

实战Acegi：使用Acegi作为基于Spring框架的WEB应用的安全框架

作者：[一餐三碗](#)

对于任何一个完整的应用系统，完善的认证和授权机制是必不可少的。在基于SpringFramework的WEB应用中，我们可以使用Acegi作为安全架构的实现。本文将介绍如何在基于Spring构架的Web应用中使用Acegi，并且详细介绍如何配置和扩展Acegi框架以适应实际需要。

1. 概述

Acegi是一个专门为SpringFramework应用提供安全机制的开放源代码项目，全称为Acegi Security System for Spring，当前版本为0.8.3。它使用了Spring的方式提供了安全和认证安全服务，包括使用Bean Context，拦截器和面向接口的编程方式。通过精心配置Acegi安全系统能够轻松地适用于复杂的安全需求。它既能应用于WEB应用也能应用于非WEB应用。在 本文的示例程序里，我将演示如何将Acegi应用于WEB应用程序。通过这个例子详细介绍如何配置Acegi的各个组件，同时介绍如何扩展Acegi使其能够从数据库中读取配置信息。

2. 例子说明

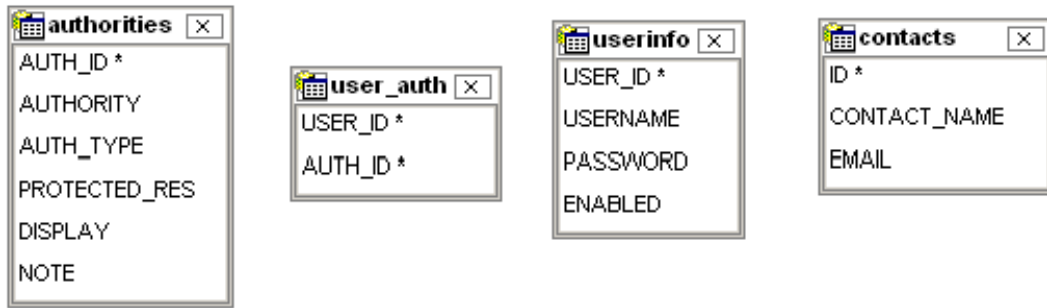
本文的例子是一个联系人管理程序，使用SpringFramework 1.2.4 和 Acegi0.8.3，数据库采用Mysql。程序的目录结构如下：

```
acegi-sample
  contactadd.jsp//增加联系人页面
  contactedit.jsp//编辑联系人页面
  contactlist.jsp//联系人列表页面
  contactmainterance.jsp//联系人操作页面
  index.jsp//主页面
  login.jsp//登录页面
  logoff.jsp//登出页面

  WEB-INF
    web.xml
    applicationContext-basic.xml
    applicationContext-security-acegi.xml
    log4j.properties

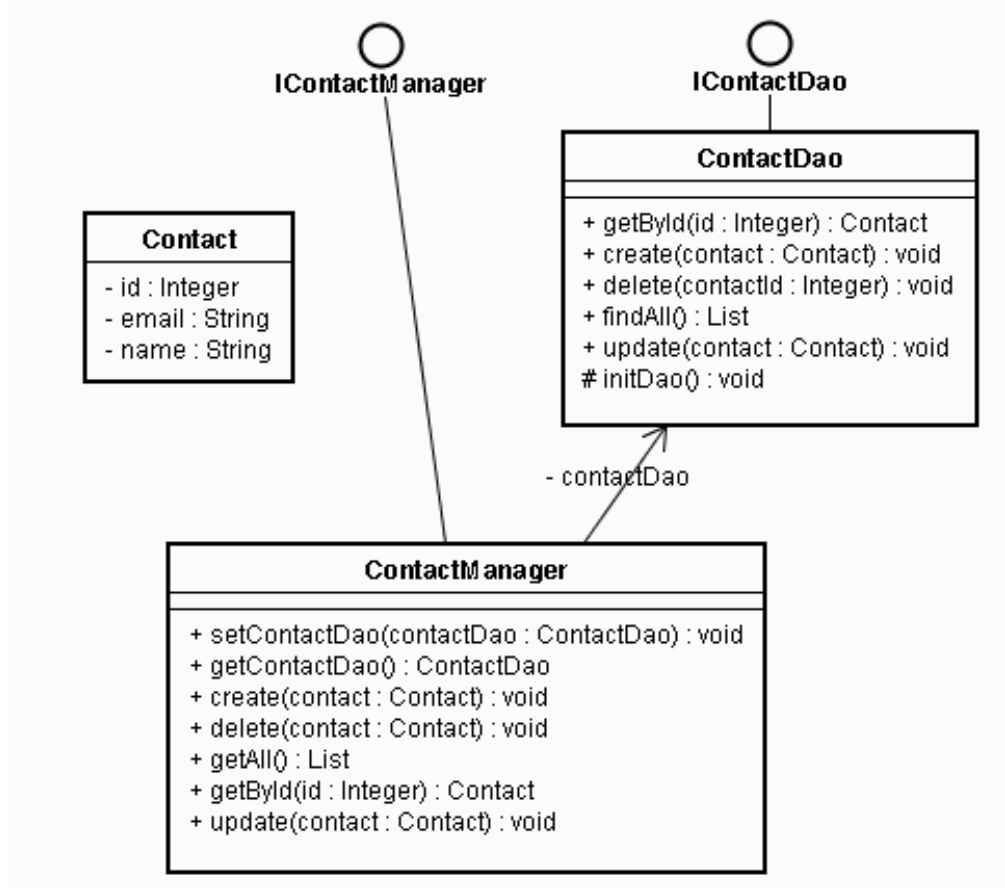
    src
      sample//java代码目录
    classes
    lib//依赖包目录
  db//建表脚本目录
```

