第1章 Shiro权限实战课程介绍

第1集 Shiro权限实战课程介绍

简介：讲解为什么要学这门课，Shiro权限框架的课程大纲和学后水平

- 公司新项目需要用到、要么就是需要接收别人的代码、个人技术栈的成长
- Springboot2.x/SpringMVC + Maven + jdk8 + IDEA/Eclipse

- 学后水平：掌握Shiro在公司中实际的使用，包括明白里面的核心原理

第2集 权限控制和初学JavaWeb处理访问权限控制

简介：讲解什么是权限控制，初学JavaWeb时处理流程

- 什么是权限控制：
    - 忽略特别细的概念，比如权限能细分很多种，功能权限，数据权限，管理权限等
    - 理解两个概念：用户和资源，让指定的用户，只能操作指定的资源（CRUD）
- 初学javaweb时怎么做
    - Filter接口中有一个doFilter方法，自己编写好业务Filter，并配置对哪个web资源进行拦截后

    - 如果访问的路径命中对应的Filter，则会执行doFilter()方法，然后判断是否有权限进行访问对应的资源
    - /api/user/info?id=1

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws Exception {
        HttpServletRequest httpRequest=(HttpServletRequest)request;
        HttpServletResponse httpResponse=(HttpServletResponse)response;

        HttpSession session=httpRequest.getSession();

        if(session.getAttribute("username")!=null){
            chain.doFilter(request, response);
        } else {
            httpResponse.sendRedirect(httpRequest.getContextPath()+"/login.jsp");
        }

