1. **pring Security**  
   11个步骤为应用程序添加安全防护
2. **历史与现状**  
   自2003年出现的Spring扩展插件Acegi Security发展而来。  
   目前最新 版本为3.x，已成为Spring的一部分。  
   为J2EE企业应用程序提供可靠的安全性服务。
3. **Authentication vs. Authorization**  
   区分概念验证与授权  
   验证  
   这 个用户是谁？  
   用户身份可靠吗？  
   授权  
   某用户A是否可以访问资源R  
   某用户A是否可以执行M操作  
   某用户A是否可以对资 源R执行M操作
4. **SS中的验证特点**  
   支持多种验证方式  
   支持多种加密格式  
   支持组件的扩展和替换  
   可以本地化 输出信息
5. **SS中的授权特点**  
   支持多种仲裁方式  
   支持组件的扩展和替换  
   支持对页面访问、方法访问、对象访问 的授权。
6. **SS核心安全实现**  
   Web安全  
   通过配置Servlet Filter激活SS中的过滤器链  
   实现 Session一致性验证  
   实现免登陆验证（Remember-Me验证）  
   提供一系列标签库进行页面元素的安全控制  
   方法安全  
   通 过AOP模式实现安全代理  
   Web安全与方法安全均可以使用表达式语言定义访问规则
7. **配置SS**  
   配置Web.xml，应用安全过滤器  
   配置Spring，验证与授权部分  
   在web页面 中获取用户身份  
   在web页面中应用安全标签库  
   实现方法级安全
8. **1：配置web.xml**

Java代码

* 1. <filter>
  2. <filter-name>springSecurityFilterChain</filter-name>
  3. <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  4. </filter>
  5. <filter-mapping>
  6. <filter-name>springSecurityFilterChain</filter-name>
  7. <url-pattern>/\*</url-pattern>
  8. </filter-mapping>
  9. <context-param>
  10. <param-name>contextConfigLocation</param-name>
  11. <param-value>classpath:spring.xml</param-value>
  12. </context-param>
  13. <listener>
  14. <listener-class>
  15. org.springframework.web.context.ContextLoaderListener
  16. </listener-class>
  17. </listener>

<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> <context-param> <param-name>contextConfigLocation</param-name> <param-value>classpath:spring.xml</param-value> </context-param> <listener> <listener-class> org.springframework.web.context.ContextLoaderListener </listener-class> </listener>

1. **2：Spring配置文件中设置命名空间**

Java代码

* 1. <?xml version="1.0" encoding="UTF-8"?>
  2. <beans:beansxmlns="http://www.springframework.org/schema/security"
  3. xmlns:beans="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  4. xsi:schemaLocation="http://www.springframework.org/schema/beans
  5. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  6. http://www.springframework.org/schema/security
  7. http://www.springframework.org/schema/security/spring-security-3.0.xsd">
  8. </beans:beans>

<?xml version="1.0" encoding="UTF-8"?> <beans:beansxmlns="http://www.springframework.org/schema/security" xmlns:beans="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.xsd"> </beans:beans>

1. **3：配置最基本的验证与授权**

Java代码

* 1. <http auto-config="true">
  2. <intercept-url pattern="/\*\*" access="ROLE\_USER" />
  3. </http>
  4. <authentication-manager>
  5. <authentication-provider>
  6. <user-service>
  7. <user name="tom" password="123" authorities="ROLE\_USER, ROLE\_A" />
  8. <user name="jerry" password="123" authorities="ROLE\_USER, ROLE\_B" />
  9. </user-service>
  10. </authentication-provider>
  11. </authentication-manager>

<http auto-config="true"> <intercept-url pattern="/\*\*" access="ROLE\_USER" /> </http> <authentication-manager> <authentication-provider> <user-service> <user name="tom" password="123" authorities="ROLE\_USER, ROLE\_A" /> <user name="jerry" password="123" authorities="ROLE\_USER, ROLE\_B" /> </user-service> </authentication-provider> </authentication-manager>

1. **4：通过数据库验证用户身份**

Java代码

* 1. <authentication-manager>
  2. <authentication-provider>
  3. <password-encoder hash=“md5”/>
  4. <jdbc-user-service data-source-ref="dataSource"/>
  5. </authentication-provider>
  6. </authentication-manager>
  7. 数 据表结构见SS说明手册附录A

