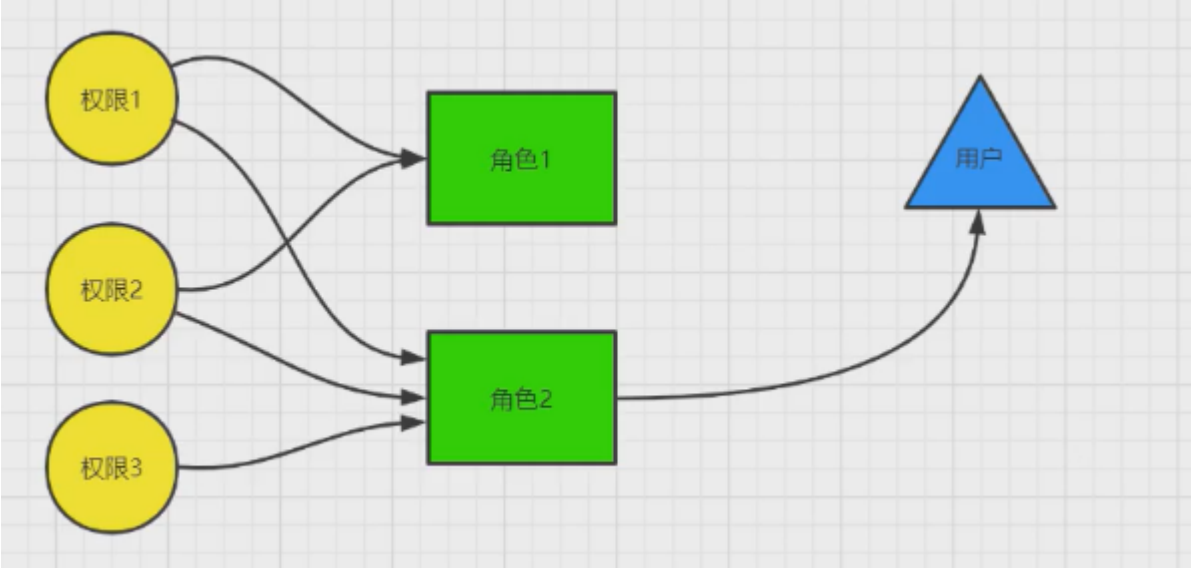
1.Shiro

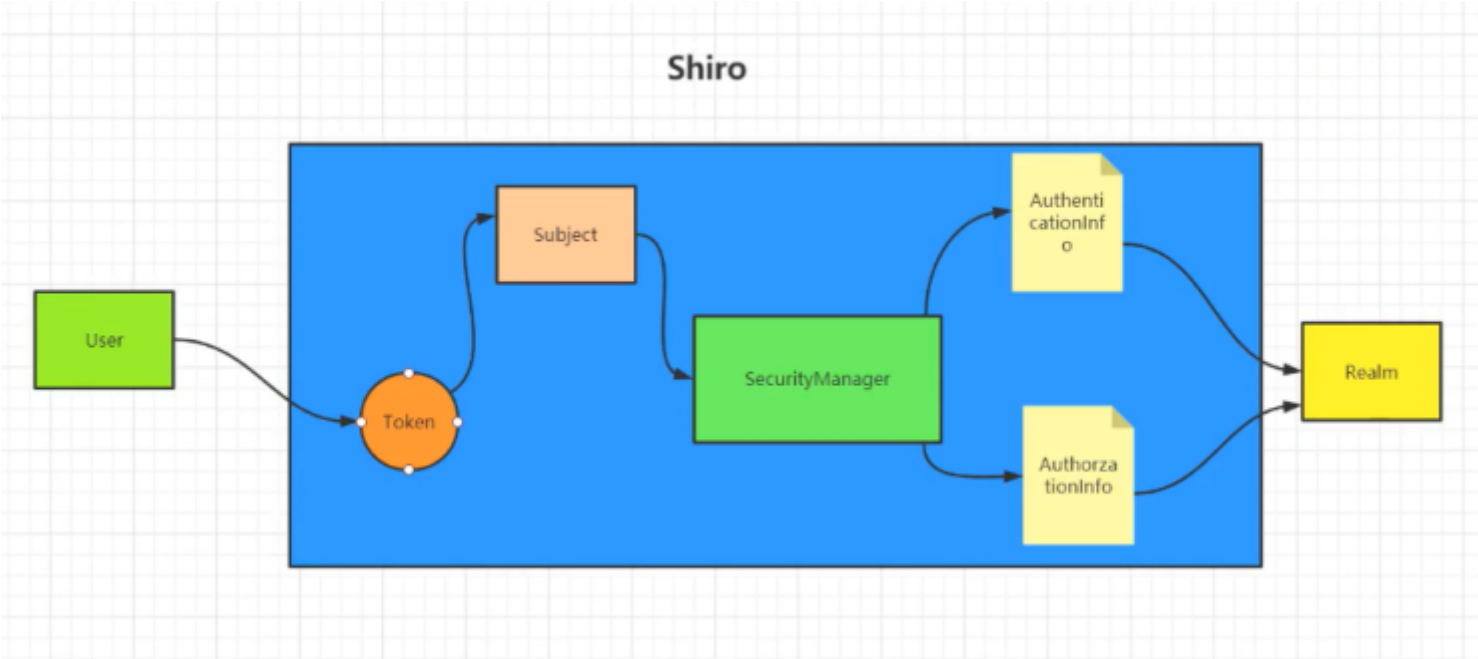
官网：http://shiro.apache.org/ <http://shiro.apache.org/>
是一款主流的java安全框架，不依赖任何容器，可以运行在JavaSE或JavaEE项目中，它的主要作用是对访问系统的用户进行身份认证、授权、会话管理、加密等操作。
shiro是用来解决安全管理的系统化框架

