作者 正文

likunkun

等级:

文章: 23

积分: 165

来自: 北京

我现在离线

发表时间：2007-01-06

相关文章:

- 使用acegi 控制用户权限
- acegi过滤器
- 学习Acegi-认证(authentication)

推荐圈子: Pipboy

更多相关推荐

Acegi配置文档

作者：javafish(likunkun)

Email:javafish@sunxin.org

Acegi是基于Spring的一个开源的安全认证框架，现在的最新版本是1.04。Acegi的特点就是有很多的过滤器：不过我们也用不到这么多的过滤器，只是可以把它们看作为一个个的模块，在用的时候加上自己用的着的即可，由于认证的流程的方面比较复杂导致它的配置很复杂，如果能摸清它的工作原理还是不太难.下面用比较顺着人思维的流程过一遍

这里只列出常用的过滤器和拦载器

1. 过滤器：HttpSessionContextIntegrationFilter，authenticationProcessingFilter，BasicProcessingFilter，RememberMeProcessingFilter，anonymousProcessingFilter,exceptionTranslationFilter

2. 拦截器：filterSecurityInterceptor（其实它是过滤器，不过把它放在这里更能说明它的功能），methodSecurityInterceptor

看着上面的用红色标出的过滤器是用来认证（表单和HTTP基本认证，当然还有别的不过这两个比较长用）它们是资源访问的入口.其它的过滤器是用来辅助的：HttpSessionContextIntegrationFilter是用来把认证信息记录到Session中的RememberMeProcessingFilter是以cookie的形式来保存认证信息的.anonymousProcessingFilter是在匿名的时候（这时候是没有认证信息的）给这个用户分配一个匿名的认证信息，exceptionTranslationFilter总结一下异常并处理.在实际中选择适合程序的即可.

上面只是资源访问的入口，真正保护资源的是这两个拦截器：filterSecurityInterceptor，拦截URL的类（它是个过滤器）metohdSecurityInterceptor，拦截类中方法的调用，它们为什么要拦截呢？就是想在访问或调用这些方法之前来判断一下用户是否有访问或调用的权限，有就通过，没有就踢出.

除此之外，Acegi专门做了两个管理器（实际上就是两个类，为什么会用做这两个管理器，因为认证和授权都有一些的操作，这就需要专门做两个管理器了）：authenticationManager（class= org.acegisecurity.providers.ProviderManager），授权管理器accessDecisionManager（class=org.acegisecurity.vote.AffirmativeBased）
说白了一个用于认证用户，一个是用于权限的授于的
先来说认证用户，认证管理器有什么东西呢？只内置了一些提供者：这些提供者呢又是什么呢，他们是提供用户的验证身份信息的，比如从数据库或配置文件里读出用户名和密码，在用户的cookie里读出身份信息（rememberMeProcessingFilter用到的[前面讲了的，有印象吧]），或在Session里读出身份验证信息（HttpSessionContextIntegrationFilter起作用的），这里我们只说一下从数据库或配置文件里读出用户名密码来装配验证信息的，其它的配置类似可以找一下对应api在Spring里配置即可，daoAuthenticationProvider是数据库的提供者class=org.acegisecurity.providers.dao.DaoAuthenticationProvider，而它提供的服务呢又有几种，数据库和配置文件（这是Acegi的两个默认的实现）当然也可以自己实现（实现userDetailsService接口就行）

Java代码

```
<bean id="authenticationManager" class="org.acegisecurity.providers.ProviderManager">
                <property name="providers">
                        <list>
                                <ref local="daoAuthenticationProvider"/>
                        </list>
                </property>
        </bean>
<bean id="daoAuthenticationProvider" class="org.acegisecurity.providers.dao.DaoAuthenticationProvider">
                <!-- <property name="userDetailsService"><ref local="InMemoryDaoImpl"/></property> --><!--
 这里有两种选择 -->
                <property name="userDetailsService"><ref local="jdbcDaoImpl"/></property>
        </bean>
```

如果用户名和密码在配置文件里可以用InMemoryDaoImpl，class=org.acegisecurity.userdetails.memory.InMemoryDaoImpl，在这个类的userMap里配置即可：javafish=java,ROLE_USER，配置了一个用户名为javafish,密码为java,用户组为ROLE_USER的用户，不过最常用的还是数据库的JDBC实现（两个二选一）org.acegisecurity.userdetails.jdbc.JdbcDaoImpl里面需要usersByUsernameQuery和authoritiesByUsernameQuery还有数据源dataSource（有人问为什么呢，userByUsernameQuery是用来通过用户名来查密码的，authoritiesByUsernameQuery是用来通过用户名来查权限的，查询数据库肯定的用数据源吧这个里是用的SpringFrameWork的DataSource）它们查询的sql语句是有讲究的，就是查密码的时候查三个第一个是username，第二个是password，第三个是是否可用，查权限的时候查两个：username和authorities（具体看例子）

