

声明：JavaEye 文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2010 JavaEye.com. All rights reserved. 上海炯耐计算机软件有限公司 [ 沪ICP备05023328号 ]

2009-03-02

[JSON通用服务端处理](#) | [nginx+tomcat集群负载均衡\(实现session复 ...](#)

## [spring security 2.0 的简单配置使用](#)

关键字: spring security 2.0 的简单配置使用

<http://snz.javaeye.com/blog/221280>

由于ss2的demo配置太过简单，要想在项目中应用的话必须进行相应扩展，这里简单写一下简单的扩展方法。

xml头中引入security命名空间

Xml代码

```
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 http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
5.         http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-2.0.xsd"
6.     default-lazy-init="true">
```

然后是启用ss2默认配置的一段代码

Xml代码

```
1. <!--
2.     2.0新增的命名空间，使得配置简化了很多
3.     auto-config 自动使用默认的配置
4.     access-denied-page 指定访问未授权页面时显示的页面
5. -->
6. <security:http auto-config="true" access-denied-page="/accessDenied.html">
7.     <security:anonymous granted-authority="BASIC" />
8. </security:http>
```

这段代码作用是以ss2的默认配置方式加入1.0时需要手工配置的AuthenticationProcessingFilter等多个必须配置的filter，详细可参考1.0配置和2.0参考手册。

auto-config="true" 表示使用ss2自动配置

access-denied-page="/accessDenied.html"表示拒绝访问时显示的页面

<security:anonymous granted-authority="BASIC" />表示匿名权限的authority为BASIC

ss对权限的管理分为认证和授权两部分，先看认证

Xml代码

```
1. <!--
2.     负责认证处理的filter
3. -->
4. <bean id="authenticationProcessingFilter"
5.     class="org.springframework.security.ui.webapp.AuthenticationProcessingFilter">
6.     <!-- 此行说明此filter会覆盖ss2默认配置的filter，before 被覆盖filter的别名 -->
7.     <security:custom-filter before="AUTHENTICATION_PROCESSING_FILTER" />
8.     <!-- 认证管理器 -->
9.     <property name="authenticationManager" ref="authenticationManager" />
10.    <!-- 认证失败后跳转到的页面，/spring_security_login是ss2默认的登录入口 -->
11.    <property name="authenticationFailureUrl"
12.        value="/spring_security_login" />
13.    <!-- 认证成功后跳转到的页面 -->
14.    <property name="defaultTargetUrl" value="/index.html" />
15. </bean>
```

这是负责认证处理的filter，中间custom-filter一行意思是将filter放在默认配置中别名AUTHENTICATION\_PROCESSING\_FILTER的filter前边，即负责认证的filter（别名列表参照参考手册）。

按官方的说法，如果需要用自定义的filter覆盖默认filter，则应该将security:http标签的auto-config属性改为false，这样的话就需要增加很多手动配置项。我试了下，不改false也可以，只是运行期间会出现一个warn信息“Possible error: Filters at position 2 and 3 are both instances of xxxx”，意思是filter串中有两个相同类型的filter。

另：在2.0.2中可以使用position代替before，真正的覆盖默认filter。但是有个bug，如果使用默认登录入口的话，还是会调用默认filter，必须连登录入口一并改掉。

其引用的authenticationManager

Xml代码

```
1. <!--
2.     认证管理器
3.     根据用户名和密码，使用多个provider进行认证
4.     认证成功会生成一个Authentication，否则抛出AuthenticationException
5. -->
6. <bean id="authenticationManager"
7.     class="org.springframework.security.providers.ProviderManager">
8.     <property name="providers">
9.         <list>
10.             <ref local="daoAuthenticationProvider" />
11.         </list>
12.     </property>
13. </bean>
```

认证管理器通过多个provider实现基于用户名和密码的认证，多个provider中只要有一个认证成功，即成功。

这里只使用了一个daoPorvider

Xml代码

```
1. <!--
2.     认证的provider
3.     userDetailsService 根据用户名获取用户信息
4.     userCache ehcache缓存user信息。
5. -->
6. <bean id="daoAuthenticationProvider"
7.     class="org.springframework.security.providers.dao.DaoAuthenticationProvider">
8.     <property name="userDetailsService" ref="userDetailsService" />
9.     <property name="userCache" ref="userCache" />
10. </bean>
```

userDetailsService：根据登录的用户名获取一个UserDetails，即代表一个用户的实体对象。

Xml代码

```
1. <!-- 通过dao查询用户信息 -->
2. <bean id="userDetailsService"
3.     class="org.catspaw.ss2test1.security.UserDetailsSerivceImpl">
4.     <property name="userDao" ref="userDao" />
5. </bean>
```

