先在pom中添加spring security依赖

<spring-security.version>4.0.3.RELEASE</spring-security.version>

<!-- spring security -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>${spring-security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring-security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring-security.version}</version>

</dependency>

在web.xml中添安全过滤器:

<!-- Spring Security -->

<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>

添加配置文件位置:

1. **<context-param>**
2. **<param-name>**contextConfigLocation**</param-name>**
3. **<param-value>**
4. classpath:spring/spring-security.xml
5. classpath:spring/applicationContext.xml
6. **</param-value>**
7. **</context-param>**

这里两处关于springsecurity的配置表示项目中所有路径的资源都要经过Spring Security。注意：最好是将DelegatingFilterProxy写在DispatcherServlet之前，否则Spring Security可能不会正常工作。

配置spring-security.xml:

添加头部声明

1. **<beans** xmlns="http://www.springframework.org/schema/beans"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns:security="http://www.springframework.org/schema/security"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
6. http://www.springframework.org/schema/security
7. http://www.springframework.org/schema/security/spring-security-3.2.xsd"**>**

对静态资源不进行过滤(可以不进行静态资源过滤而指定需要登录的界面过滤,如下下)

<!--静态资源过滤-->

<security:http pattern="/\*\*/\*.css" security="none"></security:http>

<security:http pattern="/\*\*/\*.js" security="none"></security:http>

<security:http pattern="/\*\*/\*.jpg" security="none"></security:http>

<security:http pattern="/\*\*/\*.gif" security="none"></security:http>

<security:http pattern="/\*\*/\*.png" security="none"></security:http>

拦截管理界面和编辑界面(登录成功则请求manageBlog.html否则返回login.html)

<security:http auto-config="true">

<security:intercept-url pattern="/manageBlog.html" access="hasRole('ROLE\_ADMIN')" />

<security:intercept-url pattern="/editor.html" access="hasRole('ROLE\_ADMIN')" />

<security:form-login

login-page="/login.html"

authentication-failure-url="/login.html?error=true"

default-target-url="/manageBlog.html"/>

<security:csrf disabled="true"/> <!—禁用csrf,谁吃饱了攻击我这个破站-->

</security:http>

定义admin用户,设置MD5编码

<security:authentication-manager>

<security:authentication-provider>

<security:user-service>

<security:user name="wu" password="941114" authorities="ROLE\_ADMIN"/>

</security:user-service>

<security:password-encoder class="org.springframework.security.authentication.encoding.Md5PasswordENcoder" />

</security:authentication-provider>

</security:authentication-manager>

上面默认没有登录的用户转到登录界面,但是login.jsp的界面转到哪个action?

spring security 3.x 默认的登录拦截URL是/j\_spring\_security\_check,而spring security 4.x默认拦截的URL是/login(<http://blog.csdn.net/yin380697242/article/details/51893397>).

在spring-security配置中还需指明页面中登录名密码的字段名:

username-parameter=*"user\_name"*

password-parameter=*"user\_password"*

卡了一整天才发现为什么自己写死的用户密码老登录不上,我吃饱了转什么MD5啊,xml中还是原来的密码就导致根本匹配不上

还有个登录老被系统赋值了ANONYMOUSLY,然后登录拒绝:把登录界面加上

<security:intercept-url pattern="/login.html" access="IS\_AUTHENTICATED\_ANONYMOUSLY"/>

然而上面这句话在使用spring默认界面/login 时有效,但是使用自己的/login.html时,一直报错:Failed to evaluate expression 'IS\_AUTHENTICATED\_ANONYMOUSLY'

查了无数的网站有说改成access="hasRole('ROLE\_USER)",但是没用,后来自己尝试了一下hasRole('ROLE\_ANONYMOUS')通过了?!!!!全世界就没人这么写吗?!!!!

登录完之后的登录信息没保存session?默认是自动保存好的,还把csrf关了.

完整的spring-security.xml配置文件:

<?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:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

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

*http://www.springframework.org/schema/security/spring-security-4.0.xsd"*>

<security:http pattern=*"/login/loading.gif"* security=*"none"*></security:http>

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

<security:intercept-url pattern=*"/login.html"* access=*"hasRole('ROLE\_ANONYMOUS')"*/>

<security:intercept-url pattern=*"/manageBlog.html"* access=*"hasRole('ROLE\_ADMIN')"* />

<security:intercept-url pattern=*"/editor.html"* access=*"hasRole('ROLE\_ADMIN')"* />

<!-- <security:intercept-url pattern="/\*\*" access="hasRole('ROLE\_ADMIN')"/> -->

<!-- login-page="/login.html"

username-parameter="user\_name"

password-parameter="user\_password"

authentication-failure-url="/login.html?error='true'"

always-use-default-target="true"-->

<security:form-login

login-page=*"/login.html"*

username-parameter=*"user\_name"*

password-parameter=*"user\_password"*

default-target-url=*"/manageBlog.html"*

/>

<security:csrf disabled=*"true"*/>

</security:http>

<security:authentication-manager>

<security:authentication-provider>

<security:user-service>

<security:user name=*"wu"* password=*"941114"* authorities=*"ROLE\_ADMIN"*/>

</security:user-service>

</security:authentication-provider>

</security:authentication-manager>

</beans>

Eclispe中的maven项目使用的框架报这个错:[**Java**](http://lib.csdn.net/base/17).lang.ClassNotFoundException: org.springframework.web.context.ContextLoaderListener

但这个包确实已经引入了，但是它却一直找不到，原因是少了一个设置，解决方法如下：

1.   Add maven dependencies

    Right click the web project “properties” -------> click the “Deployment Assembly” ---->

    Click add button

          Choose Java Build Path Entires ---> click Next

    Choose Maven Dependencies------>  Finish

部署了nginx后,maven无法远程部署项目,一直断开连接,但是把nginx代理关了又成功了,但是这个问题不好描述也不应该出现,初步判断是在path中的路径应该不对,可能相对路径部署到了nginx中.

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<encoding>UTF-8</encoding>

<charset>UTF-8</charset>

<url>http://112.74.165.23/manager/text</url>

<username>wujialong</username>

<password>wjl941114.</password>

<!-- <update>true</update>-->

<path>/blog</path>

<port>80</port>

</configuration>

</plugin>

</plugins>

部署blog:

1. 远程部署项目,同上
2. 建立数据库blog:建立6张表,最关键的为t\_blog和两个视图以及t\_blogtype

建立表存储过程:

Delimiter //

create PROCEDURE auto\_insert(num int)

BEGIN

DECLARE s1 char(1) DEFAULT '第';

DECLARE s2 char(5) DEFAULT '篇文章..';

DECLARE s3 char(10);

DECLARE i int DEFAULT 1;

while i<=num DO

SET s3=CONCAT(s1,i,s2);

insert into t\_blog(title,summary,releaseDate,type,author,clickHit) values

(s3,'摘要摘要摘要',NOW(),'测试','乌龟',2);

SET i=i+1;

END WHILE;

END //

同时插入t\_blogtype表:

delimiter //

create PROCEDURE auto\_insertBlogType(num int)

BEGIN

DECLARE i int DEFAULT 1;

while i<=num DO

insert into t\_blogtype(type,blogID) values

(‘测试’,i);

SET i=i+1;

END WHILE;

END //