程序的ER图如下：



说明：authorities用来存放用户权限配置信息，其中，AUTHORITY字段存放以“AUTH_”开头的权限标志串。AUTH_TYPE存放权限类型。PROTECTED_RES字段存放受保护的资源字符串。DISPLAY字段存放权限显示名称。userinfo用来存放用户信息。authorities和userinfo是多对多关联，使用user_auth作为关联表。contacts存放用户信息。在这个例子中，为了使程序简单明了，我们没有引入一般情况下会有的用户组或者角色的概念，不过在后面的内容里我们将看到对现有的安全模型进行扩展是一件很容易的事。

程序的静态类图如下：



3.配置文件说明

接下来，我们将进入本文的重要内容，开始对Acegi应用程序所牵涉到的配置文件进行一一说明。

3.1 web.xml

首先声明SpringFramework的配置文件列表。为了便于管理，将业务方法相关的配置文件和Acegi安全配置相关的配置文件分开。

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
/WEB-INF/applicationContext-basic.xml
/WEB-INF/applicationContext-security-acegi.xml
</param-value>
</context-param>
```

其次声明Acegi过滤器。

```
<!--Acegi Filter Chain Proxy -->
<filter>
<filter-name>Acegi Filter Chain Proxy</filter-name>
<filter-class>net.sf.acegisecurity.util.FilterToBeanProxy</filter-class>
<init-param>
<param-name>targetClass</param-name>
<param-value>net.sf.acegisecurity.util.FilterChainProxy</param-value>
</init-param>
</filter>

<filter-mapping>
<filter-name>Acegi Filter Chain Proxy</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

说明：Acegi对WEB应用的支持主要是依靠servlet 过滤器（filter）来实现的。每一个http request都将被这些过滤器层层拦截 并进行安全处理（包括认证和授权）。针对不同的安全处理，Acegi提供了不同的过滤器。过滤器的配置信息位于web.xml，但是我们又希望把Acegi的过滤器配置信息放在SpringFramework的配置文​​件里（applicationContext-security-acegi.xml），从而实现对这些过滤器的“控制反转”。解决这个问题的方法是采用Acegi提供的FilterToBeanProxy。FilterToBeanProxy顾名思义就是对Acegi过滤器Bean的代理，它的主要功能就是将http请求依次分派给对应的过滤器Bean。

3.2 applicationContext-security-acegi.xml

applicationContext-security-acegi.xml主要包括认证相关配置信息和WEB资源授权配置信息。首先是声明过滤器序列。

```
<bean id="filterChainProxy" class="net.sf.acegisecurity.util.FilterChainProxy">
  <property name="filterInvocationDefinitionSource">
    <value>
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
      PATTERN_TYPE_APACHE_ANT
      /**=httpSessionContextIntegrationFilter,authenticationProcessingFilter,anonymousProcessingFilter,securityEnforcementFilter
    </value>
  </property>
</bean>
```