Java代码

```
<bean id="InMemoryDaoImpl" class="org.acegisecurity.userdetails.memory.InMemoryDaoImpl">
                <property name="userMap">
                        <value>
                                javafish=java,ROLE_USER
                        </value>
                </property>
        </bean>
        <bean id="jdbcDaoImpl" class="org.acegisecurity.userdetails.jdbc.JdbcDaoImpl">
                <property name="usersByUsernameQuery">
                        <value>select username,password,enabled from users where username=?</value>
                </property>
```

```
                <property name="authoritiesByUsernameQuery">
                        <value>select username,authority from authorities where username=?</value>
                </property>
                <property name="dataSource">
                        <ref local="dataSource"/>
                </property>
        </bean>

        <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName">
            <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
            <value>jdbc:mysql://localhost:3306/test</value>
        </property>
        <property name="username">
            <value>root</value>
        </property>
        <property name="password">
            <value>javafish</value>
        </property>
    </bean>
```

下面说一下授权，授权管理器又有什么东西呢？accessDecisionManager，Acegi把授权方面弄的比较的形象化，把某个URL或方法是否可以被访问按投票的形式来决定，

Acegi提出来了几种方案：
1. 如果有一个赞成就同意（具体的说就是只要你在那个URL对应的几个用户组中的一个就让你访问）
2. 如果都赞成就同意（具本的说就是那个URL对应的几个用户组里都有你，你才能访问）
3. 如果都不反对就同意（这个在下面讲投票者的时候再说）

Java代码

```
<bean id="accessDecisionManager" class="org.acegisecurity.vote.AffirmativeBased">
                <property name="allowIfAllAbstainDecisions"><!-- 是否让全部弃权的通过 -->
                        <value>false</value>
                </property>
                <property name="decisionVoters"><!-- 投票者们 -->
                        <ref bean="roleVoter"/>
                </property>
        </bean>
```

而投票者呢：Acegi自己实现了一个投票者的类RoleVoter：
现在我用第一种方案，RoleVoter只是在URL对应的用户组里有ROLE_为前缀的才进行投票，否则的话弃权.（我们也可以在配置RoleVoter的时候把ROLE_配置成为别的前缀如JAVA_），分别对URL对应的每个用户组投票，如果用户在这个用户组里就投赞成，不在投反对（在用户组的前缀

是ROLE_的前提下）这样就不难体会第三种方案的用途了吧

Java代码

```
<bean id="roleVoter" class="org.acegisecurity.vote.RoleVoter">
                <property name="rolePrefix">
                        <value>ROLE_</value><!-- 可以改成别的 -->
                </property>
        </bean>
```

这样认证管理器和授权管理器就ok了，别的无论是过滤器还是拦截器都会用到它们两个，因为它们都要验证而这两个就是凭证.
那么那两个访问过滤器呢，先说authenticationProcessingFilter是用于表单登陆的

Java代码

```
<bean id="authenticationProcessingFilter" class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter"
>
                <property name="authenticationManager"><ref bean="authenticationManager"/></property>
                <property name="authenticationFailureUrl"><value>/failure.html</value></property><!--登陆失
败转向的页面  -->
                <property name="defaultTargetUrl"><value>/ok.html</value></property><!-- 登陆成功转向的页面
-->
        <property name="filterProcessesUrl"><value>/check</value></property><!-- 要验证的地址 -->
        </bean>
```

这样的话加上上面配置的认证管理器就已经可以处理登陆了（注意的是它没有用到授权管理器，因为它只是个访问入口还没有权限的授予）
再说一下HTTP基本认证：它比上面的略复杂一点
需要配置一个

Java代码

```
<bean id="BasicProcessingFilterEntryPoint" class="org.acegisecurity.ui.basicauth.BasicProcessingFilterEntryP
oint">
                <property name="realmName"><value>javafish</value></property><!-- 基本认证对话框上显示的字 -
->
        </bean>
然后
<bean id="BasicProcessingFilter" class="org.acegisecurity.ui.basicauth.BasicProcessingFilter">
                <property name="authenticationManager">
                        <ref bean="authenticationManager"/>
                </property>
                <property name="authenticationEntryPoint">
                        <ref bean="BasicProcessingFilterEntryPoint"/>
                </property>
        </bean>
```

即可.

