b:property name="securityMetadataSource"
  ref="mySecurityMetadataSource"/>
</b:bean>



<!-- 注意能够为authentication-manager 设置alias别名 -->
<authentication-manager alias="authenticationManager">
 <authentication-provider user-service-ref="userDetailsManager">
  <password-encoder ref="passwordEncoder">
   <salt-source user-property="username" />
  </password-encoder>
 </authentication-provider>
</authentication-manager>


<!-- 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。 -->
<b:bean id="myAccessDecisionManager"
 class="avatar.base.security.MyAccessDecisionManager">
</b: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去访问。 -->
<b:bean id="mySecurityMetadataSource"
 class="avatar.base.security.MyInvocationSecurityMetadataSourceService">
```

```
 </b:bean>

</b:beans>
```

applicationContext-service.xml:

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

<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:util="http://www.springframework.org/schema/util"
 xmlns:jee="http://www.springframework.org/schema/jee"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:context="http://www.springframework.org/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
   http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
   http://www.springframework.org/schema/jee
   http://www.springframework.org/schema/jee/spring-jee-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd
   http://www.springframework.org/schema/util
   http://www.springframework.org/schema/util/spring-util-3.0.xsd">


 <!-- 定义上下文返回的消息的国际化。 -->
 <bean id="messageSource"
  class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
  <property name="basename"
   value="classpath:org/springframework/security/messages_zh_CN"/>
 </bean>

 <!--
  事件监听:实现了 ApplicationListener监听接口,
  包括AuthenticationCredentialsNotFoundEvent 事件,
  AuthorizationFailureEvent事件，AuthorizedEvent事件， PublicInvocationEvent事
  件。 -->
 <bean
  class="org.springframework.security.authentication.event.LoggerListener" />

 <!-- 用户的密码加密或解密 -->
```

```
 <bean id="passwordEncoder"
  class="org.springframework.security.authentication.encoding.Md5PasswordEncoder" />


 <!-- 用户详细信息管理：数据源、用户缓存（通过数据库管理用户、角色、权限、资源）。 -->
 <bean id="userDetailsManager" class="avatar.base.security.MyUserDetailsService">
  <property name="pubUsersHome" ref="pubUsersHome" />
  <property name="pubAuthoritiesResourcesHome" ref="pubAuthoritiesResourcesHome" />
  <property name="dataSource" ref="dataSource" />
  <property name="userCache" ref="userCache" />
 </bean>

 <!-- 启用用户的缓存功能 -->
 <bean id="userCache"
  class="org.springframework.security.core.userdetails.cache.EhCacheBasedUserCache">
  <property name="cache" ref="userEhCache" />
 </bean>

 <bean id="userEhCache" class="org.springframework.cache.ehcache.EhCacheFactoryBean">
  <property name="cacheName" value="userCache" />
  <property name="cacheManager" ref="cacheManager" />
 </bean>

 <bean id="cacheManager"
  class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean" />


 <!-- spring security自带的与权限有关的数据读写Jdbc模板 -->
 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource" />
 </bean>

</beans>
```

第三种方法扩展后Spring Security3.0.2的验证和授权方法

为了叙述的严谨性，这里说的是Spring Security3.0.2，而非其他版本，这是因为我只读过Spring Security3.0.2的代码，并且在该版本上面扩展自定义的动态管理用户、角色、权限和资源成功。 估计其他版本的验证和授权方法是差不太多的，因为没有接触过，也不敢大胆猜测。

在扩展后的Spring Security3.0.2中，验证及授权的过程如下：
1、当Web服务器启动时，通过Web.xml中对于Spring Security的配置，加载过滤器链，那么在加载MyFilterSecurityInterceptor类时，会注入MyInvocationSecurityMetadataSourceService、MyUserDetailsService、MyAccessDecisionManager类。

2、该MyInvocationSecurityMetadataSourceService类在执行时会提取数据库中所有的用户权限，形成权限列表；
并循环该权限列表，通过每个权限再从数据库中提取出该权限所对应的资源列表，并将资源（URL）作为key，权限列表作为value，形成Map结构的数据。

