eryk

浏览: 88717 次

性别:

来自: 北京

最近访客 更多访客>>

meler aaaaaaall

aiyanxu sohu001

文章分类

- 全部博客 (279)
- Java (49)
- linux (66)
- nutch (6)
- ajax+JQuery (4)
- Hibernate (9)
- Eclipse/MyEclipse (7)
- hadoop (30)
- hbase (2)
- lucene (14)
- Java基础 (17)
- 数据库 (10)
- Perl (6)
- Spring (9)
- struts (7)

spring security 11种过滤器介绍

Security Spring Access JSP J#

1.HttpSessionContextIntegrationFilter

位于过滤器顶端，第一个起作用的过滤器。

用途一，在执行其他过滤器之前，率先判断用户的session中是否已经存在一个SecurityContext了。如果存在，就把SecurityContext拿出来，放到SecurityContextHolder中，供Spring Security的其他部分使用。如果不存在，就创建一个SecurityContext出来，还是放到SecurityContextHolder中，供Spring Security的其他部分使用。

用途二，在所有过滤器执行完毕后，清空SecurityContextHolder，因为SecurityContextHolder是基于ThreadLocal的，如果在操作完成后清空ThreadLocal，会受到服务器的线程池机制的影响。

2.LogoutFilter

只处理注销请求，默认为/j_spring_security_logout。

用途是在用户发送注销请求时，销毁用户session，清空SecurityContextHolder，然后重定向到注销成功页面。可以与rememberMe之类的机制结合，在注销的同时清空用户cookie。

3.AuthenticationProcessingFilter

处理form登陆的过滤器，与form登陆有关的所有操作都是在此进行的。
默认情况下只处理/j_spring_security_check请求，这个请求应该是用户使用form登陆后的提交地址，form所需的其他参数可以参考：

此过滤器执行的基本操作时，通过用户名和密码判断用户是否有效，如果登录成功就跳转到成功页面（可能是登陆之前访问的受保护页面，也可能是默认的成功页面），如果登录失败，就跳转到失败页面。

```
<form action="${pageContext.request.contextPath}/j_spring_security_check" style="width:260px;text-align:center;">
  <fieldset>
    <legend>登陆</legend>
    用户： <input type="text" name="j_username" style="width:150px;"
value="${sessionScope['SPRING_SECURITY_LAST_USERNAME']}"/><br />
```

- Flex4 (7)
- 常用工具类 (1)
- 其它 (13)
- zookeeper (8)
- java 桌面 (4)

社区版块
- 我的资讯 (0)
- 我的论坛 (3)
- 我解决的问题 (0)

存档分类
- 2011-12 (8)
- 2011-11 (10)
- 2011-10 (17)
- 更多存档...

评论排行榜
- zookeeper3.3学习笔记2：配置参数介绍

最新评论

eryk： gaohaizhao777 写道这个原因其实是因为在执行sto ...
hadoop执行stop-all.sh的时候总是出现 “no namenode to stop”

gaohaizhao777： 这个原因其实是因为在执行stop-all.sh时，找不到pid ...
hadoop执行stop-all.sh的时候总是出现 “no namenode to stop”

xxjjyy2008： 我怎么不能运行哦，还有错
《hadoop权威指南》第二章的例子（修改已可用）

xxjjyy2008： 楼主QQ留一下。。。
《hadoop权威指南》第二章的例子（修改已可用）

eryk： xrogzu 写道格式化，太不好了吧可以修改 “ha ...
hadoop执行stop-all.sh的时候总是出现 “no namenode to stop”


```
密码： <input type="password" name="j_password" style="width:150px;" /><br />
<input type="checkbox" name="_spring_security_remember_me" />两周之内不必登陆<br />
<input type="submit" value="登陆"/>
<input type="reset" value="重置"/>
</fieldset>
</form>
```

/j_spring_security_check，提交登陆信息的URL地址。
自定义form时，要把form的action设置为/j_spring_security_check。注意这里要使用绝对路径，避免登陆页面存放的页面可能带来的问题。

j_username，输入登陆名的参数名称。

j_password，输入密码的参数名称

_spring_security_remember_me，选择是否允许自动登录的参数名称。
可以直接把这个参数设置为一个checkbox，无需设置value，Spring Security会自行判断它是否被选中。

