Shiro的使用

Shiro是什么?

Shiro是一款安全登录验证,用户权限管理的一款适用性强的插件

shiro的执行是怎样的,如何过滤是否登录, 如何过滤动态资源

如何使用Shiro

在javaEE中使用Shiro需要在配置文件中加入一个`.xml文件`在文件中需要配置

1. 引入.xml文件
2. 在xml文件中需要配置安全管理器

```
<bean class="org.apache.shiro.web.mgt.DefaultWebSecurityManager"
id="securityManager">    <property name="realm" ref="crmRealm">
</property></bean>
```

3. 还需要配置Shiro内置的过滤器

```
<bean id="shiroFilter"
 class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">    <!--引用
指定的安全管理器-->
<property name="securityManager" ref="securityManager"/>
    <property name="loginUrl" value="/login.html"/>
    <property name="filterChainDefinitions">
        <!--配置当前过滤的地址    anon是不过滤,authc是过滤-->
        <value>
            /js/**=anon
            /static/**=anon
            /**=authc
        </value>
    </property>

    <property name="filters">        <map>
        <entry key="authc" value-ref="crmTormAuthenticationFilter">
</entry>        </map>
    </property>
</bean>
```

`还需要设置数据源集成授权的数据源`(集成Shiro内部的数据源让他重写)

```java
@Component("crmRealm")
public class crmRealm extends AuthorizingRealm {

    public String getName() { return "crmRealm"; }

    @Autowired
    private IEmployeeService employeeService;

    @Override
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {
        //实现认证器
        //获取数据库里面的内容
        /*将数据库里面的内容封住从令牌中查找*/
        return null;
    }
    @Override
    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {
        //授权器
        //调用认证器
        //页面传递过来的用户名
        String principal = (String) token.getPrincipal();
        //获取数据库的用户名
        Employee employee = employeeService.getLoginName(principal);
        //如果令牌中的用户名和数据库中的用户名一致就验证密码
        if (employee.getName().equals(principal)) {
            String password = employee.getPassword();
            return new SimpleAuthenticationInfo(employee.getName(), password, getName());
        }
        /*if ("admin".equals(principal)) {
            return new SimpleAuthenticationInfo(principal, "555", getName());//用户名,密码,数据源自定义的名称
        }*/
        return null;
    }
}
```

需要设置响应回来的数据做什么这时候需要需要自己创建一个过滤响应器用来响应成功或者失败分别做什么

```java
@Component("crmFormAuthenticationFilter")
public class CRMFormAuthenticationFilter extends FormAuthenticationFilter {
    //成功
    @Override
    protected boolean onLoginSuccess(AuthenticationToken token, Subject subject, ServletRequest request, ServletResponse response) throws Exception {
        response.setContentType("application/json;charset=UTF-8");
        JsonResult json = new JsonResult();
        //将结果集响应给页面然后做出响应
        String jsonString = JSON.toJSONString(json);

        response.getWriter().print(jsonString);
        return false;
    }

    //失败
    @Override
    protected boolean onLoginFailure(AuthenticationToken token, AuthenticationException e, ServletRequest request, ServletResponse response) {
        //因为浏览器响应回去的不是一个json格式需要我们告诉浏览器它是一个json格式的数据
        response.setContentType("application/json;charset=UTF-8");
        JsonResult json = new JsonResult();
        try {
            json.mark("用户名或者密码错误");
            response.getWriter().print(JSON.toJSONString(json));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        return false;
    }
```