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

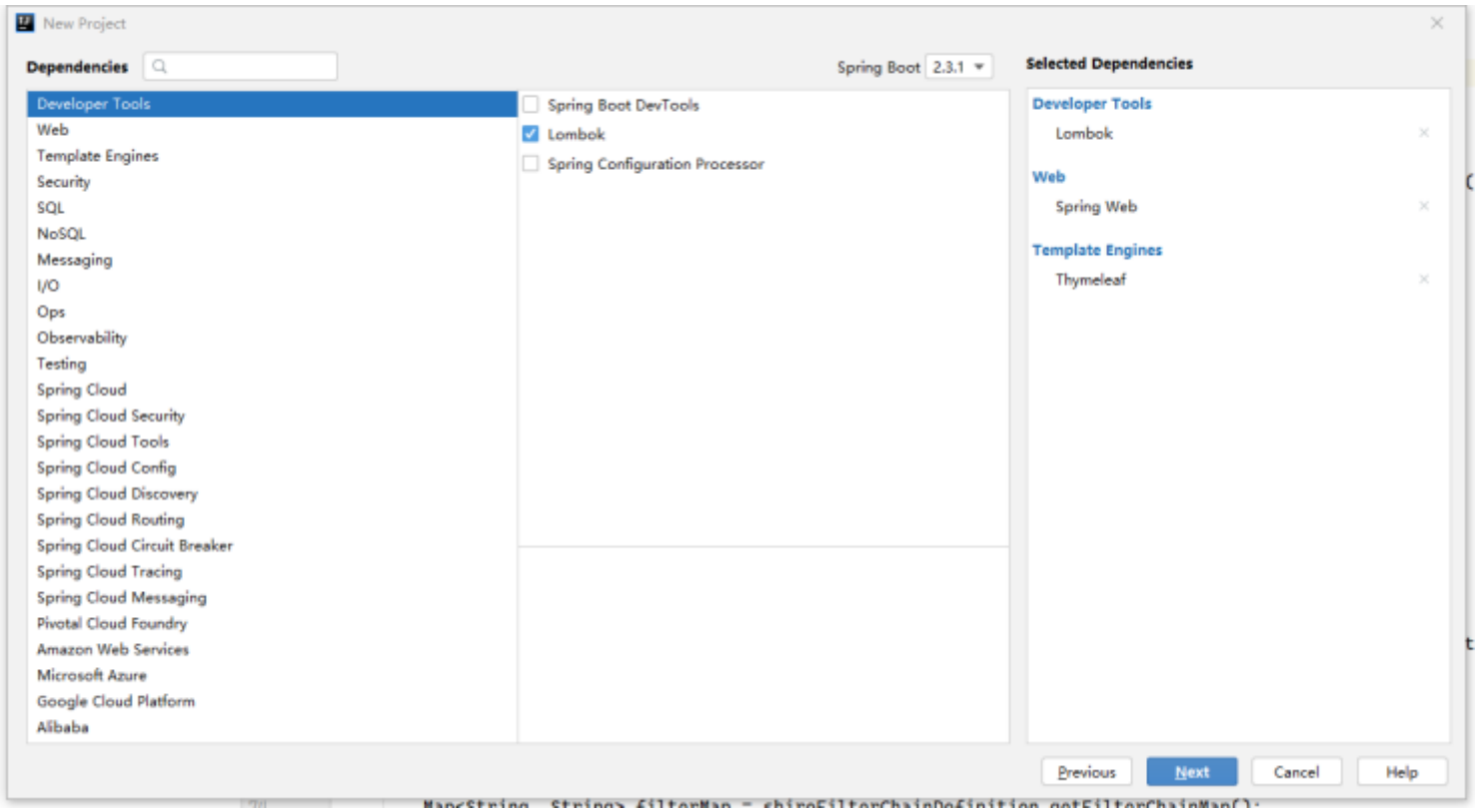


1.UsernamePasswordToken,Shiro用来封装用户登录信息，使用用户的登录信息来创建令牌Token。
2.SecurityManager，Shiro的核心部分，负责安全认证和授权
3.Suject，Shiro的一个抽象概念，包含了用户信息
4.Realm，开发者自定义的模块，根据项目的需求，验证和授权的逻辑全部写在Realm中。
5.AuthenticationInfo，用户的角色信息集合，认证时使用。
6.AuthorzationInfo角色的权限信息集合，授权时使用。
7.DefaultWebSecurityManager，安全管理器，开发者自定义的Realm需要注入到DefaultWevSecurityDManager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。



3.SpringBoot集成Shiro

在IDEA中创建项目：



1.建好后导入spring-shiro依赖

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>

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

        <!--支持mysql-->
        <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

```
package com.snailjw.realm;

import com.snailjw.entity.Account;
import com.snailjw.service.AccountService;
import org.apache.shiro.authc.*;
import org.apache.shiro.authz.AuthorizationInfo;
import org.apache.shiro.realm.AuthorizingRealm;
import org.apache.shiro.subject.PrincipalCollection;
import org.springframework.beans.factory.annotation.Autowired;
```

```
public class AccountRealm 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)
        UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;
        Account account = accountService.findByUsername(token.getUsername());
        if(account != null){
            return new SimpleAuthenticationInfo(account,account.getPassword(),getName());
        }
        return null;
    }
}
```

```
package com.snailjw.config;

import com.snailjw.realm.AccountRealm;
import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@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("accountRealm") AccountRealm accou
        DefaultWebSecurityManager manager= new DefaultWebSecurityManager();
        manager.setRealm(accountRealm);
        return manager;
    }

    @Bean
    public AccountRealm accountRealm(){
        return new AccountRealm();
    }
}
```

编写认证和授权规则：

认证过滤器：

anon:无需认证

authc:必须认证

authBassic: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。例如：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、administator.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manager授权才能访问manager.html

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

Shiro整个Thymeleaf

1.pom.xml引入依赖

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

2.配置类中添加方言

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

3.index.html使用