UserDetailsSerivceImpl代码

Java代码

```
1. package org.catspaw.ss2test1.security;
2.
3. import org.catspaw.ss2test1.dao.UserDao;
4. import org.springframework.dao.DataAccessException;
5. import org.springframework.security.userdetails.UserDetails;
6. import org.springframework.security.userdetails.UserDetailsService;
7. import org.springframework.security.userdetails.UsernameNotFoundException;
8.
9. /**
10.  * 获取UserDetails
11.  * 使用UserDao查询User
12.  *
13.  * @author 孙宁振
14.  */
```

```
15.  */
16.  public class UserDetailsSerivceImpl implements UserDetailsService {
17.
18.      private UserDao userDao;
19.
20.      public UserDao getUserDao() {
21.          return userDao;
22.      }
23.
24.      public void setUserDao(UserDao userDao) {
25.
```

[Spring Casters & Wheels](#)

One Stop Company For All Casters & Wheels. Top Service. Same Day Ship.  
[www.ServiceCaster.com](http://www.ServiceCaster.com)

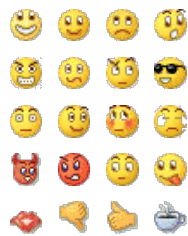
[JSON通用服务端处理](#) | [nginx+tomcat集群负载均衡\(实现session复 ...](#)

16:14 | [浏览 \(934\)](#) | [评论 \(0\)](#) | 分类: [spring acegi](#) | [相关推荐](#)

评论

发表评论

表情图标



字体颜色:  字体大小:  对齐:

提示: 选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录, 请[登录](#)后发表评论(快捷键 Alt+S / Ctrl+Enter)



talangniao

浏览: 22095 次

来自: ...



[详细资料](#)

[留言簿](#)

搜索本博客

最近访客 [>>更多访客](#)



[feng\\_1086](#)



[81641143](#)



[finallygo](#)

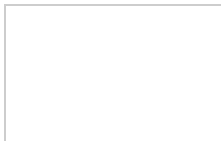


[wuxianjun](#)

博客分类

- [全部博客 \(54\)](#)
- [Word 操作管理 \(1\)](#)
- [prototype1.6 \(2\)](#)
- [spring acegi \(2\)](#)
- [spring2 \(0\)](#)
- [hibernate3 \(7\)](#)
- [js操作 \(5\)](#)
- [struts2 \(4\)](#)
- [springside3 \(0\)](#)
- [struts1 \(1\)](#)
- [springside2 \(1\)](#)
- [公共类库 \(1\)](#)
- [freemarker \(6\)](#)
- [登陆实现 \(2\)](#)
- [jquery \(2\)](#)
- [nginx负载均衡 \(3\)](#)
- [json \(2\)](#)
- [java 基础 \(2\)](#)
- [杂文 \(2\)](#)
- [搜索引擎 \(2\)](#)
- [css \(1\)](#)

我的相册





复件 (9) 复件  
taobao.products.get获取  
多个产品.png

[共 1 张](#)

其他分类

- [我的收藏 \(25\)](#)
- [我的论坛主题贴 \(2\)](#)
- [我的所有论坛贴 \(47\)](#)
- [我的精华良好贴 \(0\)](#)

最近加入圈子

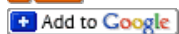
- [理财投资圈](#)
- [GT-Grid](#)

存档

- [2010-07 \(1\)](#)
- [2010-05 \(3\)](#)
- [2010-03 \(2\)](#)
- [更多存档...](#)

评论排行榜

- [jdbcTemplate分页](#)
- [JS 中如何判断 undefined,null](#)
- [PL/SQL 中文乱码解决方案\(转载\)](#)
- [编辑器](#)
- [hibernate 排序结果](#)



[\[什么是RSS?\]](#)