3、当用户登录时，AuthenticationManager进行响应，通过用户输入的用户名和密码，然后再根据用户定义的密码算法和盐值等进行计算并和数据库比对，

当正确时通过验证。此时MyUserDetailsService进行响应，根据用户名从数据库中提取该用户的权限列表，组合成UserDetails供Spring Security使用。

4、当用户点击某个功能时，触发MyAccessDecisionManager类，该类通过decide方法对用户的资源访问进行拦截。
用户点击某个功能时，实际上是请求某个URL或Action， 无论.jsp也好，.action或.do也好，在请求时无一例外的表现为URL。
还记得第2步时那个Map结构的数据吗？ 若用户点击了"login.action"这个URL之后，那么这个URL就跟那个Map结构的数据中的key对比，若两者相同，则根据该url提取出Map结构的数据中的value来，这说明：若要请求这个URL，必须具有跟这个URL相对应的权限值。这个权限有可能是一个单独的权限，也有可能是一个权限列表，也就是说，一个URL有可能被多种权限访问。

那好，我们在MyAccessDecisionManager类的decide这个方法里，将通过URL取得的权限列表进行循环，然后跟第3步中登录的用户所具有的权限进行比对，若相同，则表明该用户具有访问该资源的权利。 不大明白吧？ 简单地说， 在数据库中我们定义了访问“LOGIN”这个URL必须是具有ROLE_ADMIN权限的人来访问，那么，登录用户恰恰具有该ROLE_ADMIN权限，两者的比对过程中，就能够返回TRUE，可以允许该用户进行访问。就这么简单！

不过在第2步的时候，一定要注意，MyInvocationSecurityMetadataSoruceService类的loadResourceDefine()方法中，形成以URL为key，权限列表为value的Map时，
要注意key和Value的对应性，避免Value的不正确对应形成重复，这样会导致没有权限的人也能访问到不该访问到的资源。
还有getAttributes()方法，要有 url.indexOf("?")这样的判断，要通过判断对URL特别是Action问号之前的部分进行匹配，防止用户请求的带参数的URL与你数据库中定义的URL不匹配，造成访问拒绝！

第三种方法BTW

当然，你在设计了7张表之后，那么对于这些之间相互关联的关系内容及信息内容，就得由你来进行维护了，大约有用户、角色、权限、资源的增删改查，并还需要设置用户和角色、角色和权限、权限和资源之间的关系。可考虑分为三个菜单进行维护，用户设置、角色设置、资源设置。 在用户设置里分别管理用户、用户与角色的关系；在角色设置里管理角色、角色与权限的关系； 在资源设置里分别管理权限、权限与资源的关系等。

第四种方法

第四种方法就是直接修改源码以达到第三种方法的效果。

本来准备是直接从源码修改来的， 但是始终认为修改源码并非终极解决之道，有违OO的精神本质，再者由于时间关系，只是对代码进行了研究，但并没有进行实现或验证。只待以后时间稍稍宽松时再做为兴趣进行研究，在次不过多的讲解。但据我从代码上来看，一是将从配置文件中获取用户及权限的功能修改为从数据库中提取出来；二是将从配置文件中获取权限和资源的对应关系修改为从数据库中提取；三是修改User增加相关信息等。

始终还是围绕着JdbcDaoImpl和DefaultFilterInvocationSecurityMetadataSource还有User这3个类进行修改。
以实现从数据库提取用户、角色、权限和资源信息。

有兴趣的就先试试吧，等试好了告诉我一声哈。

Spring Security的优缺点

不可否认，Spring Security依赖于Spring的Ioc、AOP等机制，横切开系统的业务组件，将通用的权限功能注入到业务组件内部，实现了通用功能和业务功能的无缝整合，但又保证了通用功能和业务功能的实现上的分离，省却了一部分工作量，这是其存在的最重要意义。