各个过滤器的作用如下：

httpSessionContextIntegrationFilter	根据session中存放的信息组装ContextHolder。ContextHolder主要用于存放SecureContext，包括用户的权限信息
authenticationProcessingFilter	处理认证请求（通常是一个登录页面的表单请求）

anonymousProcessingFilter	匿名用户处理。如果用户尚未登录，将生成一个匿名用户的Authentication存放到ContextHolder中
securityEnforcementFilter	强制安全验证过滤器。验证所请求的url是否在用户的权限范围内。

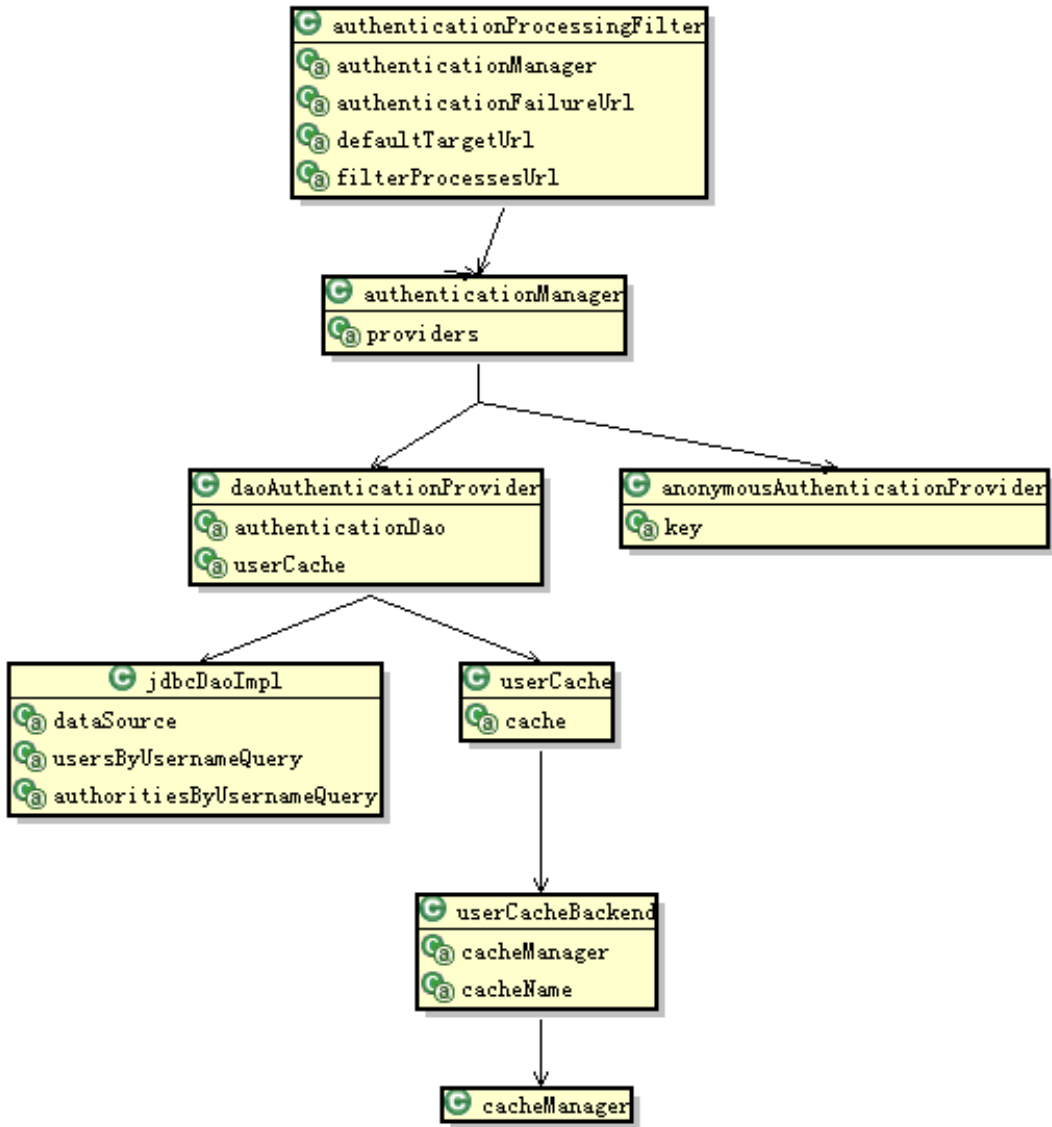
3.2.1 HttpSessionContextIntegrationFilter相关配置信息

```
<bean id="httpSessionContextIntegrationFilter" class="net.sf.acegisecurity.context.HttpSessionContextIntegrationFilter">
<property name="context"><value>net.sf.acegisecurity.context.security.SecureContextImpl</value></property>
</bean>
```