不过在HTTP基本认证里需要注意的地方是：好多人配置好了怎么看不到效果啊，一开始我也是很郁闷，看了BasicProcessingFilter的源代码：String header = httpRequest.getHeader("Authorization");／／我们一般进入网页测试的时候这里的header始终是null的

Java代码

```
if (logger.isDebugEnabled()) {
            logger.debug("Authorization header: " + header);
        }
        if ((header != null) && header.startsWith("Basic ")) {//从这里可以看到一般的登陆基本认证是不起作用的
.................
```

只有在服务器上配置哪个目录在访问的时候用HTTP基本认证，它才会起作用（一开始还以为是Acegi的BUG呢）
下面说一下真正对URL资源的保护了filterSecurityInterceptor它的本质是个过滤器，有了前面＊管理器的基础了这就很容易了：

Java代码

```
<bean id="filterSecurityInterceptor" class="org.acegisecurity.intercept.web.FilterSecurityInterceptor">
                <property name="authenticationManager">
                        <ref local="authenticationManager"/>
                </property>
                <property name="accessDecisionManager">
                        <ref local="accessDecisionManager"/>
                </property>
                <property name="objectDefinitionSource"><!-- 把URL和可访问的用户组对应起来 -->
                        <value>
                                CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON<!-- 把URL全部转化为小写 -->
                                PATTERN_TYPE_APACHE_ANT<!-- 以ANT的形式来配置路径 -->
                                /ok.html=ROLE_USER
                        </value>
                </property>
        </bean>
```

光这样配置还是不够的，因为当授权失败的时候会抛出异常的，我们应该配置一个异常过滤器来捕获它，exceptionTranslationFilter它是用来捕获异常的，看一下配置吧：

Java代码

```
<bean id="exceptionTranslationFilter" class="org.acegisecurity.ui.ExceptionTranslationFilter">
      <property name="authenticationEntryPoint"><ref local="authenticationProcessingFilterEntryPoint"/></pr
operty>
      <property name="accessDeniedHandler">
        <bean class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
                <property name="errorPage" value="/failure.html"/><!-- 发生异常转向的网页 -->
        </bean>
      </property>
   </bean>
   <bean id="authenticationProcessingFilterEntryPoint" class="org.acegisecurity.ui.webapp.AuthenticationPro
```

```
cessingFilterEntryPoint">
                <property name="loginFormUrl"><value>/Login.html</value></property><!-- 得到表单的信息 -->

        <property name="forceHttps"><value>false</value></property><!-- 不用https -->
    </bean>
```

这样就OK了
最后说一下对类中方法的保护:
首先写一个类并在spring中配置好:

Java代码

```
package org.li.acegi;

public class TestAcegi
{
        public void Role()
        {
                System.out.println("javafish");
        }
}
<bean id="testAcegi" class="org.li.acegi.TestAcegi"/>
```

然看写个servlet访问一下它

Java代码

```
package org.li.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.li.acegi.TestAcegi;
import org.springframework.context.ApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;

public class TestServlet extends HttpServlet
{
        private static final long serialVersionUID = -5610016980827214773L;

        public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
                        throws ServletException, IOException
        {
                response.setContentType("text/html;charset=GBK");
                PrintWriter out = response.getWriter();
                ApplicationContext ctx =
            WebApplicationContextUtils.getRequiredWebApplicationContext(request.getSession().getServletCont
ext());
                TestAcegi test = (TestAcegi)ctx.getBean("testAcegi");
                test.Role();//访问TestAcegi类的Role方法
                out.println("调用成功");
        }

        public void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException
        {
                doGet(request,response);
        }

}
```

准备工作做好了，开始配置Acegi
先在Spring里给Acegi做个代理：

Java代码

```
<bean id="autoProxyCreator" class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">
                <property name="beanNames">
                        <list>
                                <value>testAcegi</value><!-- 要代理的Bean的id -->
                        </list>
                </property>
                <property name="interceptorNames">
                        <list>
                                <value>methodSecurityInterceptor</value><!-- 代理为... -->
                        </list>
                </property>
        </bean>
```

里面的methodSecurityInterceptor呢配置为：

Java代码

```
<bean id="methodSecurityInterceptor" class="org.acegisecurity.intercept.method.aopalliance.MethodSecurityInt
erceptor">
                <property name="authenticationManager">
                        <ref bean="authenticationManager"/>
```

```
                </property>
                <property name="accessDecisionManager">
                        <ref bean="accessDecisionManager"/>
                </property>
                <property name="objectDefinitionSource"><!-- 对代理的类的方法开始配置权限 -->
                        <value>org.li.acegi.TestAcegi.Role=ROLE_USER</value>
                </property>
        </bean>
```