但又不可否认，Spring Security所具有的缺乏动态资源管理的硬伤（若是能够提供用户、角色、权限和资源的数据库管理，并且提供管理界面那实在是太完美了，可惜这两样一样都不能实现），又令国人用户爱恨交加。

该何去何从，就请自己做个选择吧！

完整例子源码下载

/Files/SpartaYew/framework.rar
例子中使用到的lib包太大了，传了几次网络中断无法上传，请朋友们自行下载或通过加我的QQ或邮箱进行下载吧。：）

－东营 sparta-紫杉 原创，转载请注明出处：）
http://www.blogjava.net/SpartaYew/
SpartaYew@163.com
QQ:22086526

posted on 2011-05-19 18:03 sparta-紫杉 阅读(15321) 评论(92) 编辑 收藏 所属分类: SSH2

评论

re: 春天的故事－Spring Security3十五日研究[未登录] 回复 更多评论
动态资源管理这块不是什么问题，改造下就行了... 楼主还没尝试过使用它的权限判断标签吧 ... 也有问题

2011-05-19 18:24 | junxy

re: 春天的故事－Spring Security3十五日研究 回复 更多评论
@junxy
嗯，应该是有问题的，很快就会浮出水面。目前尚未深入研究。

2011-05-20 07:48 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论
LZ 加我Q 请教：676662066

2011-05-24 10:11 | 学员

re: 春天的故事－Spring Security3十五日研究 回复 更多评论
@学员
嗯，今天上午去见用户了，现在刚回来，已经通过你的QQ申请，在QQ谈吧，哈哈。

2011-05-24 12:05 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究[未登录] 回复 更多评论
动态资源管理这块我想可以这样,你现在的做法是:
2、该MyInvocationSecurityMetadataSourceService类在执行时会提取数据库中所有的用户权限，形成权限列表;
并循环该权限列表，通过每个权限再从数据库中提取出该权限所对应的资源列表，并将资源（URL）作为key，权限列表作为value，形成Map结构的数据。
4、当用户点击某个功能时，触发MyAccessDecisionManager类，该类通过decide方法对用户的资源访问进行拦截。
用户点击某个功能时，实际上是请求某个URL或Action， 无论.jsp也好，.action或.do也好，在请求时无一例外的表现为URL。

还记得第2步时那个Map结构的数据吗？ 若用户点击了"login.action"这个URL之后，那么这个URL就跟那个Map结构的数据中的key对比，若两者相同，则根据该url提取出Map结构的数据中的value来，这说明：若要请求这个URL，必须具有跟这个URL相对应的权限值。这个权限有可能是一个单独的权限，也有可能是一个权限列表，也就是说，一个URL有可能被多种权限访问。

只要修改这两部的实现,实时的从数据库中读取就行了,如果觉得性能比较差,就加上缓存,我也没试过这个行不行

LZ 加我Q 请教：303208751

2011-05-25 17:27 | chen

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@chen

没错。说得好。

2011-05-25 17:44 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

还是有问题，能不能给个例子源码下载，谢谢，邮箱：liningwangyi@163.com

2011-05-29 13:49 | 求问

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

我最近急着用spring security，急求帮助。能不能把你做的小例子发给我
邮箱：liningwangyi@163.com

2011-05-29 13:55 | 求问

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@求问

嗯，您好，现已经将完整Struts2＋spring3+hibernate3+security3的例子发送到您的邮箱，请查收!