说明：context属性指定context的实现类。

3.2.2 authenticationProcessingFilter相关配置信息

authenticationProcessingFilter的配置比较复杂，我们通过下图来看一下：



authenticationProcessingFilter配置说明：

authenticationManager	认证管理器
authenticationFailureUrl	认证失败后，重定向的url
defaultTargetUrl	认证成功后，重定向的url
filterProcessesUrl	该过滤器拦截的url，通常是/j_acegi_security_check，和登录页面（login.jsp）的登录表单的action相同

authenticationManager（认证管理器）用于管理AuthenticationProvider（认证提供者）。它的作用是使你能够通过多个不同的认证管理源来对用户进行认证。认证管理器将依次调用认证提供者的认证方法，直到认证通过。本程序使用两种认证提供者。authenticationManager配置说明：

daoAuthenticationProvider	基于数据库的认证提供者。
anonymousAuthenticationProvider	用于认证匿名用户。

daoAuthenticationProvider主要功能是从数据库取出用户名和密码，判断登录信息是否正确，如果是，则取出用户权限等用户信息，并且存放到cache中，以便以后再次使用。具体流程可以参考：DaoAuthenticationProvider的authenticate方法。daoAuthenticationProvider配置说明：

authenticationDao	认证数据访问对象，用于获取用户信息，包括：用户名，用户密码，用户状态和用户权限。
userCache	用户信息cache实现bean

jdbcDaoImpl是认证数据访问对象，它能够从默认的数据库结构中获取用户信息，由于Acegi默认的数据库结构和本程序的不同，因此需要修改jdbcDaoImpl的默认sql。注意，采用这种方式能够使Acegi很好的兼容旧的应用程序，因为它对底层的数据结构并没有强制要求。jdbcDaoImpl配置说明：

dataSource	数据源bean
usersByUsernameQuery	用户信息查询sql
authoritiesByUsernameQuery	用户权限查询sql