4.DefaultLoginPageGeneratingFilter

此过滤器用来生成一个默认的登录页面，默认的访问地址为/spring_security_login，这个默认的登录页面虽然支持用户输入用户名，密码，也支持rememberMe功能，但是因为太难看了，只能是在演示时做个样子，不可能直接用在实际项目中。

自定义登陆页面

```
<http auto-config='true'>
  <intercept-url pattern="/login.jsp" access="IS_AUTHENTICATED_ANONYMOUSLY" />
  <intercept-url pattern="/admin.jsp" access="ROLE_ADMIN" />
  <intercept-url pattern="/**" access="ROLE_USER" />
  <form-login login-page="/login.jsp"
        authentication-failure-url="/login.jsp?error=true"
        default-target-url="/" />
</http>
```

5.BasicProcessingFilter

此过滤器用于进行basic验证，功能与AuthenticationProcessingFilter类似，只是验证的方式不同。
添加basic认证，去掉auto-config="true"，并加上<http-basic />

```
<http auto-config="true">
  <http-basic />
  <intercept-url pattern="/admin.jsp" access="ROLE_ADMIN" />
  <intercept-url pattern="/" access="ROLE_USER" />
</http>
```

6.SecurityContextHolderAwareRequestFilter

此过滤器用来包装客户的请求。目的是在原始请求的基础上，为后续程序提供一些额外的数据。比如getRemoteUser()时直接返回当前登陆的用户名之类的。

7.RememberMeProcessingFilter

此过滤器实现RememberMe功能，当用户cookie中存在rememberMe的标记，此过滤器会根据标记自动实现用户登陆，并创建SecurityContext，授予对应的权限。
在配置文件中使用auto-config="true"就会自动启用rememberMe
实际上，Spring Security中的rememberMe是依赖cookie实现的，当用户在登录时选择使用rememberMe，系统就会在登录成功后将为用户生成一个唯一标识，并将这个标识保存进cookie中，我们可以通过浏览器查看用户电脑中的cookie。

8.AnonymousProcessingFilter

为了保证操作统一性，当用户没有登陆时，默认为用户分配匿名用户的权限。
在配置文件中使用auto-config="true"就会启用匿名登录功能。在启用匿名登录之后，如果我们希望允许未登录就可以访问一些资源，可以在进行如下配置。

```
<http auto-config='true'>
   <intercept-url pattern="/" access="IS_AUTHENTICATED_ANONYMOUSLY" />
   <intercept-url pattern="/admin.jsp" access="ROLE_ADMIN" />
   <intercept-url pattern="/**" access="ROLE_USER" />
</http>
```

设置成 ROLE_ANONYMOUS 也可以。

```
<http auto-config='true'>
   <intercept-url pattern="/" filters="none" />
   <intercept-url pattern="/admin.jsp" access="ROLE_ADMIN" />
   <intercept-url pattern="/**" access="ROLE_USER" />
</http>
```

filters="none"表示当我们访问“/”时，是不会使用任何一个过滤器去处理这个请求的，它可以实现无需登录即可访问资源的效果，但是因为没有使用过滤器对请求进行处理，所以也无法利用安全过滤器为我们带来的好处，最简单的，这时SecurityContext内再没有保存任何一个权限主体了，我们也无法从中取得主体名称以及对应的权限信息。

9.ExceptionTranslationFilter

此过滤器的作用是处理中FilterSecurityInterceptor抛出的异常，然后将请求重定向到对应页面，或返回对应的响应错误代码。

10.SessionFixationProtectionFilter

防御会话伪造攻击。
解决session fix的问题其实很简单，只要在用户登录成功之后，销毁用户的当前session，并重新生成一个session就可以了。

```
<http auto-config='true' session-fixation-protection="none">
   <intercept-url pattern="/admin.jsp" access="ROLE_ADMIN" />
   <intercept-url pattern="/**" access="ROLE_USER" />
```

</http>

session-fixation-protection的值共有三个可供选择，none，migrateSession和newSession。默认使用的是migrationSession

---

11.FilterSecurityInterceptor

用户的权限控制都包含在这个过滤器中。
功能一：如果用户尚未登陆，则抛出AuthenticationCredentialsNotFoundException“尚未认证异常”。
功能二：如果用户已登录，但是没有访问当前资源的权限，则抛出AccessDeniedException“拒绝访问异常”。
功能三：如果用户已登录，也具有访问当前资源的权限，则放行。

总结来源：http://www.family168.com