1 Shiro

什么是 Shiro

官网：http://shiro.apache.org/

是一款主流的 Java 安全框架，不依赖任何容器，可以运行在 Java SE 和 Java EE 项目中，它的主要作用是对访问系统的用户进行身份认证、授权、会话管理、加密等操作。

Shiro 就是用来解决安全管理的系统化框架。

2 Shiro 核心组件

用户、角色、权限

会给角色赋予权限，给用户赋予角色

1、UsernamePasswordToken，Shiro 用来封装用户登录信息，使用用户的登录信息来创建令牌 Token。

2、SecurityManager，Shiro 的核心部分，负责安全认证和授权。

3、Suject，Shiro 的一个抽象概念，包含了用户信息。

4、Realm，开发者自定义的模块，根据项目的需求，验证和授权的逻辑全部写在 Realm 中。

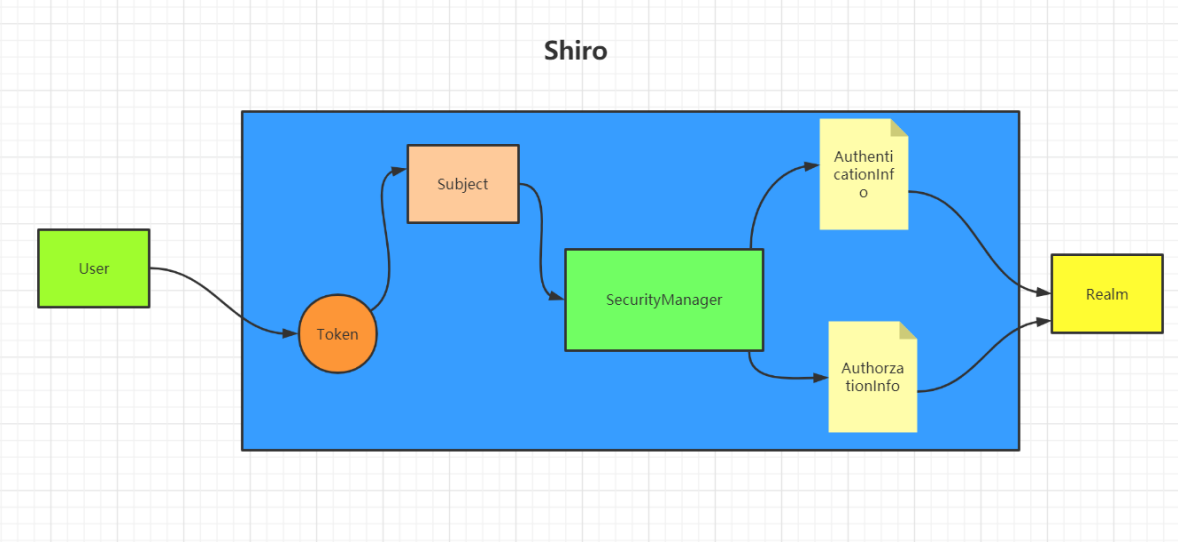
5、AuthenticationInfo，用户的角色信息集合，认证时使用。

6、AuthorzationInfo，角色的权限信息集合，授权时使用。

7、DefaultWebSecurityManager，安全管理器，开发者自定义的 Realm 需要注入到 DefaultWebSecurityManager 进行管理才能生效。

8、ShiroFilterFactoryBean，过滤器工厂，Shiro 的基本运行机制是开发者定制规则，Shiro 去执行，具体的执行操作就是由 ShiroFilterFactoryBean 创建的一个个 Filter 对象来完成。

Shiro 的运行机制如下图所示。



3 Spring Boot 整合 Shiro

1、创建 Spring Boot 应用，集成 Shiro 及相关组件，pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
```

```
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-spring</artifactId>
        <version>1.5.3</version>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-
starter</artifactId>
        <version>3.3.1.tmp</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-
engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

2、自定义 Shiro 过滤器

```
```

```
public class AccoutRealm extends AuthorizingRealm {

    @Autowired
    private AccountService accountService;

    /**
     * 授权
     * @param principalCollection
     * @return
     */
    @Override
    protected AuthorizationInfo
doGetAuthorizationInfo(PrincipalCollection
principalCollection) {
        return null;
    }


    /**
     * 认证
     * @param authenticationToken
     * @return
     * @throws AuthenticationException
     */
    @Override
    protected AuthenticationInfo
doGetAuthenticationInfo(AuthenticationToken
authenticationToken) throws AuthenticationException {
        UsernamePasswordToken token =
(UsernamePasswordToken) authenticationToken;
        Account account =
accountService.findByUsername(token.getUsername());
        if(account != null){
            return new
SimpleAuthenticationInfo(account,account.getPassword(),
getName());
        }
        return null;
```

```
    }
}
```

3、配置类

```
@Configuration
public class ShiroConfig {

    @Bean
    public ShiroFilterFactoryBean
shiroFilterFactoryBean(@Qualifier("securityManager")
DefaultWebSecurityManager securityManager){
        ShiroFilterFactoryBean factoryBean = new
ShiroFilterFactoryBean();

 factoryBean.setSecurityManager(securityManager);
        return factoryBean;
    }



    @Bean
    public DefaultWebSecurityManager
securityManager(@Qualifier("accoutRealm") AccoutRealm
accoutRealm){
        DefaultWebSecurityManager manager = new
DefaultWebSecurityManager();
        manager.setRealm(accoutRealm);
        return manager;
    }

    @Bean
    public AccoutRealm accoutRealm(){
        return new AccoutRealm();
    }
}
```

编写认证和授权规则：

> 认证过滤器

anon：无需认证。

authc：必须认证。

authcBasic：需要通过 HTTPBasic 认证。

user：不一定通过认证，只要曾经被 Shiro 记录即可，比如：记住我。

> 授权过滤器

perms：必须拥有某个权限才能访问。

role：必须拥有某个角色才能访问。

port：请求的端口必须是指定值才可以。

rest：请求必须基于 RESTful，POST、PUT、GET、DELETE。

ssl：必须是安全的 URL 请求，协议 HTTPS。

创建 3 个页面，main.html、manage.html、administrator.html

访问权限如下：

1、必须登录才能访问 main.html

2、当前用户必须拥有 manage 授权才能访问 manage.html

3、当前用户必须拥有 administrator 角色才能访问 administrator.html

> Shiro 整合 Thymeleaf

1、pom.xml 引入依赖

```
<dependency>
    <groupId>com.github.theborakompanioni</groupId>
    <artifactId>thymeleaf-extras-shiro</artifactId>
    <version>2.0.0</version>
</dependency>
```

2、配置类添加 ShiroDialect

```
@Bean
public ShiroDialect shiroDialect(){
    return new ShiroDialect();
}
```

3、index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
xmlns:shiro="http://www.thymeleaf.org/thymeleaf-extras-
shiro">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="shortcut icon" href="#"/>
</head>
<body>
    <h1>index</h1>
    <div th:if="${session.account != null}">
        <span th:text="${session.account.username}+'欢
迎回来！'"></span><a href="/logout">退出</a>
    </div>
    <a href="/main">main</a> <br/>
    <div shiro:hasPermission="manage">
        <a href="manage">manage</a> <br/>
    </div>
    <div shiro:hasRole="administrator">
        <a href="/administrator">administrator</a>
    </div>
</body>
```

```
</html>
```