userCache用于定义用户信息cache功能的提供者。userCache配置说明：

cache	定义ehcache工厂 bean
-------	------------------

本程序采用ehcache作为cache实现。由于认证管理器在每次对http请求进行认证之前都会查找用户信息，通过使用cache就可以避免每次都重复访问数据库。

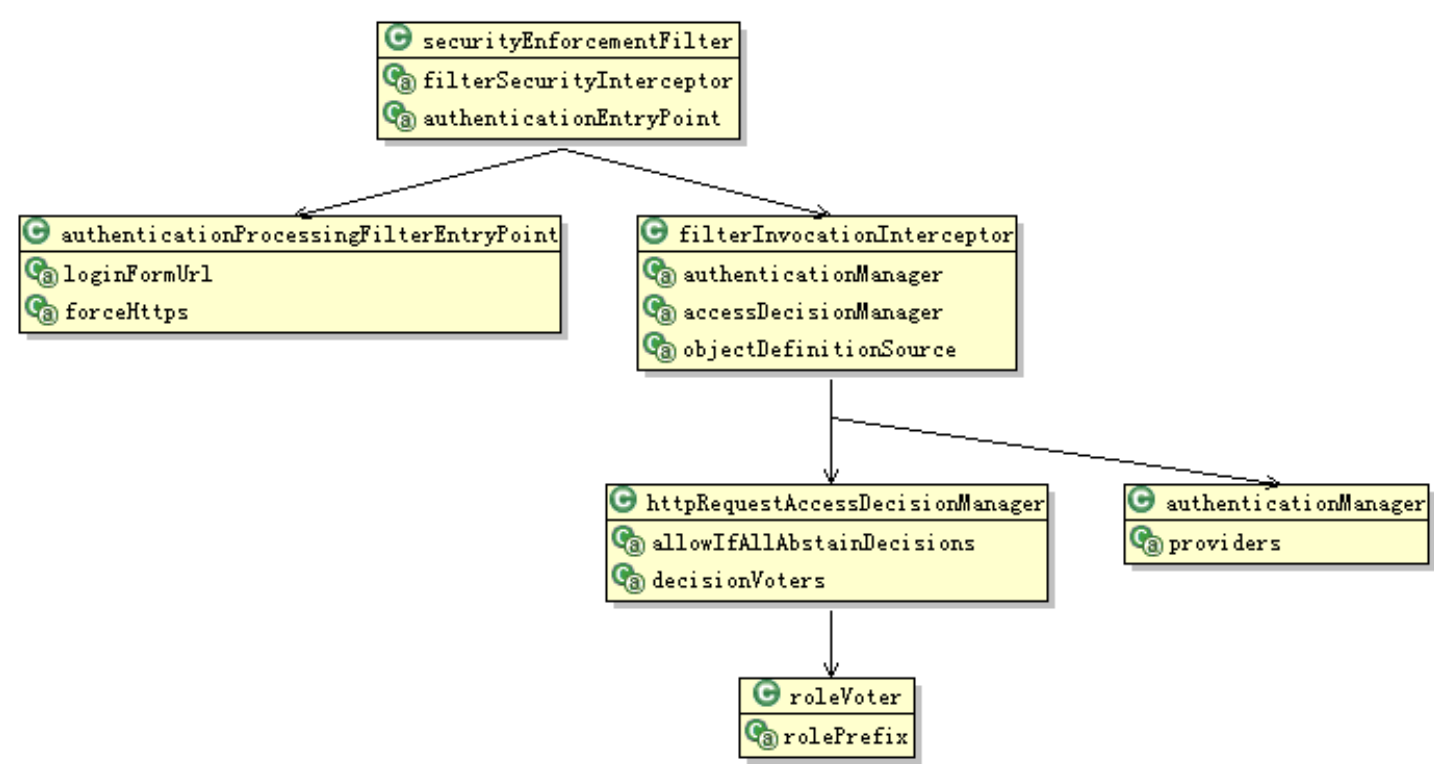
3.2.3 anonymousProcessingFilter相关配置信息：

```
<bean id="anonymousProcessingFilter" class="net.sf.acegisecurity.providers.anonymous.AnonymousProcessingFilter">
<property name="key"><value>foobar</value></property>
<property name="userAttribute"><value>anonymousUser,AUTH_ANONYMOUS</value></property>
</bean>
```

说明：anonymousProcessingFilter的作用是判断ContextHolder中是否有Authentication对象，如果没有就创建一个Authentication对象，其中包含的用户名是anonymousUser，用户权限是AUTH_ANONYMOUS。这使得没有登录的匿名用户能够自动的获得匿名的用户名和权限。

3.2.4 securityEnforcementFilter相关配置信息

securityEnforcementFilter的配置比较复杂，我们通过下图来看一下：



securityEnforcementFilter配置说明：

filterSecurityInterceptor	实现对URL资源进行授权访问。
authenticationEntryPoint	配置登录界面信息。

securityEnforcementFilter的作用主要是将http请求转发给filterSecurityInterceptor，由filterSecurityInterceptor来对HTTP请求的合法性进行判断。

filterInvocationInterceptor配置说明：

authenticationManager	认证管理器
accessDecisionManager	投票通过策略管理器
objectDefinitionSource	URL的权限配置信息。用于指定不同的URL资源对应的权限。配置如下： /**/.jpg=AUTH_ANONYMOUS,AUTH_USER /**/.gif=AUTH_ANONYMOUS,AUTH_USER /**/.png=AUTH_ANONYMOUS,AUTH_USER /login.jsp*=AUTH_ANONYMOUS,AUTH_USER /**=AUTH_USER 以上配置指定AUTH_ANONYMOUS权限的用户（即匿名用户）只可以访问图片资源和登录页面，AUTH_USER权限的用户可以访问全部WEB资源。

