# Shiro框架开发步骤

使用shiro框架进行项目的认证和授权操作

### 第一步：导入shiro的jar到项目中

### 第二步：在web.xml中配置一个过滤器代理对象，在项目启动时到spring工厂中加载一个和当前过滤器name同名的bean对象

web.xml

<!-- 加入spring框架提供的过滤器代理对象 -->

<filter>

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

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

</filter>

<filter-mapping>

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

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

</filter-mapping>

### 第三步：在spring配置文件中配置一个名称为shiroFilter的bean

<!-- 配置shiro的bean， 使用当前工厂bean对象创建过滤器用于进行权限控制-->

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

<!-- 注入安全管理器对象 -->

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

<!-- 登录页面url地址 -->

<property name="loginUrl" value="/login.jsp"></property>

<!-- 成功页面 -->

<property name="successUrl" value="/index.jsp"></property>

<!-- 权限不足提示页面 -->

<property name="unauthorizedUrl" value="/unauthorized.jsp"></property>

<!-- 基于url拦截，使用过滤器进行拦截 -->

<property name="filterChainDefinitions">

<value>

/css/\*\* = anon

/images/\*\* = anon

/js/\*\* = anon

/validatecode.jsp\* = anon

/login.jsp = anon

/userAction\_login.action = anon

/page\_base\_staff.action = perms["staff"]

/\*\* = authc

</value>

</property>

</bean>

<!-- 配置安全管理器 -->

<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">

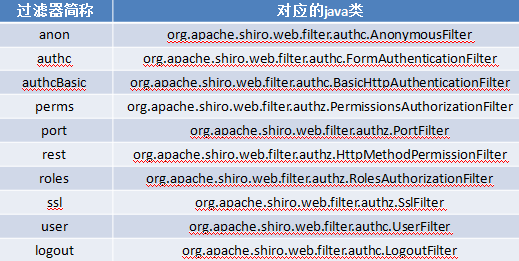
<property name="realm" ref="bosRealm"></property>

</bean>

<!-- 配置自定义的realm -->

<bean id="bosRealm" class="cn.itcast.bos.shiro.BOSRealm"/>

Shiro框架提供的过滤器：



### 第四步：修改UserAction的login登录访问，使用shiro框架提供的方式进行认证操作

//验证码输入成功

//使用shiro提供的方式进行权限认证

//获得当前用户对象，现在状态为“未认证”

Subject subject = SecurityUtils.*getSubject*();

String username = model.getUsername();

String password = model.getPassword();

password = MD5Utils.*md5*(password);

AuthenticationToken token = **new** UsernamePasswordToken(username,password);

**try**{

subject.login(token);//调用安全管理器，安全管理器调用Realm

User user = (User) subject.getPrincipal();

// 登录成功，将user放入session，跳转到系统首页

ServletActionContext.*getRequest*().getSession().setAttribute("loginUser", user);

}**catch** (UnknownAccountException e) {

e.printStackTrace();

//用户名不存在，跳转到登录页面

**this**.addActionError("用户名不存在！");

**return** "login";

}**catch** (IncorrectCredentialsException e) {

// 密码错误，跳转到登录页面

**this**.addActionError("密码错误！");

e.printStackTrace();