<authentication-manager> <authentication-provider> <password-encoder hash=“md5”/> <jdbc-user-service data-source-ref="dataSource"/> </authentication-provider> </authentication-manager> 数据表结构见SS说明手册附录A

1. **5：完善web页面验证规则**

Java代码

* 1. <http auto-config="true">
  2. <intercept-url pattern="/js/\*\*" filters="none"/>
  3. <intercept-url pattern="/css/\*\*" filters="none"/>
  4. <intercept-url pattern="/images/\*\*" filters="none"/>
  5. <intercept-url pattern="/a.jsp" access="ROLE\_A" />
  6. <intercept-url pattern="/b.jsp" access="ROLE\_B" />
  7. <intercept-url pattern="/c.jsp" access="ROLE\_A, ROLE\_B" />
  8. <intercept-url pattern="/\*\*" access="ROLE\_USER" />
  9. </http>

<http auto-config="true"> <intercept-url pattern="/js/\*\*" filters="none"/> <intercept-url pattern="/css/\*\*" filters="none"/> <intercept-url pattern="/images/\*\*" filters="none"/> <intercept-url pattern="/a.jsp" access="ROLE\_A" /> <intercept-url pattern="/b.jsp" access="ROLE\_B" /> <intercept-url pattern="/c.jsp" access="ROLE\_A, ROLE\_B" /> <intercept-url pattern="/\*\*" access="ROLE\_USER" /> </http>

1. **6：自定义验证配置**

Java代码

* 1. <!-- 指 定登陆页面、成功页面、失败页面-->
  2. <form-login login-page="/login.jsp" default-target-url="/index.jsp" authentication-failure-url="/login.jsp" />
  3. <!-- 尝 试访问没有权限的页面时跳转的页面 -->
  4. <access-denied-handler error-page="/accessDenied.jsp"/>
  5. <!-- 使 用记住用户名、密码功能，指定数据源和加密的key -->
  6. <remember-me data-source-ref="dataSource" />
  7. <!-- logout 页面，logout后清除session -->
  8. <logout invalidate-session="true" logout-success-url="/login.jsp" />
  9. <!-- session 失 效后跳转的页面，最大登陆次数 -->
  10. <session-management invalid-session-url="/sessionTimeout.htm">
  11. <concurrency-control max-sessions="1" expired-url="/sessionTimeout.htm" />
  12. </session-management>
  13. </http>
  14. 可 以使用SS自带的登陆页面作为login.jsp的模板

<http auto-config="true"> <!-- 指定登陆页面、成功页面、失败页面--> <form-login login-page="/login.jsp" default-target-url="/index.jsp" authentication-failure-url="/login.jsp" /> <!-- 尝试访问没有权限的页面时跳转的页面 --> <access-denied-handler error-page="/accessDenied.jsp"/> <!-- 使用记住用户名、密码功能，指定数据源和加密的key --> <remember-me data-source-ref="dataSource" /> <!-- logout页面，logout后清除session --> <logout invalidate-session="true" logout-success-url="/login.jsp" /> <!-- session 失效后跳转的页面，最大登陆次数 --> <session-management invalid-session-url="/sessionTimeout.htm"> <concurrency-control max-sessions="1" expired-url="/sessionTimeout.htm" /> </session-management> </http> 可以使用SS自带的登陆页面作为login.jsp的模板

1. **7：本地化消息输出**  
   拷贝本地化资源文件后，在配置文件中加载该文件：

Java代码

* 1. <!-- 加 载错误信息资源文件 -->
  2. <beans:bean id="messageSource"
  3. class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
  4. <beans:property name="basename" value="classpath:messages"/>
  5. </beans:bean>
  6. 资 源文件在SS核心包：spring-security-core-3.0.2.RELEASE.jar的orgspringframeworksecurity目录 中

<!-- 加载错误信息资源文件 --> <beans:bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSource"> <beans:property name="basename" value="classpath:messages"/> </beans:bean> 资源文件在SS核心包：spring-security-core-3.0.2.RELEASE.jar的orgspringframeworksecurity目录中

1. **8：在web页面中获取用户信息**

Java代码

* 1. 方式 一：Java代码
  2. Authentication auth = SecurityContextHolder.getContext().getAuthentication();
  3. Collection<GrantedAuthority> col = auth.getAuthorities();
  4. 方 式二：标签库
  5. <%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
  6. <sec:authentication property="name“/>
  7. <sec:authentication property="authorities“/>