    }
```


第2章 大话权限框架核心知识ACL和RBAC

第1集 权限框架设计之ACL和RBAC讲解

简介：介绍什么是ACL和RBAC

- ACL: Access Control List 访问控制列表
    - 以前盛行的一种权限设计，它的核心在于用户直接和权限挂钩
    - 优点：简单易用，开发便捷
    - 缺点：用户和权限直接挂钩，导致在授予时的复杂性，比较分散，不便于管理
    - 例子：常见的文件系统权限设计, 直接给用户加权限
- RBAC: Role Based Access Control
    - 基于角色的访问控制系统。权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限
    - 优点：简化了用户与权限的管理，通过对用户进行分类，使得角色与权限关联起来
    - 缺点：开发对比ACL相对复杂
    - 例子：基于RBAC模型的权限验证框架与应用 Apache Shiro、spring Security
- BAT企业 ACL，一般是对报表系统，阿里的ODPS

- 总结：不能过于复杂，规则过多，维护性和性能会下降， 更多分类 ABAC、PBAC等

第2集 主流权限框架介绍和技术选型讲解

简介：介绍主流的权限框架 Apache Shiro、spring Security

- 什么是 spring Security：官网基础介绍
    - 官网：https://spring.io/projects/spring-security

  Spring Security是一个能够为基于Spring的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在Spring应用上下文中配置的Bean，充分利用了Spring IoC，DI（控制反转Inversion of Control ,DI:Dependency Injection 依赖注入）和AOP（面向切面编程）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。

  一句话：Spring Security 的前身是 Acegi Security ，是 Spring 项目组中用来提供安全认证服务的框架

- 什么是 Apache Shiro：官网基础介绍
    - https://github.com/apache/shiro

      Apache Shiro是一个强大且易用的Java安全框架,执行身份验证、授权、密码和会话管理。使用Shiro的易于理解的API,您可以快速、轻松地获得任何应用程序,从最小的移动应用程序到最大的网络和企业应用程序。

      一句话：Shiro是一个强大易用的Java安全框架,提供了认证、授权、加密和会话管理等功能

- 两个优缺点，应该怎么选择
    - Apache Shiro比Spring Security，前者使用更简单
    - Shiro 功能强大、 简单、灵活， 不跟任何的框架或者容器绑定，可以独立运行
    - Spring Security 对Spring 体系支持比较好，脱离Spring体系则很难开发
    - SpringSecutiry 支持Oauth鉴权 https://spring.io/projects/spring-security-oauth，Shiro需要自己实现
    - ......

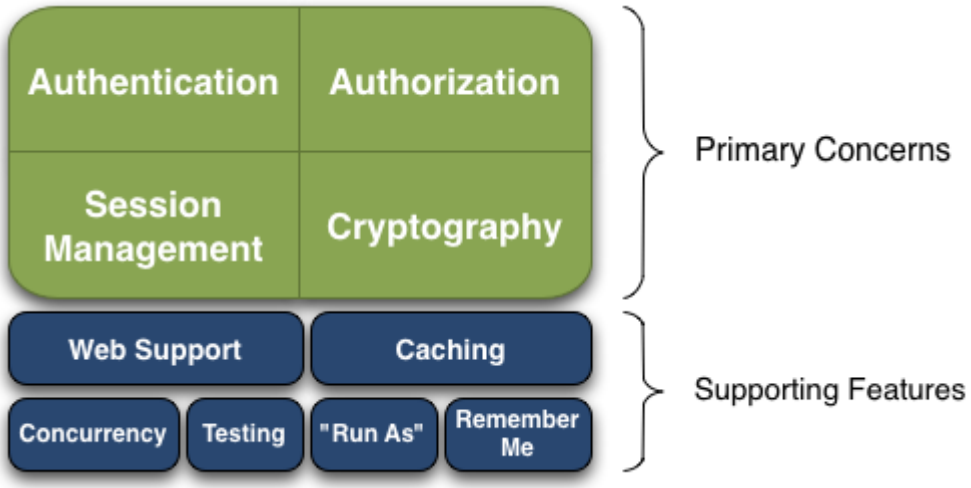
      总结：两个框架没有谁超过谁，大体功能一致，新手一般先推荐Shiro，学习会容易点

第3章 Apache Shiro基础概念知识和架构讲解

第1集 Shiro核心知识之架构图交互和四大模块讲解

简介：讲解Shiro架构图交互和四大核心模块 身份认证，授权，会话管理和加密

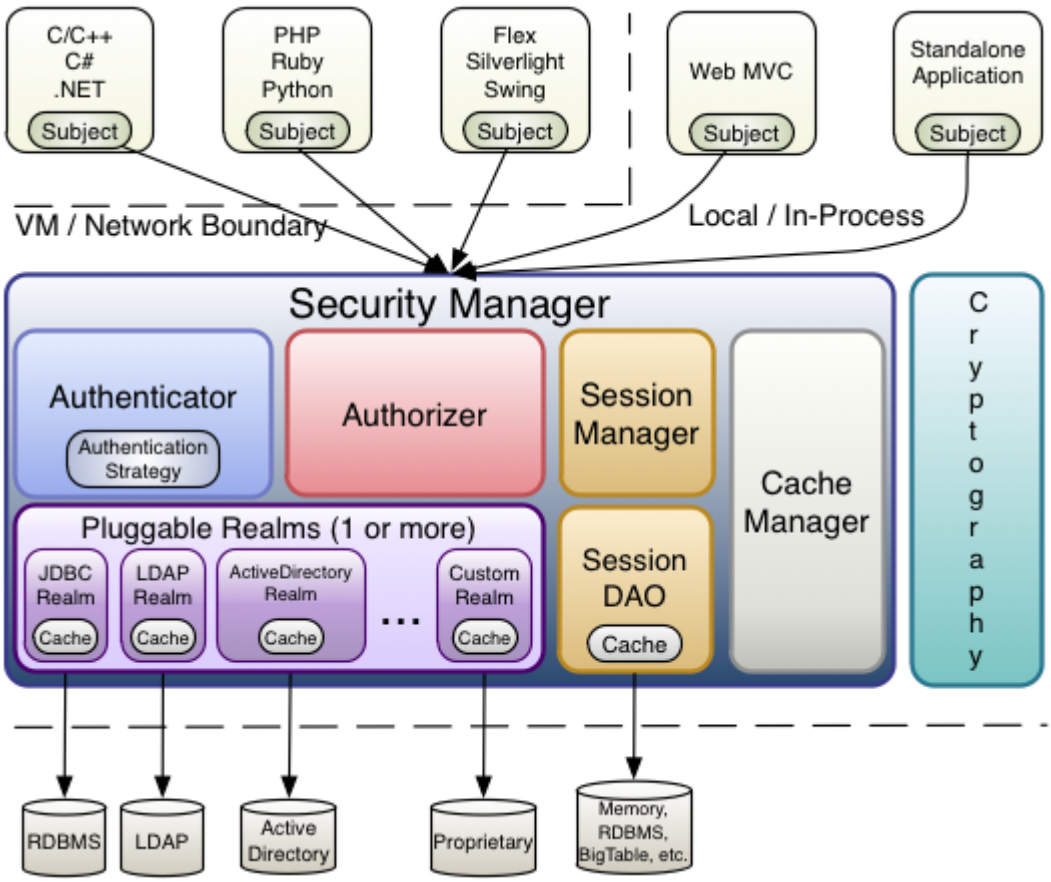
- 直达Apache Shiro官网 http://shiro.apache.org/introduction.html
- 什么是身份认证
  - Authentication，身份证认证，一般就是登录
- 什么是授权
  - Authorization，给用户分配角色或者访问某些资源的权限
- 什么是会话管理
  - Session Management, 用户的会话管理员，多数情况下是web session
- 什么是加密
  - Cryptography, 数据加解密，比如密码加解密等



第2集 用户访问Shrio权限控制运行流程和常见概念讲解

简介：讲解用户访问整合Shrio的系统，权限控制的运行流程和Shiro常见名称讲解

- 直达官网 ：http://shiro.apache.org/architecture.html
- Subject
  - 我们把用户或者程序称为主体（如用户，第三方服务，cron作业），主体去访问系统或者资源
- SecurityManager
  - 安全管理器，Subject的认证和授权都要在安全管理器下进行
- Authenticator
  - 认证器，主要负责Subject的认证
- Realm
  - 数据域，Shiro和安全数据的连接器，好比jdbc连接数据库； 通过realm获取认证授权相关信息
- Authorizer
  - 授权器，主要负责Subject的授权, 控制subject拥有的角色或者权限
- Cryptography
  - 加解密，Shiro的包含易于使用和理解的数据加解密方法，简化了很多复杂的api
- Cache Manager
  - 缓存管理器，比如认证或授权信息，通过缓存进行管理，提高性能



更多资料导航：http://shiro.apache.org/reference.html

第4章 Springboot2.x整合 Apache Shiro快速上手实战

第1集 SpringBoot2.x整合Shiro权限认证项目搭建

简介：使用SpringBoot2.x整合Shiro权限认证

- Maven3.5 + Jdk8 + Springboot 2.X + IDEA (Eclipse也可以)
- 创建SpringBoot项目
    - https://start.spring.io/

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid</artifactId>
    <version>1.1.6</version>
</dependency>
```