**return** "login";

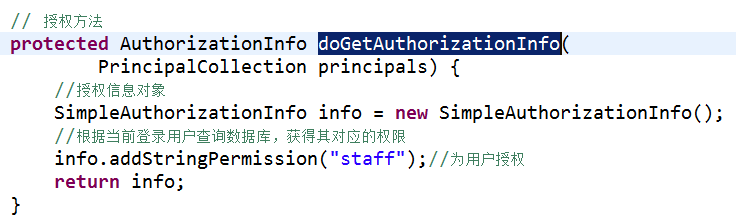
}

**return** "home";

}

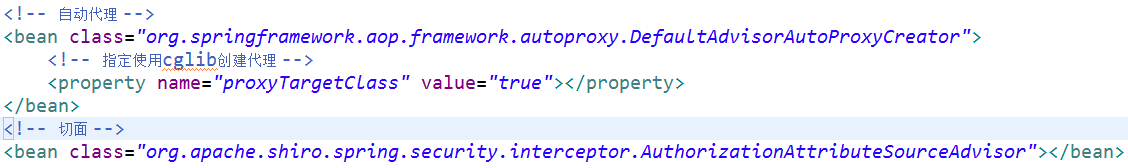
### 第五步：自定义一个Realm，进行认证和授权操作



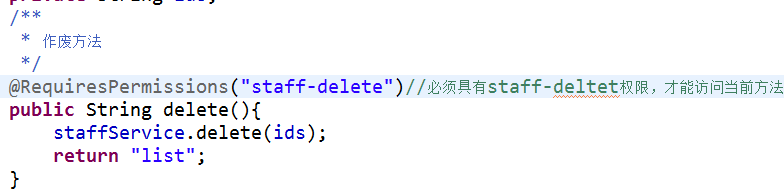


# Shiro提供的注解方式权限控制

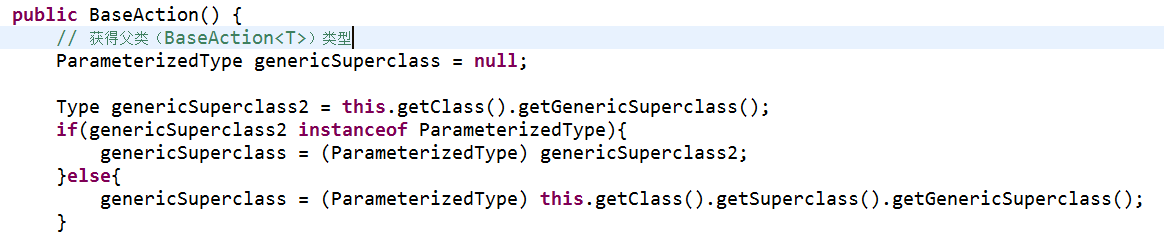
### 第一步：在spring配置文件中配置自动代理和切面



### 第二步：在Action方法上使用注解



### 第三步：修改BaseAction的构造方法



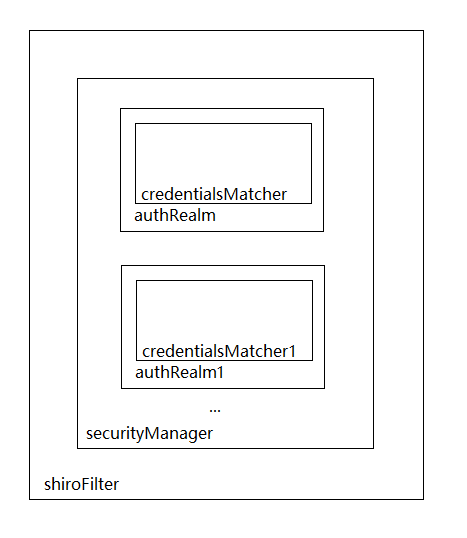
# Shiro多入口登入

概念讲解：

A.subject-->用户对象

B.Token -->登入所需token

C.Realm -->



步骤：

### 第一步：自定义token，参考UsernamePasswordToken

|  |
| --- |
| public class UsernamePasswordToken implements HostAuthenticationToken, RememberMeAuthenticationToken {  private String username;  private char[] password;  private boolean rememberMe;  private String host;  public UsernamePasswordToken() {  this.rememberMe = false;  }  public UsernamePasswordToken(String username, char[] password) {  this(username, (char[])password, false, (String)null);  }  public UsernamePasswordToken(String username, String password) {  this(username, (char[])(password != null ? password.toCharArray() : null), false, (String)null);  }  public UsernamePasswordToken(String username, char[] password, String host) {  this(username, password, false, host);  }  public UsernamePasswordToken(String username, String password, String host) {  this(username, password != null ? password.toCharArray() : null, false, host);  }  public UsernamePasswordToken(String username, char[] password, boolean rememberMe) {  this(username, (char[])password, rememberMe, (String)null);  }  public UsernamePasswordToken(String username, String password, boolean rememberMe) {  this(username, (char[])(password != null ? password.toCharArray() : null), rememberMe, (String)null);  }  public UsernamePasswordToken(String username, char[] password, boolean rememberMe, String host) {  this.rememberMe = false;  this.username = username;  this.password = password;  this.rememberMe = rememberMe;  this.host = host;  }  public UsernamePasswordToken(String username, String password, boolean rememberMe, String host) {  this(username, password != null ? password.toCharArray() : null, rememberMe, host);  }  public String getUsername() {  return this.username;  }  public void setUsername(String username) {  this.username = username;  }  public char[] getPassword() {  return this.password;  }  public void setPassword(char[] password) {  this.password = password;  }  public Object getPrincipal() {  return this.getUsername();  }  public Object getCredentials() {  return this.getPassword();  }  public String getHost() {  return this.host;  }  public void setHost(String host) {  this.host = host;  }  public boolean isRememberMe() {  return this.rememberMe;  }  public void setRememberMe(boolean rememberMe) {  this.rememberMe = rememberMe;  }  public void clear() {  this.username = null;  this.host = null;  this.rememberMe = false;  if (this.password != null) {  for(int i = 0; i < this.password.length; ++i) {  this.password[i] = 0;  }  this.password = null;  }  }  public String toString() {  StringBuilder sb = new StringBuilder();  sb.append(this.getClass().getName());  sb.append(" - ");  sb.append(this.username);  sb.append(", rememberMe=").append(this.rememberMe);  if (this.host != null) {  sb.append(" (").append(this.host).append(")");  }  return sb.toString();  } } |