方式一：Java代码 Authentication auth = SecurityContextHolder.getContext().getAuthentication(); Collection<GrantedAuthority> col = auth.getAuthorities(); 方式二：标签库 <%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %> <sec:authentication property="name“/> <sec:authentication property="authorities“/>

1. **9：在web页面进行元素安全控制**

Java代码

* 1. 方式一
  2. <sec:authorizeifAnyGranted="ROLE\_A">
  3. <a href="a.jsp">你可以访问a.jsp</a>
  4. </sec:authorize>
  5. <sec:authorizeifNotGranted="ROLE\_A">
  6. 你 不可以访问a.jsp
  7. </sec:authorize>
  8. 方 式二
  9. <sec:authorizeurl="/a.jsp">
  10. <a href="a.jsp">你可以访问a.jsp</a>
  11. </sec:authorize>

方式一 <sec:authorizeifAnyGranted="ROLE\_A"> <a href="a.jsp">你可以访问a.jsp</a> </sec:authorize> <sec:authorizeifNotGranted="ROLE\_A"> 你不可以访问a.jsp </sec:authorize> 方式二 <sec:authorizeurl="/a.jsp"> <a href="a.jsp">你可以访问a.jsp</a> </sec:authorize>

1. **10：全局方法安全控制**

Java代码

* 1. <global-method-security pre-post-annotations="enabled">
  2. <protect-pointcut expression="execution(\* com.xasxt.\*Service.add\*(..))" access="ROLE\_A"/>
  3. <protect-pointcut expression="execution(\* com.xasxt.\*Service.delete\*(..))" access="ROLE\_B"/>
  4. </global-method-security>
  5. 此 处使用了AspectJ中常用的切入点表达式（百度：AspectJ execution）

<global-method-security pre-post-annotations="enabled"> <protect-pointcut expression="execution(\* com.xasxt.\*Service.add\*(..))" access="ROLE\_A"/> <protect-pointcut expression="execution(\* com.xasxt.\*Service.delete\*(..))" access="ROLE\_B"/> </global-method-security> 此处使用了AspectJ中常用的切入点表达式（百度：AspectJ execution）

1. **11：使用注解进行方法安全控制**

Java代码

* 1. public class DemoService {
  2. @PreAuthorize("hasRole(&apos;ROLE\_A&apos;)")
  3. public void methodA() {
  4. }
  5. @PreAuthorize("hasAnyRole(&apos;ROLE\_A, ROLE\_B&apos;)")
  6. public void methodB() {
  7. }
  8. }
  9. hasRole 与hasAnyRole为SS通用内置表达式（google : spring security Common Built- In Expressions）

public class DemoService { @PreAuthorize("hasRole(&apos;ROLE\_A&apos;)") public void methodA() { } @PreAuthorize("hasAnyRole(&apos;ROLE\_A, ROLE\_B&apos;)") public void methodB() { } } hasRole与hasAnyRole为SS通用内置表达式（google : spring security Common Built-In Expressions）

1. **12：下一步做什么？？？**  
   采用更安全的验证方式  
   采用安全的数据传输方式  
   实现动态授权  
   自 定义验证与授权部件  
   实现数据级安全
2. 本Blog所有内容不得随意转载，版权属于作者所有。如需转载请与作者联系（ [fastzch@163.com](mailto:fastzch@163.com)    QQ：9184314）。  
   未经许可的转载，本人保留一切法律权益。  
   一直以来，发现有某些人完全不尊重我的劳动成果，随意转载，提醒一下那些人小心哪天惹上官司。  
     
   Spring Security 3.x 出来一段时间了，跟Acegi是大不同了，与2.x的版本也有一些小小的区别，网上有一些文档，也有人翻译Spring Security 3.x的guide，但通过阅读guide，无法马上就能很容易的实现一个完整的实例。  
     
   我花了点儿时间，根据以前的实战经验，整理了一份完整的入门教程，供需要的朋友们参考。  
   1，建一个web project，并导入所有需要的lib，这步就不多讲了。  
   2，配置web.xml，使用Spring的机制装载：
3. http://www.blogjava.net/Images/OutliningIndicators/None.gif<?xml version="1.0" encoding="UTF-8"?>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee   
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <context-param>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <param-name>contextConfigLocation</param-name>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <param-value>classpath:applicationContext\*.xml</param-value>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </context-param>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <listener>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <listener-class>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            org.springframework.web.context.ContextLoaderListener  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        </listener-class>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </listener>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <filter>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-name>springSecurityFilterChain</filter-name>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-class>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            org.springframework.web.filter.DelegatingFilterProxy  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        </filter-class>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </filter>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <filter-mapping>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <filter-name>springSecurityFilterChain</filter-name>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <url-pattern>/\*</url-pattern>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </filter-mapping>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <welcome-file-list>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <welcome-file>login.jsp</welcome-file>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </welcome-file-list>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif</web-app>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif
4. 这个文件中的内容我相信大家都很熟悉了，不再多说了。  
     