- 整合Shiro相关jar包

```
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-spring</artifactId>
    <version>1.4.0</version>
</dependency>
```

第2集 快速上手之Shiro认证和授权流程实操上集

简介：讲解Shrio的认证和授权实操上集

认证：用户身份识别，俗称为用户“登录”

第3集Shiro认证和授权流程和常用API梳理下集

简介:讲解Shiro的授权实操和常用Api 梳理下集

```
//是否有对应的角色
subject.hasRole("root")

//获取subject名
subject.getPrincipal()

//检查是否有对应的角色，无返回值，直接在SecurityManager里面进行判断
subject.checkRole("admin")

//检查是否有对应的角色
subject.hasRole("admin")

//退出登录
subject.logout();
```


第5章 详细讲解Apache Shiro realm实战

第1集 Shiro安全数据来源之Realm讲解

简介：讲解shiro默认自带的realm和常见使用方法

- realm作用：Shiro 从 Realm 获取安全数据
- 默认自带的realm：idae查看realm继承关系，有默认实现和自定义继承的realm
- 两个概念
    - principal : 主体的标示，可以有多个，但是需要具有唯一性，常见的有用户名，手机号，邮箱等
    - credential：凭证, 一般就是密码
    - 所以一般我们说 principal + credential 就账号 + 密码
- 开发中，往往是自定义realm , 即集成 AuthorizingRealm

第2集 快速上手之Shiro内置IniRealm实操和权限验证api

简介：讲解Shiro内置 ini realm实操

```
# 格式 name=password,role1,role2,..roleN
[users]
# user 'root' with password 'secret' and the 'admin' role,
jack = 456, user

# user 'guest' with the password 'guest' and the 'guest' role
xdcalss = 123, root



# 格式 role=permission1,permission2...permissionN   也可以用通配符
# 下面配置user的权限为所有video:find,video:buy，如果需要配置video全部操作crud 则 user = video:*
[roles]
user = video:find,video:buy
# 'admin' role has all permissions, indicated by the wildcard '*'
admin = *
```

第3集 快速上手之Shiro内置JdbcRealm实操

简介：讲解Shiro内置 JdbcRealm实操

- 使用jdbcrealm.ini

```
#注意  文件格式必须为ini，编码为ANSI

#声明Realm，指定realm类型
jdbcRealm=org.apache.shiro.realm.jdbc.JdbcRealm

#配置数据源
#dataSource=com.mchange.v2.c3p0.ComboPooledDataSource

dataSource=com.alibaba.druid.pool.DruidDataSource

# mysql-connector-java 5 用的驱动url是com.mysql.jdbc.Driver，mysql-connector-java6以后用的是com.mysql.cj.jdbc.Driver
dataSource.driverClassName=com.mysql.cj.jdbc.Driver

#避免安全警告
dataSource.url=jdbc:mysql://120.76.62.13:3606/xdclass_shiro?characterEncoding=UTF-8&serverTimezone=UTC&useSSL=false

dataSource.username=test

dataSource.password=Xdclasstest

#指定数据源
jdbcRealm.dataSource=$dataSource


#开启查找权限，默认是false，不会去查找角色对应的权限，坑！！！！！
jdbcRealm.permissionsLookupEnabled=true


#指定SecurityManager的Realms实现，设置realms，可以有多个，用逗号隔开
securityManager.realms=$jdbcRealm
```