2011-05-29 16:04 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

：） 嗯，盆友们好，现将例子源码放到文后提供下载，有需要的盆友，去下载吧。
另外，源码是使用了jdk1.6 + struts2.1.8 + spring3.01 + hibernate3.3.1 + security3.0.2 + weblogic10.3 + oracle9i + C3P0-0.9.1的，所使用的lib比较多，无法上传到博客园，因此有需要lib包的，请加我QQ或通过我邮箱来拿吧。 ：）

2011-05-29 22:17 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

：），最新更新了第三种方法的“SQL语句及预置数据”，之前上传的SQL是老版本的了，和框架中的hibernate的配置文件不符，现在相符了，给造成不便的盆友们表示歉意。并且感谢“云行天下”盆友指出这个错误。

2011-06-01 11:00 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

赞

2011-06-14 23:22 | eric

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@eric

哈哈，谢谢。

2011-06-15 07:36 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

看你的文章，一下就明白了，太好了，希望有代码供学习
邮箱：zhangjf@bc-info.net

2011-06-16 00:15 | 阿龙

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@阿龙

您好，代码中使用的lib包大大了，用邮箱发不过去，你加我QQ吧，通过QQ发送，22086526。

2011-06-16 07:51 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

请问下，当使用了iframe session过期怎么处理，意思是session过期不会自动跳到登陆页面，而是当你点击页面的时候iframe由于session过期所以呈现的是登陆页面。

2011-07-19 14:44 | joe

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@joe

试试在sessionTimeout.jsp页面底部增加如下js：

```
<script type="text/javascript">

if (self != top){
window.top.location = window.location;
}

</script>
```

应该能够解决你的问题。

2011-07-19 20:37 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

受益匪浅啊。您好，可以吧lib下的包发给我吗？谢谢。
邮箱：jkl456asd@126.com

2011-07-29 13:36 | 易风

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@易风

当然可以，不过你还是加我的QQ吧，邮箱发不过去，太大了。

2011-07-29 14:24 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

写得灰常详细，学习了。

2011-08-16 11:01 | goldgood

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

可以把完整的例子发我一份吗？谢谢
邮箱:luckytqm@163.com

2011-08-17 18:50 | 学路人

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@学路人

当然可以，不过你还是加我的QQ吧，邮箱发不过去，太大了。

2011-08-17 21:55 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

我正准备在项目中应用ss3，使用的是spring mvc按照你的文章，配置了spring security相关的配置，但是在使用@Autowired的时候遇到问题，调用resourceService类始终都是null，看您的代码中有用到注解，这个问题困扰了我好几天，采用了很多种方法都无法解决。包括将配置在开始时加载（我原来是通过一个servlet加载的所有sping bean配置）。
不知道您使用过@Autowired方法时是否遇到此问题。如果遇到，恳请为小弟答疑解惑。
联系方式 QQ:305226596

2011-09-05 19:21 | TechNick

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

很多地方想请教LZ，可以加Q吗

2011-09-08 10:22 | 谑孩无牙

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@谑孩无牙

加吧，朋友。

2011-09-08 10:44 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@TechNick
这种问题，之前碰到过类似的问题，最大的可能就是在Spring MVC中没有配置好的原因，跟Security基本没有关系。

2011-09-08 10:45 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

你有做spring security 和 spring webflow 的整合吗~ 网上资料特别少

2011-09-29 18:04 | 小哈

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@小哈

这个没有。

2011-09-30 08:53 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@sparta-紫杉
我现在项目里有这个需求，而我根据webflow reference 的整合那一块做了配置，就是报错，这个问题纠结我两天了，我做了一个webflow 的Demo，一个security的Demo，整合到一起，就是不行...

2011-09-30 11:10 | 小哈

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

是了， 如果需要 有后台和前台 多页面登录 该怎么配呢

网上的只有一篇文章，好乱的，请高手有空总结这方面的知识，像这篇那样 通俗易懂!

2011-10-24 06:04 | 饼干

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@饼干

嗯，有需要时研究看看。

2011-10-24 10:15 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

您好，能不能发一份完整的源码给我，我刚接触spring security，最好能附上数据库，谢谢!
195738261@qq.com

2011-10-31 13:49 | Demon

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

你好.最近在学习这方面的东西. 能不能给一个整合的小例子.我喜欢你的通俗易懂!我QQ:475589699 加好友要回答问题,请说明一下是你.谢谢.

2011-11-01 16:22 | 土豆

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@土豆
你加我吧，亲爱的，回答问题怪麻烦的!