   2，来看看applicationContext-security.xml这个配置文件，关于Spring Security的配置均在其中：
5. http://www.blogjava.net/Images/OutliningIndicators/None.gif<?xml version="1.0" encoding="UTF-8"?>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif<beans:beans xmlns="http://www.springframework.org/schema/security"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:beans="http://www.springframework.org/schema/beans"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    xsi:schemaLocation="http://www.springframework.org/schema/beans  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif           http://www.springframework.org/schema/security  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif           http://www.springframework.org/schema/security/spring-security-3.0.xsd">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <http access-denied-page="/403.jsp"><!-- 当访问被拒绝时，会转到403.jsp -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <intercept-url pattern="/login.jsp" filters="none" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <form-login login-page="/login.jsp"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            authentication-failure-url="/login.jsp?error=true"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            default-target-url="/index.jsp" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <logout logout-success-url="/login.jsp" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <http-basic />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <!-- 增加一个filter，这点与Acegi是不一样的，不能修改默认的filter了，这个filter位于FILTER\_SECURITY\_INTERCEPTOR之前 -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <custom-filter before="FILTER\_SECURITY\_INTERCEPTOR"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            ref="myFilter" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </http>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 一个自定义的filter，必须包含authenticationManager,accessDecisionManager,securityMetadataSource三个属性，  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    我们的所有控制将在这三个类中实现，解释详见具体配置 -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <beans:bean id="myFilter" class="com.robin.erp.fwk.security.MyFilterSecurityInterceptor">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <beans:property name="authenticationManager"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            ref="authenticationManager" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <beans:property name="accessDecisionManager"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            ref="myAccessDecisionManagerBean" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <beans:property name="securityMetadataSource"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            ref="securityMetadataSource" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </beans:bean>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif      
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 认证管理器，实现用户认证的入口，主要实现UserDetailsService接口即可 -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <authentication-manager alias="authenticationManager">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        <authentication-provider  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            user-service-ref="myUserDetailService">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            <!--   如果用户的密码采用加密的话，可以加点“盐”  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif                <password-encoder hash="md5" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif            -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        </authentication-provider>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </authentication-manager>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <beans:bean id="myUserDetailService"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="com.robin.erp.fwk.security.MyUserDetailService" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 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 -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <beans:bean id="myAccessDecisionManagerBean"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="com.robin.erp.fwk.security.MyAccessDecisionManager">  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    </beans:bean>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif      
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <!-- 资源源数据定义，即定义某一资源可以被哪些角色访问 -->  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif    <beans:bean id="securityMetadataSource"  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif        class="com.robin.erp.fwk.security.MyInvocationSecurityMetadataSource" />  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif</beans:beans>  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif
6. 3，来看看自定义filter的实现：
7. http://www.blogjava.net/Images/OutliningIndicators/None.gifpackage com.robin.erp.fwk.security;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.io.IOException;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.Filter;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.FilterChain;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.FilterConfig;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.ServletException;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.ServletRequest;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport javax.servlet.ServletResponse;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.SecurityMetadataSource;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.intercept.AbstractSecurityInterceptor;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.intercept.InterceptorStatusToken;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.FilterInvocation;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.access.intercept.FilterInvocationSecurityMetadataSource;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifpublic class MyFilterSecurityInterceptor extends AbstractSecurityInterceptor  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif        implements Filter http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private FilterInvocationSecurityMetadataSource securityMetadataSource;  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // ~ Methods  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // ========================================================================================================  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    /\*\* \*//\*\*  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* Method that is actually called by the filter chain. Simply delegates to  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* the {@link #invoke(FilterInvocation)} method.  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*   
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param request  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*            the servlet request  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param response  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*            the servlet response  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @param chain  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*            the filter chain  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*   
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @throws IOException  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*             if the filter chain fails  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* @throws ServletException  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \*             if the filter chain fails  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif     \*/  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void doFilter(ServletRequest request, ServletResponse response,  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            FilterChain chain) throws IOException, ServletException http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        FilterInvocation fi = new FilterInvocation(request, response, chain);  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        invoke(fi);  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public FilterInvocationSecurityMetadataSource getSecurityMetadataSource() http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return this.securityMetadataSource;  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public Class<? extends Object> getSecureObjectClass() http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return FilterInvocation.class;  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void invoke(FilterInvocation fi) throws IOException,  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            ServletException http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        InterceptorStatusToken token = super.beforeInvocation(fi);  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        try http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            fi.getChain().doFilter(fi.getRequest(), fi.getResponse());  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        } finally http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            super.afterInvocation(token, null);  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public SecurityMetadataSource obtainSecurityMetadataSource() http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return this.securityMetadataSource;  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void setSecurityMetadataSource(  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            FilterInvocationSecurityMetadataSource newSource) http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        this.securityMetadataSource = newSource;  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public void destroy() http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public void init(FilterConfig arg0) throws ServletException http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}
8. 最核心的代码就是invoke方法中的InterceptorStatusToken token = super.beforeInvocation(fi);这一句，即在执行doFilter之前，进行权限的检查，而具体的实现已经交给accessDecisionManager了，下文中会讲述。  
     