accessDecisionManager（访问决策管理器）首先通过authenticationManager判断用户是否通过认证（即是否已经登录），然后根据objectDefinitionSource的配置信息调用accessDecisionManager对用户权限进行投票。

httpRequestAccessDecisionManager配置说明：

allowIfAllAbstainDecisions	设定是否允许：“没人反对就通过”的投票策略
decisionVoters	投票者

httpRequestAccessDecisionManager（投票通过策略管理器）用于管理投票通过策略。Acegi提供三种投票通过策略的实现：AffirmativeBased（至少一个投票者同意方可通过），ConsensusBased（多数投票者同意方可通过），UnanimousBased（所有投票者同意方可通过）。本程序采用AffirmativeBased策略，并且禁止“没人反对就通过”的投票策略。

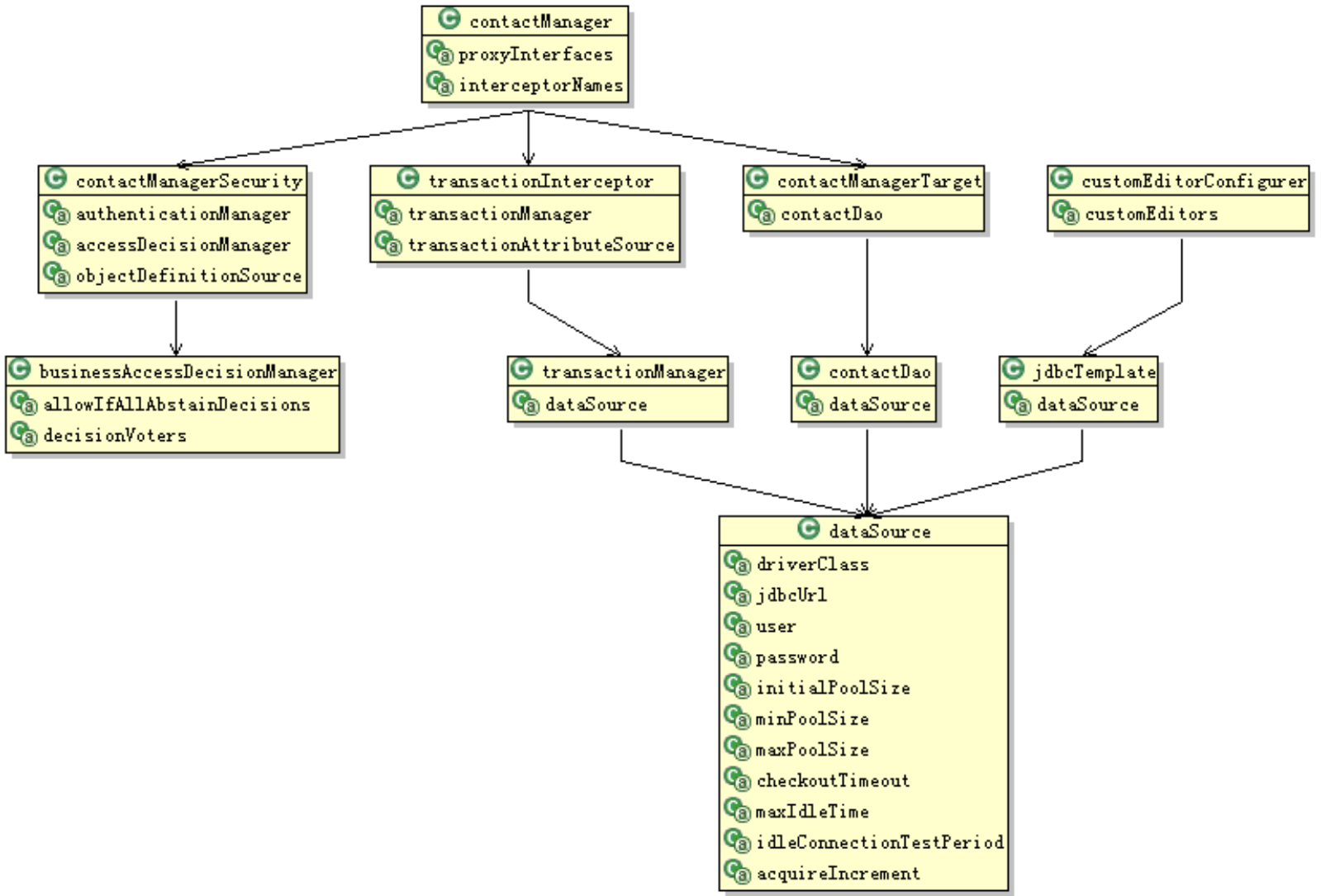
roleVoter配置说明：

rolePrefix	该投票者支持的权限前缀，默认是“ROLE_”，本程序所有的权限字符串均以“AUTH_”开头，故设为“AUTH_”
------------	--