- 方式二

```
 DefaultSecurityManager securityManager = new DefaultSecurityManager();

        DruidDataSource ds = new DruidDataSource();
        ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
        ds.setUrl("jdbc:mysql://120.76.62.13:3606/xdclass_shiro?characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false");
        ds.setUsername("test");
        ds.setPassword("Xdclasstest");
```

第4集 Apache Shiro 自定义Realm实战

简介：讲解自定义Realm实战基础

- 步骤：
  - 创建一个类 ，继承AuthorizingRealm->AuthenticatingRealm->CachingRealm->Realm
  - 重写授权方法 doGetAuthorizationInfo
  - 重写认证方法 doGetAuthenticationInfo
- 方法：
  - 当用户登陆的时候会调用 doGetAuthenticationInfo
  - 进行权限校验的时候会调用: doGetAuthorizationInfo
- 对象介绍
  - UsernamePasswordToken ： 对应就是 shiro的token中有Principal和Credential
    - `UsernamePasswordToken-》HostAuthenticationToken-》AuthenticationToken`
  - SimpleAuthorizationInfo：代表用户角色权限信息
  - SimpleAuthenticationInfo ：代表该用户的认证信息

第5集 深入Shiro源码解读认证授权流程

简介：Shiro认证和授权流程的源码解读，和断点测试

认证流程解读：subject.login(usernamePasswordToken);

```
DelegatingSubject->login()
DefaultSecurityManager->login()
AuthenticatingSecurityManager->authenticate()
AbstractAuthenticator->authenticate()
ModularRealmAuthenticator->doAuthenticate()
ModularRealmAuthenticator->doSingleRealmAuthentication()
AuthenticatingRealm->getAuthenticationInfo()

补充：有些同学找不到密码验证方法 AuthenticatingRealm-> assertCredentialsMatch()
```

授权流程解读：subject.checkRole("admin")

```
DelegatingSubject->checkRole()
AuthorizingSecurityManager->checkRole()
ModularRealmAuthorizer->checkRole()
AuthorizingRealm->hasRole()
AuthorizingRealm->doGetAuthorizationInfo()
```

第6章 Shiro权限认证Web案例知识点讲解

###第1集 Shiro内置的Filter过滤器讲解

简介：讲解shiro内置的过滤器讲解

- 核心过滤器类：DefaultFilter, 配置哪个路径对应哪个拦截器进行处理
- authc：org.apache.shiro.web.filter.authc.FormAuthenticationFilter
  - 需要认证登录才能访问
- user：org.apache.shiro.web.filter.authc.UserFilter
  - 用户拦截器，表示必须存在用户。
- anon：org.apache.shiro.web.filter.authc.AnonymousFilter
  - 匿名拦截器，不需要登录即可访问的资源，匿名用户或游客，一般用于过滤静态资源。
- roles：org.apache.shiro.web.filter.authz.RolesAuthorizationFilter
  - 角色授权拦截器，验证用户是或否拥有角色。
  - 参数可写多个，表示某些角色才能通过，多个参数时写 roles["admin,user"]，当有多个参数时必须每个参数都通过才算通过
- perms：org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter
  - 权限授权拦截器，验证用户是否拥有权限
  - 参数可写多个，表示需要某些权限才能通过，多个参数时写 perms["user, admin"]，当有多个参数时必须每个参数都通过才算可以

- authcBasic：org.apache.shiro.web.filter.authc.BasicHttpAuthenticationFilter
  - httpBasic 身份验证拦截器。
- logout：org.apache.shiro.web.filter.authc.LogoutFilter
  - 退出拦截器，执行后会直接跳转到`shiroFilterFactoryBean.setLoginUrl()`; 设置的 url

- port：org.apache.shiro.web.filter.authz.PortFilter
  - 端口拦截器, 可通过的端口。
- ssl：org.apache.shiro.web.filter.authz.SslFilter
  - ssl拦截器，只有请求协议是https才能通过。

第2集 Shiro的Filter配置路径讲解

简介：讲解Filter配置路径

- /admin/video /user /pub
- 路径通配符支持 ?、*、**，注意通配符匹配不 包括目录分隔符“/”
- 心 可以匹配所有，不加*可以进行前缀匹配，但多个冒号就需要多个 * 来匹配

```
URL权限采取第一次匹配优先的方式
? : 匹配一个字符，如 /user? ， 匹配 /user3，但不匹配/user/;
* : 匹配零个或多个字符串，如 /add* ,匹配 /addtest，但不匹配 /user/1
** : 匹配路径中的零个或多个路径，如 /user/** 将匹 配 /user/xxx 或 /user/xxx/yyy

例子
/user/**=filter1
/user/add=filter2

请求 /user/add  命中的是filter1拦截器
```

