最近一段时间，我一直在将shiro集成到我的一个项目中，用作认证和授权处理。  
            shiro对我来说是个新东西，以下是我学习过的内容：  
            http://shiro.apache.org/authorization.html  
            http://www.cnblogs.com/skyme/archive/2011/09/11/2173760.html  系列  
            http://www.infoq.com/cn/articles/apache-shiro  
            http://kdboy.iteye.com/blog/1103794  
http://www.ibm.com/developerworks/cn/java/j-lo-shiro/  
如果我那个地方没说明白，可以看这些。  
 集成shiro,需要配置web.xml文件，spring的applicationContext.xml配置文件（当然，独立配置一个shiro.xml文件交给spring容器处理也是可以的）。  
web.xml文件中的配置：  
  
<!-- shiro filter的名字是shiroFilter,那么在spring的配置文件中要有一个名字为shiroFilter的bean-->    
   <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>

applicationContext.xml文件中的配置：  
  
<!-- 自定义的Shiro Filter-->    
   <bean id="simplePermFilter" class="frame.security.PermissionsAuthorizationFilter"></bean>    
<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">    
    <property name="securityManager" ref="securityManager" />    
    <property name="loginUrl" value="/login" />    
    <property name="successUrl" value="/user/list" />    
    <property name="unauthorizedUrl" value="/login" />    
    <property name="filters">      
           <map>      
               <entry key="sperm" value-ref="simplePermFilter"/>    
           </map>      
       </property>      
    <property name="filterChainDefinitions">    
        <value>    
            /Flat-UI-master/\*\*=anon    
            /index.jsp\* = anon    
            /test.jsp\*=anon    
            /jsp/\*\* = authc    
            /test/objT.do = sperm    
        </value>    
    </property>    
</bean>    
  
   <bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">    
    <property name="realm" ref="myRealm" />    
</bean>    
   <!-- 配置shiro bean processor-->    
   <bean id="lifecycleBeanPostProcessor" class="org.apache.shiro.spring.LifecycleBeanPostProcessor" />    
    <!--myRealm 继承自AuthorizingRealm-->    
    <bean id="myRealm" class="frame.security.MonitorRealm" ></bean>    
    <bean class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">    
         <property name="staticMethod"    
              value="org.apache.shiro.SecurityUtils.setSecurityManager" />    
         <property name="arguments" ref="securityManager" />    
    </bean>

代码说明：  
shiroFilter 中 loginUrl 为登录页面地址，successUrl 为登录成功页面地址（如果首先访问受保护 URL 登录成功，则跳转到实际访问页面），unauthorizedUrl 认证未通过访问的页面（前面提到的“未经授权页面”）。  
shiroFilter 中 filters 属性，formAuthenticationFilter 配置为基于表单认证的过滤器。  
shiroFilter 中 filterChainDefinitions 属性，anon 表示匿名访问（不需要认证与授权），authc 表示需要认证，perms[SECURITY\_ACCOUNT\_VIEW] 表示用户需要提供值为“SECURITY\_ACCOUNT\_VIEW”Permission 信息。由此可见，连接地址配置为 authc 或 perms[XXX] 表示为受保护资源。  
securityManager 中 realm 属性，配置为我们自己实现的 Realm。关于 Realm，参见前面“Shiro Realm”章节。  
myShiroRealm 为我们自己需要实现的 Realm 类，为了减小数据库压力，添加了缓存机制。  
shiroCacheManager 是 Shiro 对缓存框架 EhCache 的配置。  
MonitorRealm.java 是自定义的realm,读取数据库的用户信息，和授权信息。  
 PermissionsAuthorizationFilter.java 是自定义的过滤器，来实现自己需要的授权过滤方式。

public class MonitorRealm extends AuthorizingRealm{    
       
    @Autowired    
    private DAO dao;<span style="font-family: Consolas, 'Courier New', Courier, mono, serif; font-size: 11.818181991577148px; line-height: 17.27272605895996px; background-color: rgb(248, 248, 248);">//这里自己需要什么就注入什么。  </span>    
   
    public MonitorRealm() {    
        super();    
    }    
       
    /\*\*   
     \* 授权操作，决定那些角色可以使用那些资源   
     \*/    
    @Override    
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection pc) {    
           
        //<span style="font-family: Consolas, 'Courier New', Courier, mono, serif; font-size: 11.818181991577148px; line-height: 17.27272605895996px; background-color: rgb(248, 248, 248);">TODO </span><span style="font-family: Consolas, 'Courier New', Courier, mono, serif; font-size: 11.818181991577148px; line-height: 17.27272605895996px; background-color: rgb(248, 248, 248);">访问授权信息</span>    
        return info;    
    }    
   
    /\*\*   
     \* 认证操作，判断一个请求是否被允许进入系统   
     \*/    
    @Override    
    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authcToken) throws AuthenticationException {    
           
        //<span style="font-family: Consolas, 'Courier New', Courier, mono, serif; font-size: 11.818181991577148px; line-height: 17.27272605895996px;">TODO 用户认证信息  </span>    
            return info;      
    }    
   
}

public class PermissionsAuthorizationFilter extends AuthorizationFilter {    
   
    public boolean isAccessAllowed(ServletRequest request, ServletResponse response, Object mappedValue) throws IOException {    
        <span style="font-family: Consolas, 'Courier New', Courier, mono, serif; font-size: 11.818181991577148px; line-height: 17.27272605895996px; background-color: rgb(248, 248, 248);"> //自定义过滤器逻辑</span>       
        return true;    
    }    
}

配置自定义过滤器的关键是配置文件中 的这几句  
<property name="filters">      
            <map>      
                <entry key="sperm" value-ref="simplePermFilter"/>    
            </map>      
        </property>      
        <property name="filterChainDefinitions">    
            <value>    
            ...       
                /test/objT.do ="sperm"    
            </value>    
        </property>

颜色相同的地方一定要一样，表示用某个过滤器过滤指定路径。因为这个我费了好长时间。  
org.apache.shiro.spring.web.ShiroFilterFactoryBean   的作用是通过spring来初始化shiro的工作环境。如果一个请求进来，shiro的过滤器会先工作，过滤器会调用realm中的授权或认证的方法来获取授权或认证信息。