配置过程 （springmvc 和 struts）

Spring

Springmvc

（配置 shiro）首先是在web.xml 中

<filter>

<filter-name>shiroFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

<init-param>

<param-name>targetFilterLifecycle</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>shiroFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

然后在applicationContext.xml 文件

配置cachemanager

<bean id="cacheManager" class="org.apache.shiro.cache.ehcache.EhCacheManager">

<!-- Set a net.sf.ehcache.CacheManager instance here if you already have one. If not, a new one

will be creaed with a default config:

<property name="cacheManager" ref="ehCacheManager"/> -->

<!-If you don't have a pre-built net.sf.ehcache.CacheManager instance to inject, but you want

a specific Ehcache configuration to be used, specify that here. If you don't, a default

will be used.:

<property name="cacheManagerConfigFile" value="classpath:some/path/to/ehcache.xml"/> -->

</bean>

Eacahe jar包在hibernate中可以找到

配置realm （实现了realm的接口）

<bean id="jdbcRealm" class="org.apache.shiro.samples.spring.realm.SaltAwareJdbcRealm">

<property name="name" value="jdbcRealm"/>

<property name="dataSource" ref="dataSource"/>

<property name="credentialsMatcher">

<!-- The 'bootstrapDataPopulator' Sha256 hashes the password

(using the username as the salt) then base64 encodes it: -->

<bean class="org.apache.shiro.authc.credential.HashedCredentialsMatcher">

<property name="hashAlgorithmName" value="SHA-256"/>

<!-- true means hex encoded, false means base64 encoded -->

<property name="storedCredentialsHexEncoded" value="false"/>

</bean>

</property>

</bean>

配置lifecycleBeanPostProcessor

<bean id="lifecycleBeanPostProcessor" class="org.apache.shiro.spring.LifecycleBeanPostProcessor"/>

必须在开启spring 的注解过程中才可以使用

<bean class="org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator"

depends-on="lifecycleBeanPostProcessor"/>

<bean class="org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor">

<property name="securityManager" ref="securityManager"/>

</bean>

<bean id="secureRemoteInvocationExecutor" class="org.apache.shiro.spring.remoting.SecureRemoteInvocationExecutor">

<property name="securityManager" ref="securityManager"/>

</bean>

配置shiroFilter (id必须和配置文件中的)

<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">

<property name="securityManager" ref="securityManager"/>

<property name="loginUrl" value="/s/login"/>

<property name="successUrl" value="/s/index"/>

<property name="unauthorizedUrl" value="/s/unauthorized"/>

资源是否可以被匿名访问

Authc 必须是在认证之后才可以访问

<property name="filterChainDefinitions">

<value>

/favicon.ico = anon

/logo.png = anon

/shiro.css = anon

/s/login = anon

# allow WebStart to pull the jars for the swing app:

/\*.jar = anon

# everything else requires authentication:

/\*\* = authc

</value>

</property>

</bean>

Realm

LifecycleBeanPost

shiroFilter （和web.xml一致）

Shiro 的认证过程

Url拦截 有优先级，先匹配的可以 覆盖后面的配置

（outline 查看大纲）

认证过程

使用realm获取安全数据

1 获取当前的subject 调用SecurityUtils.getSubject();

2 测试当前的用户是否已经被认证