2011-11-02 11:18 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

赞~！写得很好很详细。其实，楼主，如果lib包太大的话可以截图上传上来哦，这样也方便找么，呵呵。找jar包是一件挺蛋疼的事来的，(*^__^*) 嘻嘻……

2011-11-05 16:46 | 杰杰

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

有哪位侠客整理好这份源码的么？麻烦发我一份啦，346527107@qq.com，如果实在难上传的话将jar包截图发我一份也行，谢谢啦。

2011-11-06 10:44 | java爱好者

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

楼主，请问如何通过你的邮箱拿？没有密码登陆不上去的吧？

2011-11-06 10:54 | java求知

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@杰杰
@Java爱好者
@java求知

加我QQ，我发给你们吧，注明一下“索Security3的jar包”。

2011-11-07 08:58 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

学习中，能发一份给我吗，谢谢。kevin588168@163.com

2011-11-10 16:54 | 菜包子

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

作者写的很详细，
能不能发一份完整的源码给我、
E-mail:1094913009@qq.com
THX~

2011-11-10 17:25 | saya

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

我看到很多人在问JAR，其实楼主已经告诉你们了，在项目的 .CLASSPATH 里就有~

2011-11-11 13:19 | 拾里桃花

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

嘿嘿~忘记，感谢楼主了。以后继续来学习~

2011-11-11 13:20 | 拾里桃花

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@saya

你好，完整开发框架已发，请查收你的邮箱。

2011-11-12 19:40 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

楼主你好，代码执行时报错：Caused by: com.mysql.jdbc.exceptions.MySQLSyntaxErrorException: Table 'security.persistent_logins' doesn't exist，没有persistent_logins这个表吧。

2011-11-13 13:16 | java求知

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

楼主你好，运行代码时出现问题：org.apache.jasper.JasperException: /jsp/sys/sysConfig.jsp(16,4) Attribute theme invalid for tag head according to TLD 麻烦楼主帮我看看是什么问题，谢谢。

2011-11-13 13:29 | spring浅尝

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

感谢楼主，能不能发一份完整的源码给我
E-mail:guyuewenwu2008@sohu.com
THX~

2011-11-15 14:15 | guyue

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

在角色与权限对应，权限与URL对应的情况下，如果我在MyAccessDecisionManager获取当前用户的角色列表，然后match当前访问的URL，每次访问URL之前都动态对比，这种实现不知效率如何。

2011-11-21 10:31 | yoya

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@yoya

我倒是觉得效率差不多。

2011-11-21 16:23 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

按照楼主的配置，程序跑到AbstractUserDetailsAuthenticationProvider类中时，userCache总是NullUserCache对象呢。已经在配置了用户缓存了...

2011-12-02 00:15 | guyue

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@guyue

盆友，不妨给我个QQ邮箱地址，发你一份配置参考下吧!

2011-12-03 08:45 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

1716030936@qq.com，感谢楼主

2011-12-04 11:33 | guyue

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

感谢楼主，能不能发一份完整的源码给我
E-mail:bflcf@126.com

2011-12-10 10:18 | 林峰

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@林峰

朋友，已发，请查收。

2011-12-12 09:21 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

可以吧代码 发一下吗？谢谢fujitac001@126.com

2011-12-13 17:53 | 求知识

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

你好 看了你的描述 对SS3了解不少 初次使用SS3 对你的第三种方式的几个DAO不能很好的理解 能发下源码吗？谢谢 邮箱verysun@126.com

2011-12-14 17:23 | SS3

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
@求知识

朋友，已发，请查收。

2011-12-17 15:35 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
@SS3

朋友，已发，请查收。

2011-12-17 15:35 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
楼主你好，我正在学习这个，但是看了很多文档都没看懂，能否把你的Struts2＋spring3+hibernate3+security3例子发一份给我，123246928@qq.com

2011-12-22 10:43 | worina

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
关于动态加载权限数据，可以在页面上做个按钮，点击一下就可以重新加载权限和资源