- 性能问题：通配符比字符串匹配会复杂点，所以性能也会稍弱，推荐是使用字符串匹配方式

第3集 Shiro 数据安全之数据加解密

简介: 讲解数据安全核心知识，介绍常见的处理办法，Shiro 里的 CredentialsMatcher使用

- 为啥要加解密
  - 明文数据容易泄露，比如密码明文存储，万一泄露则会造成严重后果
- 什么是散列算法
  - 一般叫hash，简单的说就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数，适合存储密码，比如MD5
- 什么是salt(盐) 667788——》aabbcc
  - 如果直接通过散列函数得到加密数据，容易被对应解密网站暴力破解，一般会在应用程序里面加特殊的自动进行处理，比如用户id，例子：加密数据 = MD5(明文密码+用户id), 破解难度会更大，也可以使用多重散列，比如多次md5
- Shiro里面 CredentialsMatcher，用来验证密码是否正确，
  - 源码：`AuthenticatingRealm -> assertCredentialsMatch()`

一般会自定义验证规则

```
        @Bean
    public HashedCredentialsMatcher hashedCredentialsMatcher(){
        HashedCredentialsMatcher hashedCredentialsMatcher = new
HashedCredentialsMatcher();

        //散列算法，使用MD5算法;
        hashedCredentialsMatcher.setHashAlgorithmName("md5");

        //散列的次数，比如散列两次，相当于 md5(md5("xxx"));
        hashedCredentialsMatcher.setHashIterations(2);

        return hashedCredentialsMatcher;
    }
```

第4集 Shiro权限控制注解和编程方式讲解

简介：讲解权限角色控制 @RequiresRoles, @RequiresPermissions等注解的使用和编程式控制

配置文件的方式

使用ShiroConfig

注解方式

- @RequiresRoles(value={"admin", "editor"}, logical= Logical.AND)
  - 需要角色 admin 和 editor两个角色 AND表示两个同时成立
- @RequiresPermissions (value={"user:add", "user:del"}, logical= Logical.OR)
  - 需要权限 user:add 或 user:del权限其中一个，OR是或的意思。

- @RequiresAuthentication
    - 已经授过权，调用Subject.isAuthenticated()返回true
- @RequiresUser
    - 身份验证或者通过记 住我登录的

编程方式

```
Subject subject = SecurityUtils.getSubject();
//基于角色判断
if(subject.hasRole(“admin”)) {
        //有角色，有权限
} else {
        //无角色，无权限

}
//或者权限判断
if(subject.isPermitted("/user/add")){
    //有权限
}else{
    //无权限
}
```

- 常见API
    - subject.hasRole("xxx");
    - subject.isPermitted("xxx");
    - subject. isPermittedAll("xxxxx","yyyy");
    - subject.checkRole("xxx"); // 无返回值，可以认为内部使用断言的方式

第5集 Shiro 缓存模块讲解

简介：讲解缓存的作用和Shiro的缓存模块

- 什么是shiro缓存
    - shiro中提供了对认证信息和授权信息的缓存。

        - 默认是关闭认证信息缓存的，对于授权信息的缓存shiro默认开启的(因为授权的数据量大)

- AuthenticatingRealm 及 AuthorizingRealm 分别提供了对AuthenticationInfo 和 AuthorizationInfo 信息的缓存。

第6集 Shiro Session模块讲解

简介：讲解Shiro Session模块作用和SessionManager

- 什么是会话session
    - 用户和程序直接的链接，程序可以根据session识别到哪个用户，和javaweb中的session类似
- 什么是会话管理器SessionManager
    - 会话管理器管理所有subject的所有操作，是shiro的核心组件
    - 核心方法：
    - ```
      //开启一个session
      Session start(SessionContext context);
      //指定Key获取session
      Session getSession(SessionKey key)
      ```
    - shiro中的会话管理器有多个实现
- SessionDao 会话存储/持久化
    - SessionDAO AbstractSessionDAO CachingSessionDAO EnterpriseCacheSessionDAO MemorySessionDAO
    - 核心方法
      ```

```
//创建
Serializable create(Session session);
//获取
Session readSession(Serializable sessionId) throws UnknownSessionException;
//更新
void update(Session session)
//删除，会话过期时会调用
void delete(Session session);
//获取活跃的session
Collection<Session> getActiveSessions();
```

- 会话存储有多个实现

附属资料：

```
RememberMe
1、 Cookie 写到客户端并 保存
2、 通过调用subject.login()前，设置 token.setRememberMe(true);
3、 关闭浏览器再重新打开;会发现浏览器还是记住你的
4、 注意点：
  - subject.isAuthenticated() 表示用户进行了身份验证登录的，即Subject.login 进行了登录
  - subject.isRemembered() 表示用户是通过RememberMe登录的
  - subject.isAuthenticated()==true，则 subject.isRemembered()==false， 两个互斥
  - 总结：特殊页面或者API调用才需要authc进行验证拦截，该拦截器会判断用户是否是通过
subject.login()登录，安全性更高，其他非核心接口或者页面则通过user拦截器处理即可
```