通过设定rolePrefix可以指定roleVoter所支持的权限范围。

3.3 applicationContext-basic.xml

applicationContext-basic.xml主要包括数据访问对象，业务方法，业务方法安全管理拦截器的配置信息。先看一下总揽图：



我将主要讲解业务方法安全管理拦截器（MethodSecurityInterceptor）的相关配置，其它的配置就不再赘述了，请参考相关文档。

3.3.1 contactManager相关配置信息


```
<bean id="contactManager" class="org.springframework.aop.framework.ProxyFactoryBean">
<property name="proxyInterfaces"><value>sample.service.IContactManager</value></property>
<property name="interceptorNames">
<list>
<idref local="transactionInterceptor"/>
<idref local="contactManagerSecurity"/>
<idref local="contactManagerTarget"/>
</list>
</property>
</bean>
```

contactManager的实现类是ProxyFactoryBean（代理工厂），它使用Spring AOP技术拦截代理接口里的方法，并依次执行拦截器列表里的拦截器对应的操作。contactManager有事务拦截器和业务方法安全拦截器。

3.3.2 contactManagerSecurity相关配置信息

authenticationManager	认证管理器
accessDecisionManager	投票通过策略管理器，和filterInvocationInterceptor采用相同的策略管理器，在实际项目中，如果有需要可以采用不同的策略管理器。
objectDefinitionSource	业务方法的权限配置信息。用于指定不同的业务方法资源对应的权限。

注意：contactManagerSecurity的实现类是MethodSecurityInterceptor，缺省情况下MethodSecurityInterceptor的objectDefinitionSource属性是通过net.sf.acegisecurity.intercept.method.MethodDefinitionSourceEditor来设置的。MethodDefinitionSourceEditor只支持属性配置文件的格式（同filterInvocationInterceptor的objectDefinitionSource属性），而实际情况中，由于业务方法比较多，显然配置信息存放在数据库中比较好。因此，我们通过注册一个CustomEditorConfigurer来修改MethodDefinitionSource类型属 性的对应读取类。配置如下：

```
<bean id="customEditorConfigurer" class="org.springframework.beans.factory.config.CustomEditorConfigurer">
<property name="customEditors">
<map>
<entry key="net.sf.acegisecurity.intercept.method.MethodDefinitionSource">
<bean class="sample.util.DataSourceMethodDefinitionSourceEditor">
<property name="jdbcTemplate"> <ref bean="jdbcTemplate"/> </property>
</bean>
</entry>
</map>
</property>
</bean>
```

DataSourceMethodDefinitionSourceEditor根据以下SQL读取业务方法安全配置信息：

```
select authority,PROTECTED_RES from authorities where AUTH_TYPE='FUNCTION' and authority like
'AUTH_FUNC_ContactManager%'
```

权限表authorities内容如下：

AUTH_ID	AUTHORITY	AUTH_TYPE	PROTECTED_RES
1	AUTH_FUNC_ContactManager.create	FUNCTION	sample.service.IContactManager.create
2	AUTH_FUNC_ContactManager.delete	FUNCTION	sample.service.IContactManager.delete
3	AUTH_FUNC_ContactManager.getAll	FUNCTION	sample.service.IContactManager.getAll
4	AUTH_FUNC_ContactManager.getByld	FUNCTION	sample.service.IContactManager.getByld
5	AUTH_FUNC_ContactManager.update	FUNCTION	sample.service.IContactManager.update
0	AUTH_USER	USER	USER