### 第二步：自定义Realm，继承AuthorizingRealm,重点重写supports方法

|  |
| --- |
| public class MyRealm extends AuthorizingRealm {  private UserService userService = new UserServiceImpl();  @Override  public boolean supports(AuthenticationToken token) {  /\*Shiro在进行登录验证时候，会检查Realm是否支持该Token，如果不支持跳过当前Realm，继续下一个Realm\*/  return token != null && token.getClass().isAssignableFrom(MobileCodeToken.class);  } //token.getClass()传递进来token的class类  @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {  String username = (String)principals.getPrimaryPrincipal();  SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();  authorizationInfo.setRoles(userService.findRoles(username));  authorizationInfo.setStringPermissions(userService.findPermissions(username));  return authorizationInfo;  }  @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {  String username = (String)token.getPrincipal();  //User user = userService.findByUsername(username);  User user = new User();  user.setUsername(username);  user.setPassword(token.getCredentials().toString());  if(user == null) {  throw new UnknownAccountException();//没找到帐号  }  if(Boolean.*TRUE*.equals(user.getLocked())) {  throw new LockedAccountException(); //帐号锁定  }  //交给AuthenticatingRealm使用CredentialsMatcher进行密码匹配，如果觉得人家的不好可以自定义实现  SimpleAuthenticationInfo authenticationInfo = new SimpleAuthenticationInfo(  user.getUsername(), //用户名  user.getPassword(), //密码 ByteSource.Util.*bytes*(user.getCredentialsSalt()),//salt=username+salt  getName() //realm name  );  return authenticationInfo;  } } |

### 第三步：自定义Matcher匹配器，用来指定加密算法

|  |
| --- |
| public class RetryLimitHashedCredentialsMatcher extends HashedCredentialsMatcher{  private Cache<String, AtomicInteger> passwordRetryCache;  public RetryLimitHashedCredentialsMatcher(CacheManager cacheManager) {  passwordRetryCache = cacheManager.getCache("passwordRetryCache");  }  public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info)  {  String username = (String)token.getPrincipal();  AtomicInteger retryCount = (AtomicInteger)passwordRetryCache.get(username);  if (retryCount == null) {  retryCount = new AtomicInteger(0);  passwordRetryCache.put(username, retryCount);  }  if (retryCount.incrementAndGet() > 5) {  throw new ExcessiveAttemptsException();  }  boolean matches = super.doCredentialsMatch(token, info);  if (matches) {  passwordRetryCache.remove(username);  }  return matches;  } } |

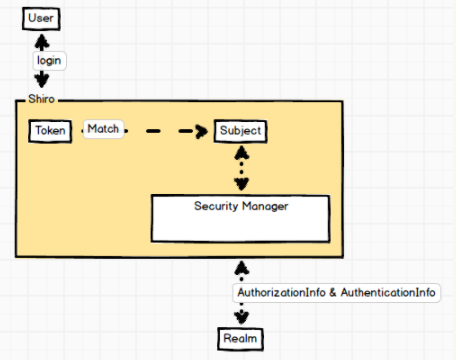
### 第四步：spring-shiro.xml相关配置

|  |
| --- |
| <!-- 凭证匹配器 --> <bean id="customCredentialsMatcher"  class="com.shubo.shiro.credentials.CustomCredentialsMatcher"> </bean> <!-- Realm实现 --> <bean id="userRealm" class="com.shubo.shiro.realm.UserRealm">  <property name="credentialsMatcher" ref="credentialsMatcher" />  <property name="cachingEnabled" value="true" />  <property name="authenticationCachingEnabled" value="false" />  <property name="authenticationCacheName" value="authenticationCache" />  <property name="authorizationCachingEnabled" value="false" />  <property name="authorizationCacheName" value="authorizationCache" /> </bean>  <!-- 安全管理器 --> <bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">  <!-- 可以配置多个Realm，其实会把realms属性赋值给ModularRealmAuthenticator的realms属性 -->  <property name="authenticator" ref="authenticator"></property>  <property name="realms">  <list>  <ref bean="userRealm"/>  <ref bean="mobileCodeRealm"/>  </list>  </property>  <property name="sessionManager" ref="sessionManager" />  <property name="cacheManager" ref="cacheManager" /> </bean> |

# 快速入门

一、架构

要学习如何使用Shiro必须先从它的架构谈起，作为一款安全框架Shiro的设计相当精妙。Shiro的应用不依赖任何容器，它也可以在JavaSE下使用。但是最常用的环境还是JavaEE。下面以用户登录为例：



根据用户名，查询当前用户的角色，将角色名称提供给info。

根据用户名查询，查询当前用户的权限，将权限名称提供给info。