第7章 Apache Shiro整合SpringBoot2.x综合案例实战

第1集 Shiro整合SpringBoot2.x案例实战介绍

简介：介绍Apache Shiro整合SpringBoot2.x综合实战和技术栈

- 技术选型：前后端分离的权限检验 + SpringBoot2.x + Mysql + Mybatis + Shiro + Redis + IDEA + JDK8

第2集 基于RBAC权限控制实战之Mysql数据库设计

简介：设计案例实战数据库 用户-角色-权限 及关联表

- 用户
- 角色
- 权限

第3集 SpringBoot2.x项目框架和依赖搭建

简介：使用springboot+mybatis+shiro搭建项目基础框架

- 官方下载 https://start.spring.io/
- 项目依赖

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

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

        <dependency>
                <groupId>org.mybatis.spring.boot</groupId>
                <artifactId>mybatis-spring-boot-starter</artifactId>
                <version>1.3.2</version>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>

        <!--阿里巴巴druid数据源-->
        <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>1.1.6</version>
        </dependency>
```

```
        <!--spring整合shiro-->
        <dependency>
                <groupId>org.apache.shiro</groupId>
                <artifactId>shiro-spring</artifactId>
                <version>1.4.0</version>
        </dependency>

</dependencies>

<build>
        <plugins>
                <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
        </plugins>
</build>

<repositories>
        <repository>
                <id>spring-snapshots</id>
                <name>Spring Snapshots</name>
                <url>https://repo.spring.io/snapshot</url>
                <snapshots>
                        <enabled>true</enabled>
                </snapshots>
        </repository>
        <repository>
                <id>spring-milestones</id>
                <name>Spring Milestones</name>
                <url>https://repo.spring.io/milestone</url>
        </repository>
</repositories>
<pluginRepositories>
        <pluginRepository>
                <id>spring-snapshots</id>
                <name>Spring Snapshots</name>
                <url>https://repo.spring.io/snapshot</url>
                <snapshots>
                        <enabled>true</enabled>
                </snapshots>
        </pluginRepository>
        <pluginRepository>
                <id>spring-milestones</id>
                <name>Spring Milestones</name>
                <url>https://repo.spring.io/milestone</url>
        </pluginRepository>
</pluginRepositories>
```

第4集 案例实战之权限相关服务接口开发

简介：开发用户-角色-权限 相关Service和Dao层

- 数据库配置

```
#==============================数据库相关配置=======================================
spring.datasource.driver-class-name =com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://120.76.62.13:3606/xdclass_shiro?
useUnicode=true&characterEncoding=utf-8&useSSL=false
spring.datasource.username =test
spring.datasource.password =Xdclasstest
#使用阿里巴巴druid数据源，默认使用自带的
#spring.datasource.type =com.alibaba.druid.pool.DruidDataSource
#开启控制台打印sql
mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

# mybatis 下划线转驼峰配置,两者都可以
#mybatis.configuration.mapUnderscoreToCamelCase=true
mybatis.configuration.map-underscore-to-camel-case=true
```

第5集 案例实战之用户角色权限多对多关联查询SQL

简介：开发用户-角色-权限 多对多关联查询SQL

- 第一步 查询用户对应的角色映射关系

```
select * from user u
left join user_role ur on u.id=ur.user_id
where  u.id=3
```

- 第二步 查询用户对应的角色信息

```
select * from user u
left join user_role ur on u.id=ur.user_id
left join role r on ur.role_id = r.id
where  u.id=3
```

- 第三步 查询角色和权限的关系

```
select * from user u
left join user_role ur on u.id=ur.user_id
left join role r on ur.role_id = r.id
left join role_permission rp on r.id=rp.role_id
where  u.id=1
```

- 第四步 查询角色对应的权限信息（某个用户具备的角色和权限集合）

```
select * from user u
left join user_role ur on u.id=ur.user_id
left join role r on ur.role_id = r.id
left join role_permission rp on r.id=rp.role_id
left join permission p on rp.permission_id=p.id
where  u.id=1
```

第6集 案例实战自定义CustomRealm实战

简介：自定义CustomRealm开发实战

- 继承 AuthorizingRealm
- 重写 doGetAuthorizationInfo
- 重写 doGetAuthenticationInfo

第7集 项目实战之ShiroFilterFactoryBean配置实战

简介：讲解ShiroFilterFactoryBean配置实战

- 配置流程和思路
- shiroFilterFactoryBean-》
    - SecurityManager-》
        - CustomSessionManager
        - CustomRealm-》hashedCredentialsMatcher
- SessionManager
    - DefaultSessionManager： 默认实现，常用于javase
    - ServletContainerSessionManager: web环境
    - DefaultWebSessionManager：常用于自定义实现

第8集 前后端分离自定义SessionManager验证

简介：讲解前后端分离情况下自定义SessionManager

第9集 API权限拦截验证实战

简介： 讲解使用filterChainDefinitionMap控制api访问角色权限

localhost:8080/pub/login

localhost:8080/authc/video/play_record

localhost:8080/admin/video/order

localhost:8080/video/update

第10集 使用Shiro Logout和加密处理

简介：Shiro logout常见错误思路和加密处理