4.程序演示

首先，配置两个用户：root，readonly。具有的权限如下图：

USERNAME	AUTHORITY	PROTECTED_RES
root	AUTH_USER	USER
root	AUTH_FUNC_ContactManager.create	sample.service.IContactManager.create
root	AUTH_FUNC_ContactManager.delete	sample.service.IContactManager.delete
root	AUTH_FUNC_ContactManager.getAll	sample.service.IContactManager.getAll
root	AUTH_FUNC_ContactManager.getByld	sample.service.IContactManager.getByld
root	AUTH_FUNC_ContactManager.update	sample.service.IContactManager.update
readonly	AUTH_USER	USER
readonly	AUTH_FUNC_ContactManager.getAll	sample.service.IContactManager.getAll
readonly	AUTH_FUNC_ContactManager.getByld	sample.service.IContactManager.getByld

登录界面：



Login

可用的用户：

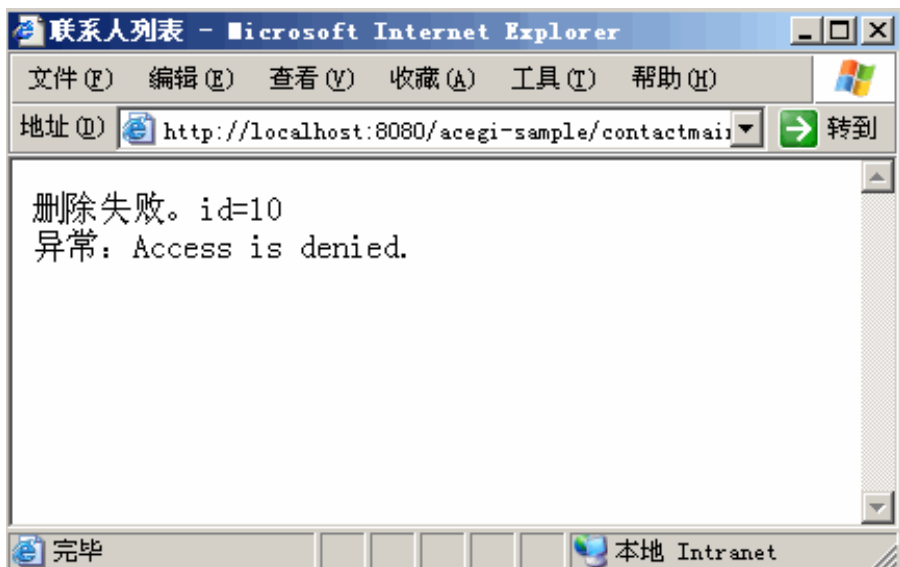
username **root**, password **root**（具有全部权限）

username **readonly**, password **readonly**（只有查看列表权限权限）

User:

Password:

以readonly用户登录后，执行删除联系人操作，显示操作被拒绝：



5.总结

在本文的示例程序中我们只对业务对象（ContactManager）进行安全保护，对业务领域对象（Contact）的访问并没有作限制，这是由于在Acegi框架中采用ACL（访问控制列表）技术实现这个功能，这使得一旦我们的业务领域对象数量很多的话，效率将变得很低，因此我们将对业务领域对象访问控制的代码放在业务对象的业务方法中。

将业务无关的代码从业务代码中剥离，使业务代码更干净，系统结构更合理是每个开发人员的梦想。随着AOP技术的日渐流行和日益发展，这个梦想已经离我们不远了。本文中的例子通过结合使用SpringFramework和Acegi两种开源框架，实现了将安全认证和授权代码和事务代码从业务代码中分离。

6.代码说明

本文相关代码acegi-sample.war.

Mysql的建表脚本在db目录下.

为了减小体积，已经将WEB-INF\lib下的依赖包删除，请自行下载以下包，并拷贝至WEB-INF\lib下：

spring-1.2.4.jar

acegi-security-0.8.3.jar

aopalliance-1.0.jar

c3p0-0.9.0.jar

commons-logging-1.0.4.jar

ehcache-1.1.jar

log4j-1.2.8.jar

mysql-connector-java-3.1.10-bin.jar

oro-2.0.8.jar

7.参考资料

Acegi Security System for Spring Reference Documentation.

Acegi Security System contacts sample。本文中的示例程序参考了Acegi的contacts演示程序。

关于作者

[一餐三碗](#) 系统架构师 从事电信行业软件开发多年，目前专注于网管软件的开发。