2011-12-23 15:01 | cs

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
你好，可不可以往我邮箱发一份完整的资料（vacume@sina.com）谢谢～

2011-12-31 14:06 | Vacume

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
您好，可不可给我发一份完整的资料(pp.myy@163.com)谢谢。

2012-01-31 16:03 | 彭鹏

re: 春天的故事—Spring Security3十五日研究 回复 更多评论
@worina

朋友，已发，请查收。

2012-02-01 08:28 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@Vacume

朋友，已发，请查收。

2012-02-01 08:28 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@彭鹏
朋友，已发，请查收。

2012-02-01 08:29 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

您好，可不可给我发一份完整的资料332690841@qq.com谢谢。

2012-02-02 10:10 | st

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@st

朋友，已发，请查收。

2012-02-02 14:43 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

非常感谢!

2012-02-02 16:58 | 彭鹏

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

我照着你这个做的，怎么启动时报错： authenticationManager is requeired

2012-02-02 20:03 | 铁血无双

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

您好，最近项目急用spring security，能给我发份详细的材料吗？灰常感谢！
邮箱：elbert_li@163.com

2012-02-09 10:06 | elbert_li

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@elbert_li

朋友，已经发给你，请查收。

2012-02-10 09:00 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

最近正在选择安全框架，能否给我一份源码！谢谢！ comwangxu@gmail.com

2012-02-10 11:59 | look

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@look

朋友，已经发给你，请查收。

2012-02-10 19:55 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

您好，我最近在研究Spring Security3，能发给我一份吗，谢谢！
E-Mail:sunjinyong@21cn.com

2012-02-21 09:02 | loverous

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

点击其他URL， 没有执行自定义的accessDecisionManager。 问题出在哪？

2012-02-21 13:02 | tyler

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@tyler

资源配错了？ 数据库url配错了？

2012-02-21 16:57 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@loverous

给你发过去了，朋友。

2012-02-21 16:58 | sparta-紫杉

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

cvc-complex-type.2.4.c: The matching wildcard is strict, but no declaration can be found for element 'http'.

请教下，这个错误是什么原因？
我的xsi:schemaLocation已经写完整，版本是spring-beans-2.5.xsd和spring-security-2.0.4.xsd

2012-02-22 10:05 | elbert_li

点击其他URL， 没有执行自定义的accessDecisionManager。 问题出在哪？ 回复 更多评论

resource表里的 URL配置有问题。action能用模式匹配吗？
发现代码有几个错误， 不输入帐号（或错误的帐号，密码无所谓）提交，报错。像这种db里不存在的User登录，是给一个默认guest还是程序处理NULL好？
TAG显示有问题， 有时报persionobject 不存在错误。
用hibernate的注解 实现时， 自动扫描 Userservice 有错误。