这样当直接访问http://localhost:8080/AcegiWeb/servlet/TestServlet的时候会发现不可访问，控件台也不输出"javafish"，当输入正确的用户名和密码之后便可以访问.
这样它就对类的方法调用起了保护的作用，这一点可以把Acegi应用到DWR上效果是很理想的.
对于Acegi有很多的过滤器不用全写在web.xml里，acegi提供了一个特殊的过滤器我们可以写成这样，在Web.xml里:

Java代码

```
<filter>
                <filter-name>Acegi</filter-name>
                <filter-class>org.acegisecurity.util.FilterToBeanProxy</filter-class>
                <init-param>
                        <param-name>targetClass</param-name>
                        <param-value>org.acegisecurity.util.FilterChainProxy</param-value>
                </init-param>
        </filter>
        <filter-mapping>
                <filter-name>Acegi</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>
        <context-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>
                        /WEB-INF/applicationContext.xml
                </param-value>
        </context-param>
        <listener>
                <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>

        <listener>
                <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
        </listener>
        <listener>
        <listener-class>org.acegisecurity.ui.session.HttpSessionEventPublisher</listener-class>
    </listener>
  <servlet>
```

```
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>org.li.servlet.TestServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>TestServlet</servlet-name>
    <url-pattern>/servlet/TestServlet</url-pattern>
  </servlet-mapping>
```

在Spring的配置文件里:

Java代码

```
<bean id="chainProxy" class="org.acegisecurity.util.FilterChainProxy">
                <property name="filterInvocationDefinitionSource">
                        <value>
                                CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                    PATTERN_TYPE_APACHE_ANT
            /**=HttpSessionContextIntegrationFilter,authenticationProcessingFilter,BasicProcessingFilter,anonymousProcessingFilter,exceptionTranslationFilter,filterSecurityInterceptor
                        </value>
                </property>
        </bean>
```

返回顶楼

lighter

等级:

发表时间：2007-01-08

这一排论坛的acegi讨论挺多的,表现最活跃的leondu同学就在做翻译的工作.
这一篇文章挺好的,不要让它沉下去,投一个良好..

文章: 776
积分: 1235
来自: 广州
我现在离线

返回顶楼 回帖地址 2 0 请登录后投票

andyandyandy
等级:

发表时间：2007-01-09

好文，授权的地方能讲得再细些就更好了

文章: 57
积分: 125
来自: ...
我现在离线

返回顶楼 回帖地址 1 0 请登录后投票

jamesby
等级:

发表时间：2007-01-09

正在研究这东西，感谢楼主!

文章: 484
积分: 845
我现在离线

返回顶楼 回帖地址 1 0 请登录后投票

stamen

等级: 

文章: 161

积分: 440

发表时间：2007-01-10

个人觉得Acegi是Spring最好、最适用的扩展框架了。

返回顶楼 回帖地址 0 0 请登录后投票

rojazz1999

等级: 初级会员

文章: 49

积分: 40

来自: 杭州

发表时间：2007-01-10

确实是好东西啊 收藏!

返回顶楼 回帖地址 0 0 请登录后投票

magice

等级: 

发表时间：2007-01-10

acegi的鉴权做的很好，但是授权方面就很薄弱了，自己做系统如果用acegi的话，那么授权方面还得好好得设计

文章: 130

积分: 288

我现在离线

返回顶楼 回帖地址 1 1 请登录后投票

daquan198163

等级:

文章: 1862

积分: 1491

来自: 吉林->北京->上海

我现在离线

发表时间：2007-01-10

请教几个问题:

1、密码加密如何处理;

2、select username,authority from authorities where username=?取出的authority是什么，应该是role吧;

3、如果像“/ok.html=ROLE_USER”这样配置，当系统运行期间，role的增删改如何处理;

返回顶楼 回帖地址 0 1 请登录后投票

jamesby

等级:

文章: 484

发表时间：2007-01-10

to daquan198163:

我也想知道你的第3个问题，我想应该从objectDifinitonSource入手吧!

积分: 845

我现在离线

返回顶楼 回帖地址 0 0 请登录后投票

zt_smile

等级: 初级会员

发表时间：2007-01-29

这是我看到ACEGI中写的最通俗易懂的文章，谢谢楼主。

文章: 3

积分: 36

来自: 广州

我现在离线

返回顶楼 回帖地址 0 0 请登录后投票