5、CAS单点登录-自定义认证之Shiro、Rest（六）

# 接4代码 ，去掉jdbc 的验证配置

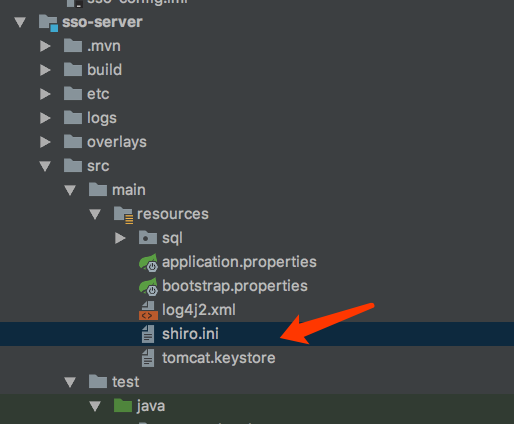
# 1、shiro校验登录

## 1、这里先使用shiro 配置文件配置用户名和密码以及角色和权限（不使用数据库）

### 1、shiro配置文件文件如下

#### 用户名 admin 密码123 角色 admin 权限admin、 staff 等

[main]  
cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager  
securityManager.cacheManager = $cacheManager  
  
[users]  
#密码123  
admin = 202cb962ac59075b964b07152d234b70, admin  
#不可登录，因为配置了需要角色admin  
#密码123456  
test = e10adc3949ba59abbe56e057f20f883e, developer  
  
[roles]  
admin = system,admin,staff,superuser:\*  
developer = commit:\*



## 2、sso-config 中添加shiro的配置信息

##  
# CAS Server Context Configuration  
#  
server.context-path=/cas  
server.port=8443  
  
  
#SSL配置  
server.ssl.enabled=true  
server.ssl.key-store=classpath:tomcat.keystore  
server.ssl.key-store-password=123456  
server.ssl.keyAlias=passport.sso.com  
#SSL配置  
  
  
# 5、shiro ,下面这里用的普通md5加密  
# Shiro Authentication 开始  
#允许登录的用户，必须要有以下角色，否则拒绝，多个逗号隔开  
cas.authn.shiro.requiredRoles=admin  
#允许登录的用户，必须要有以下权限，否则拒绝，多个逗号隔开  
cas.authn.shiro.requiredPermissions=staff  
#shir配置文件位置  
#由于目前classpath:shiro.ini会去临时目录找一遍，会发生一些错误，若用shiro，建议放到硬盘其他地方，写其他绝对路径  
cas.authn.shiro.config.location=classpath:shiro.ini  
#shiro name 唯一  
cas.authn.shiro.name=cas-shiro  
# 与Query Authentication一致的加密策略  
cas.authn.shiro.passwordEncoder.type=DEFAULT  
cas.authn.shiro.passwordEncoder.characterEncoding=UTF-8  
cas.authn.shiro.passwordEncoder.encodingAlgorithm=MD5  
# Shiro Authentication 结束  
  
  
  
  
server.max-http-header-size=2097152  
server.use-forward-headers=true  
server.connection-timeout=20000  
server.error.include-stacktrace=NEVER  
  
server.tomcat.max-http-post-size=2097152  
server.tomcat.basedir=build/tomcat  
server.tomcat.accesslog.enabled=true  
server.tomcat.accesslog.pattern=%t %a "%r" %s (%D ms)  
server.tomcat.accesslog.suffix=.log  
server.tomcat.max-threads=10  
server.tomcat.port-header=X-Forwarded-Port  
server.tomcat.protocol-header=X-Forwarded-Proto  
server.tomcat.protocol-header-https-value=https  
server.tomcat.remote-ip-header=X-FORWARDED-FOR  
server.tomcat.uri-encoding=UTF-8  
  
spring.http.encoding.charset=UTF-8  
spring.http.encoding.enabled=true  
spring.http.encoding.force=true  
  
##  
# CAS Cloud Bus Configuration  
#  
spring.cloud.bus.enabled=false  
  
endpoints.enabled=false  
endpoints.sensitive=true  
  
endpoints.restart.enabled=false  
endpoints.shutdown.enabled=false  
  
management.security.enabled=true  
management.security.roles=ACTUATOR,ADMIN  
management.security.sessions=if\_required  
management.context-path=/status  
management.add-application-context-header=false  
  
security.basic.authorize-mode=role  
security.basic.enabled=false  
security.basic.path=/cas/status/\*\*  
  