2012-02-22 20:17 | Tyler

re: 春天的故事－Spring Security3十五日研究[未登录] 回复 更多评论

下代码下来研究下，多谢楼主的share精神。不可否认，楼主绝对是个好淫啊：）

2012-02-27 13:15 | David

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

楼主，能不能把你做的demo发给我下，非常感谢，
我的邮箱：631362465@qq.com

2012-03-02 08:59 | leeson

re: 春天的故事－Spring Security3十五日研究 回复 更多评论

@leeson

发给你了，请查收。

2012-03-05 11:24 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

楼主，能也发一份demo给我不，谢谢：

2785153232@qq.com

2012-03-06 14:47 | 风过留痕

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

@风过留痕

朋友，发给你了，请查收。

2012-03-06 20:53 | sparta-紫杉

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

楼主，最近项目中要用到ss3,能不能麻烦你也发份demo给我，liuweify@sina.com. 谢谢了！

2012-03-07 22:49 | @outofmemory

在tomcat下经常发生session过期 回复 更多评论

错误信息如下： 是设置问题吗？web.xml里已经设了session30分钟才过期。

HTTP Status 500 -

---

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

java.lang.IllegalStateException: setAttribute: Session [690C0360AD0D665F61955CAEB5F80E6A] has already been invalidated
org.apache.catalina.session.StandardSession.setAttribute(StandardSession.java:1425)
org.apache.catalina.session.StandardSession.setAttribute(StandardSession.java:1390)
org.apache.catalina.session.StandardSessionFacade.setAttribute(StandardSessionFacade.java:154)
org.springframework.security.web.authentication.session.SessionFixationProtectionStrategy.onAuthentication(SessionFixationProtectionStrategy.java:102)
org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter.doFilter(AbstractAuthenticationProcessingFilter.java:205)

org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:355)
org.springframework.security.web.authentication.logout.LogoutFilter.doFilter(LogoutFilter.java:105)
org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:355)
org.springframework.security.web.context.SecurityContextPersistenceFilter.doFilter(SecurityContextPersistenceFilter.java:79)
org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:355)
org.springframework.security.web.FilterChainProxy.doFilter(FilterChainProxy.java:149)
org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:237)
org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:167)
org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:88)
org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:76)

note The full stack trace of the root cause is available in the Apache Tomcat/7.0.5 logs.

2012-03-09 16:12 | tyler

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

帮楼主补充一点，spring密码加密那块，判断密码spring是区分大小写的，用小写，不要用大写，否则验证过不去，这个问题很坑爹啊，可能是我太白痴了，调试了好久。

2012-03-13 11:58 | 赵亮

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

楼主给我发个jar包吧，老是出错hww_315@163.com,谢谢

2012-03-15 16:04 | Bruce

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

我想要一份不要数据库的例子。
最近学这个，现在是能帮我拦截配置的URL了 但是就是登陆的时候没有进入到
UserDdetalsService的loadUserByUsername方法。

2012-03-15 17:59 | 我

re: 春天的故事—Spring Security3十五日研究[未登录] 回复 更多评论

忘记留邮箱了
1140384524

2012-03-15 18:00 | 我

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

求Demo楼主。。。651125352@qq.com

2012-03-16 11:27 | jamf

re: 春天的故事—Spring Security3十五日研究 回复 更多评论

正在学习这块！辛苦了lz
2628329@qq.com

2012-03-25 12:34 | cno

新用户注册 刷新评论列表

博问 - 解决您的IT难题

博客园 博问 IT新闻 Java程序员招聘

标题 re: 春天的故事—Spring Security3十五日研究

姓名

主页

验证码 *

内容(请不要发表任何与政治相关的内容)

Remember Me?

提交 登录

[使用Ctrl+Enter键可以直接提交]

IT新闻:
· 10幅图让你了解Google与其它科技公司的不同之处
· Zynga或将与微软结盟 进军电视游戏领域
· RIM非洲开拓新市场 拟推出更低价格手机
· 六款优秀的Linux吉他工具
· 当当网俞渝呼吁政府应在网购立法上有所作为
博客园首页随笔:
· UMDF驱动程序快速上手
· C# 温故而知新：Stream篇（三）
· 微软官方windows phone开发视频教程第一天视频(附下载地址)
· 用原生JS进行CSS格式化和压缩
· Sql正则替换
知识库:
· Windows Runtime - 面向对象化的C++（并非意味着托管）
· 谈一谈 Windows 8 的软件开发架构
· 开发Metro版浏览器
· 开发WinRT自定义组件
· Win8探索学习笔记

网站导航:
博客园 IT新闻 知识库 C++博客 程序员招聘 管理
相关文章:
Birt + SSH2 完整实践
Struts2的dojo使用与/template/ajax/head.ftl not found.
Struts2中以非迭代方式提取Map中的值
Struts2标签之Checkbox详解
hibernate中提倡持久类实现equals()和hashCode()的原因分析
hibernate3关联映射表的级联关系维护
Hibernate3查询返回Map探秘
Spring3中XmlBeanDefinitionReader类的玩笑