   4，来看看authentication-provider的实现：
9. http://www.blogjava.net/Images/OutliningIndicators/None.gifpackage com.robin.erp.fwk.security;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.ArrayList;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Collection;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.dao.DataAccessException;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.GrantedAuthority;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.authority.GrantedAuthorityImpl;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.userdetails.User;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.userdetails.UserDetails;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.userdetails.UserDetailsService;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.userdetails.UsernameNotFoundException;  
   http://www.blogjava.net/Images/OutliningIndicators/None.gif  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gifpublic class MyUserDetailService implements UserDetailsService http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public UserDetails loadUserByUsername(String username)  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            throws UsernameNotFoundException, DataAccessException http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        Collection<GrantedAuthority> auths=new ArrayList<GrantedAuthority>();  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        GrantedAuthorityImpl auth2=new GrantedAuthorityImpl("ROLE\_ADMIN");  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        auths.add(auth2);  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        if(username.equals("robin1"))http://www.blogjava.net/Images/dot.gif{  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            auths=new ArrayList<GrantedAuthority>();  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            GrantedAuthorityImpl auth1=new GrantedAuthorityImpl("ROLE\_ROBIN");  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            auths.add(auth1);  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif          
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif//        User(String username, String password, boolean enabled, boolean accountNonExpired,  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif//                    boolean credentialsNonExpired, boolean accountNonLocked, Collection<GrantedAuthority> authorities) {  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        User user = new User(username,  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                "robin", true, true, true, true, auths);  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return user;  
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
   http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
   http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}
10. 在这个类中，你就可以从数据库中读入用户的密码，角色信息，是否锁定，账号是否过期等，我想这么简单的代码就不再多解释了。  
      
    5，对于资源的访问权限的定义，我们通过实现FilterInvocationSecurityMetadataSource这个接口来初始化数据。
11. http://www.blogjava.net/Images/OutliningIndicators/None.gifpackage com.robin.erp.fwk.security;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.ArrayList;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Collection;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.HashMap;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Iterator;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Map;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.ConfigAttribute;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.SecurityConfig;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.FilterInvocation;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.access.intercept.FilterInvocationSecurityMetadataSource;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.util.AntUrlPathMatcher;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.web.util.UrlMatcher;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif/\*\* \*//\*\*  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif \*   
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif \* 此类在初始化时，应该取到所有资源及其对应角色的定义  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif \*   
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif \* @author Robin  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif \*   
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif \*/  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifpublic class MyInvocationSecurityMetadataSource  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif        implements FilterInvocationSecurityMetadataSource http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private UrlMatcher urlMatcher = new AntUrlPathMatcher();;  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    private static Map<String, Collection<ConfigAttribute>> resourceMap = null;  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public MyInvocationSecurityMetadataSource() http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        loadResourceDefine();  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    private void loadResourceDefine() http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        resourceMap = new HashMap<String, Collection<ConfigAttribute>>();  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        Collection<ConfigAttribute> atts = new ArrayList<ConfigAttribute>();  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        ConfigAttribute ca = new SecurityConfig("ROLE\_ADMIN");  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        atts.add(ca);  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        resourceMap.put("/index.jsp", atts);  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        resourceMap.put("/i.jsp", atts);  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // According to a URL, Find out permission configuration of this URL.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public Collection<ConfigAttribute> getAttributes(Object object)  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            throws IllegalArgumentException http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        // guess object is a URL.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        String url = ((FilterInvocation)object).getRequestUrl();  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        Iterator<String> ite = resourceMap.keySet().iterator();  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        while (ite.hasNext()) http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            String resURL = ite.next();  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            if (urlMatcher.pathMatchesUrl(url, resURL)) http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                return resourceMap.get(resURL);  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return null;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public boolean supports(Class<?> clazz) http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return true;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif      
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public Collection<ConfigAttribute> getAllConfigAttributes() http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return null;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}
12. 看看loadResourceDefine方法，我在这里，假定index.jsp和i.jsp这两个资源，需要ROLE\_ADMIN角色的用户才能访问。  
    这个类中，还有一个最核心的地方，就是提供某个资源对应的权限定义，即getAttributes方法返回的结果。注意，我例子中使用的是AntUrlPathMatcher这个path matcher来检查URL是否与资源定义匹配，事实上你还要用正则的方式来匹配，或者自己实现一个matcher。  
      