```
@Test
public void testMD5(){
            //加密算法
    String hashName = "md5";
            //密码明文
    String pwd = "123";
            //加密函数，使用shiro自带的
    Object result = new SimpleHash(hashName, pwd, null, 2);
    System.out.println(result);
}
```


第8章 权限控制综合案例实战进阶

第1集 实战进阶之自定义Shiro Filter过滤器上集

简介：权限控制综合案例 自定义Shiro Filter过滤器

- 背景知识：
  - /admin/order= roles["admin, root"] ，表示 /admin/order 这个接口需要用户同时具备 admin 与 root 角色才可访问, 相当于hasAllRoles() 这个判断方法
- 我们的需求：
  - 订单信息，可以由角色 普通管理员 admin 或者 超级管理员 root 查看
  - 只要用户具备其中一个角色即可

- 自定义Filter

```
// mappedValue -》roles[admin,root]
public class CustomRolesOrAuthorizationFilter extends AuthorizationFilter {

    @Override
    protected boolean isAccessAllowed(ServletRequest req, ServletResponse resp, Object
mappedValue) throws Exception {
```

```
        Subject subject = getSubject(req, resp);

        //获取当前访问路径所需要的角色集合
        String[] rolesArray = (String[]) mappedValue;

                //没有角色限制，有权限访问
        if (rolesArray == null || rolesArray.length == 0) {
            return true;
        }

        //当前subject是rolesArray中的任何一个，则有权限访问
        for (int i = 0; i < rolesArray.length; i++) {
            if (subject.hasRole(rolesArray[i])) {
                return true;
            }
        }

        return false;
    }
}
```

- FactoryBean配置

```
Map<String, Filter> filtersMap = new LinkedHashMap<>();
filtersMap.put("roleOrFilter",new CustomRolesOrAuthorizationFilter());
filterChainDefinitionMap.put("/admin/**","roleOrFilter[admin,root]");
shiroFilterFactoryBean.setFilters(filtersMap);
```

第2集 实战进阶之自定义Shiro Filter过滤器下集

简介：自定义Shiro Filter过滤器验证

- 配置不同的角色，验证自定义过滤器是否有效

```
登录(post)
localhost:8080/pub/login

管理员查看后台信息(get)
localhost:8080/admin/video/order
```

第3集 性能提升之Redis整合CacheManager

简介：讲解使用Redis整合CacheManager

- 使用原因？
  - 授权的时候每次都去查询数据库，对于频繁访问的接口，性能和响应速度比较慢，所以使用缓存

- 步骤
  - 加依赖

```
<!-- shiro+redis缓存插件 -->
    <dependency>
                        <groupId>org.crazycake</groupId>
                        <artifactId>shiro-redis</artifactId>
                        <version>3.1.0</version>
              </dependency>
```

  - 配置bean

```
        //使用自定义的cacheManager
    securityManager.setCacheManager(cacheManager());

     /**
      * 配置redisManager
      *
      */
     public RedisManager getRedisManager(){
         RedisManager redisManager = new RedisManager();
         redisManager.setHost("localhost");
         redisManager.setPort(6379);
         return redisManager;
     }


     /**
      * 配置具体cache实现类
      * @return
      */
     public RedisCacheManager cacheManager(){
         RedisCacheManager redisCacheManager = new RedisCacheManager();
         redisCacheManager.setRedisManager(getRedisManager());
          //设置过期时间，单位是秒，20s,
         redisCacheManager.setExpire(20);
```

```
        return redisCacheManager;
    }
```

- 安装redis （如何不会使用redis，则参考网上的博客文章）
  - 建议本地安装，默认是不能外网访问的，然后不是守护进程方式
  - https://www.cnblogs.com/it-cen/p/4295984.html
- 原有的问题

```
class java.lang.String must has getter for field: authCacheKey or id\nWe need a field to identify this Cache Object in Redis. So you need to defined an id field which you can get unique id to identify this principal. For example, if you use UserInfo as Principal class, the id field maybe userId, userName, email, etc. For example, getUserId(), getUserName(), getEmail(), etc.\nDefault value is authCacheKey or id, that means your principal object has a method called \"getAuthCacheKey()\" or \"getId()\""
```

- 改造原有的逻辑，修改缓存的唯一key
  - doGetAuthorizationInfo 方法
    原有：

```
        String username = (String)principals.getPrimaryPrincipal();
        User user = userService.findAllUserInfoByUsername(username);
```

    改为