    6，剩下的就是最终的决策了，make a decision，其实也很容易，呵呵。
13. http://www.blogjava.net/Images/OutliningIndicators/None.gifpackage com.robin.erp.fwk.security;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Collection;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport java.util.Iterator;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.AccessDecisionManager;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.AccessDeniedException;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.ConfigAttribute;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.access.SecurityConfig;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.authentication.InsufficientAuthenticationException;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.Authentication;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gifimport org.springframework.security.core.GrantedAuthority;  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gifpublic class MyAccessDecisionManager implements AccessDecisionManager http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    //In this method, need to compare authentication with configAttributes.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // 1, A object is a URL, a filter was find permission configuration by this URL, and pass to here.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // 2, Check authentication has attribute in permission configuration (configAttributes)  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    // 3, If not match corresponding authentication, throw a AccessDeniedException.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    public void decide(Authentication authentication, Object object,  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            Collection<ConfigAttribute> configAttributes)  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            throws AccessDeniedException, InsufficientAuthenticationException http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        if(configAttributes == null)http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            return ;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        System.out.println(object.toString());  //object is a URL.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        Iterator<ConfigAttribute> ite=configAttributes.iterator();  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        while(ite.hasNext())http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            ConfigAttribute ca=ite.next();  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            String needRole=((SecurityConfig)ca).getAttribute();  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            for(GrantedAuthority ga:authentication.getAuthorities())http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                if(needRole.equals(ga.getAuthority()))http://www.blogjava.net/Images/dot.gif{  //ga is user's role.  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    return;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                }  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        throw new AccessDeniedException("no right");  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public boolean supports(ConfigAttribute attribute) http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        // TODO Auto-generated method stub  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return true;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif    @Override  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif    public boolean supports(Class<?> clazz) http://www.blogjava.net/Images/dot.gif{  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return true;  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
    http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
    http://www.blogjava.net/Images/OutliningIndicators/None.gif
14. 在这个类中，最重要的是decide方法，如果不存在对该资源的定义，直接放行；否则，如果找到正确的角色，即认为拥有权限，并放行，否则throw new AccessDeniedException("no right");这样，就会进入上面提到的403.jsp页面。  
      
    参考资料：  
    1，Spring官方网站：<http://www.springframework.org>  
    2，文章所用的代码，MyEclipse工程，去掉了lib，请自行下载Spring Security 3.x的包，并copy至对应目录。[工程源代码](http://download.csdn.net/source/2113130)  
    3，根据网络上的资料，制作的CHM版的 [Spring Security 3.x 参考手册中文版](http://download.csdn.net/source/2112823)  
    4，2009年3月，我在“IBM WebSphere技术专家沙龙（华南区广州站）”演讲时的PPT：[《Spring Security--Protect your web application》](http://www.webspherechina.net/club/thread-7823-1-1.html)，当时是Spring Security 2.x，很多原理是一样，可作参考。  
      
    教程中为了尽可能不跟其它框架关联上，所以去掉了访问数据库的部分，比如用户信息和资源配置信息的读取，直接写死在代码中了，大家可以根据自己的实际情况补充完整。  
      
    如有任何疑问，欢迎大家以评论的方式提问，也欢迎大家讨论！