```
        User newUser = (User)principals.getPrimaryPrincipal();
    User user = userService.findAllUserInfoByUsername(newUser.getUsername());
```

    doGetAuthenticationInfo方法
    原有：

```
return new SimpleAuthenticationInfo(username, user.getPassword(),
this.getClass().getName());
```

    改为

```
return new SimpleAuthenticationInfo(user, user.getPassword(),
this.getClass().getName());
```

第4集 性能提升之Redis整合SessionManager

简介：讲解使用Redis整合SessionManager，管理Session会话

- 为啥session也要持久化？
  - 重启应用，用户无感知，可以继续以原先的状态继续访问
- 怎么持久化？

```
//配置session持久化
customSessionManager.setSessionDAO(redisSessionDAO());
```

```
    /**
  * 自定义session持久化
  * @return
  */
 public RedisSessionDAO redisSessionDAO(){
     RedisSessionDAO redisSessionDAO = new RedisSessionDAO();
     redisSessionDAO.setRedisManager(getRedisManager());
     return redisSessionDAO;
 }
```

注意点：

- DO对象需要实现序列化接口 Serializable
- logout接口和以前一样调用，请求logout后会删除redis里面的对应的key,即删除对应的token

第5集 ShiroConfig常用bean类配置

简介：讲解ShiroConfig常用bean 介绍

- LifecycleBeanPostProcessor
  - 作用：管理shiro一些bean的生命周期 即bean初始化 与销毁

```
@Bean
public LifecycleBeanPostProcessor lifecycleBeanPostProcessor() {
        return new LifecycleBeanPostProcessor();
}
```

- AuthorizationAttributeSourceAdvisor
  - 作用：加入注解的使用，不加入这个AOP注解不生效(shiro的注解 例如 @RequiresGuest)

```
@Bean
 public AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor()
{
  AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor = new
AuthorizationAttributeSourceAdvisor();
        authorizationAttributeSourceAdvisor.setSecurityManager(securityManager());
        return authorizationAttributeSourceAdvisor;
}
```

- DefaultAdvisorAutoProxyCreator
  - 作用: 用来扫描上下文寻找所有的Advistor(通知器), 将符合条件的Advisor应用到切入点的Bean中，需要在LifecycleBeanPostProcessor创建后才可以创建

```
@Bean
@DependsOn("lifecycleBeanPostProcessor")
public  DefaultAdvisorAutoProxyCreator getDefaultAdvisorAutoProxyCreator(){
        DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator=new
DefaultAdvisorAutoProxyCreator();
        defaultAdvisorAutoProxyCreator.setUsePrefix(true);
        return defaultAdvisorAutoProxyCreator;
}
```


第9章 大话分布式应用的鉴权方式

第1集 单体应用到分布式应用下的鉴权方式介绍

简介：介绍单体应用到分布式应用下的鉴权方式

电商项目：商品服务，支付服务，用户服务

- 分布式session
- UUID
- JWT
    - https://www.cnblogs.com/cjsblog/p/9277677.html
- Oauth2.0
    - https://www.cnblogs.com/flashsun/p/7424071.html

第2集 分布式应用鉴权方式之Shiro整合SpringBoot下自定义SessionId

简介：基于原先项目，实现自定义sessionid

- Shiro 默认的sessionid生成 类名 SessionIdGenerator
- 创建一个类，实现 SessionIdGenerator 接口的方法

```
    /**
 * 自定义session持久化
 * @return
 */
public RedisSessionDAO redisSessionDAO(){
    RedisSessionDAO redisSessionDAO = new RedisSessionDAO();
    redisSessionDAO.setRedisManager(getRedisManager());

    //设置sessionid生成器
    redisSessionDAO.setSessionIdGenerator(new CustomSessionIdGenerator());

    return redisSessionDAO;
}
```

- 没有100%可靠的算法，暴力破解，穷举
    - 限制时间内ip登录错误次数
    - 增加图形验证码，不能过于简单，常用的OCR可以识别验证码

- 建议：微服务里面，特别是对C端用户的应用，不要做过于复杂的权限校验，特别是影响性能这块

第10章 Shiro课程总结

第1集 Apache shiro从入门到高级实战课程总结

简介：Apache shiro从入门到高级实战课程总结

小D课堂，愿景：让编程不在难学，让技术与生活更加有趣

相信我们，这个是可以让你学习更加轻松的平台，里面的课程绝对会让你技术不断提升

欢迎加小D讲师的微信： jack794666918

我们官方网站：https://xdclass.net

千人IT技术交流QQ群： 718617859

重点来啦：免费赠送你干货文档大集合，包含前端，后端，测试，大数据，运维主流技术文档（持续更新）

https://mp.weixin.qq.com/s/qYnjcDYGFDQorWmSfE7lpQ