##  
# CAS Web Application Session Configuration  
#  
server.session.timeout=300  
server.session.cookie.http-only=true  
server.session.tracking-modes=COOKIE  
  
##  
# CAS Thymeleaf View Configuration  
#  
spring.thymeleaf.encoding=UTF-8  
spring.thymeleaf.cache=true  
spring.thymeleaf.mode=HTML  
##  
# CAS Log4j Configuration  
#  
# logging.config=file:/etc/cas/log4j2.xml  
server.context-parameters.isLog4jAutoInitializationDisabled=true  
  
##  
# CAS AspectJ Configuration  
#  
spring.aop.auto=true  
spring.aop.proxy-target-class=true  
  
##  
# CAS Authentication Credentials  
#  
#cas.authn.accept.users=casuser::Mellon

## 3、添加shiro依赖

<!--加了依赖即将支持三种校验方式，包括文件存储用户校验器、拒绝用户校验器、shiro校验器-->  
<dependency>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-support-generic</artifactId>  
 <version>${cas.version}</version>  
</dependency>

## 4、启动 sso-config 和 sso-server 开始验证shiro登录 （admin 123 登录成功）

# 2、Rest认证

## 解释：cas明确规定如下：

## 1、cas 服务端会通过post请求，并且把用户信息以”用户名:密码”进行Base64编码放在authorization请求头中

## 2、返回200状态码并且格式为{“@class”:”org.apereo.cas.authentication.principal.SimplePrincipal”,”id”:”casuser”,”attributes”:{}} 是成功的； 返回状态码403用户不可用；404账号不存在；423账户被锁定；428过期；其他登录失败

## 1、新建spring boot项目 sso-rest-client ，聚合

#### pom 如下

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.carl.auth</groupId>  
 <artifactId>sso-rest-client</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>sso-rest-client</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.10.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
  
</project>

## 2、新建controller

@RestController  
public class AuthUserController {  
 private static final Logger ***LOGGER*** = LoggerFactory.getLogger(AuthUserController.class);  
  
 @Autowired  
 private UserRepertory userRepertory;  
  
  
 /\*\*  
 \* 1. cas 服务端会通过post请求，并且把用户信息以"用户名:密码"进行Base64编码放在authorization请求头中  
 \* 2. 返回200状态码并且格式为{"@class":"org.apereo.cas.authentication.principal.SimplePrincipal","id":"casuser","attributes":{}} 是成功的  
 \* 2. 返回状态码403用户不可用；404账号不存在；423账户被锁定；428过期；其他登录失败  
 \*  
 \* @param httpHeaders  
 \* @return  
 \*/  
 @PostMapping("/login")  
 public Object login(@RequestHeader HttpHeaders httpHeaders) {  
 ***LOGGER***.info("Rest api login.");  
 ***LOGGER***.debug("request headers: {}", httpHeaders);  
 SysUser user = null;  
 try {  
 UserTemp userTemp = obtainUserFormHeader(httpHeaders);  
 //尝试查找用户库是否存在  
 user = userRepertory.getUser(userTemp.username);  
 if (user != null) {  
 if (!user.getPassword().equals(userTemp.password)) {  
 //密码不匹配  
 return new ResponseEntity(HttpStatus.***BAD\_REQUEST***);  
 }  
 if (user.isDisable()) {  
 //禁用 403  
 return new ResponseEntity(HttpStatus.***FORBIDDEN***);  
 }  
 if (user.isLocked()) {  
 //锁定 423  
 return new ResponseEntity(HttpStatus.***LOCKED***);  
 }  
 if (user.isExpired()) {  
 //过期 428  
 return new ResponseEntity(HttpStatus.***PRECONDITION\_REQUIRED***);  
 }  
 } else {  
 //不存在 404  
 return new ResponseEntity(HttpStatus.***NOT\_FOUND***);  
 }  
 } catch (UnsupportedEncodingException e) {  
 ***LOGGER***.error("", e);  
 new ResponseEntity(HttpStatus.***BAD\_REQUEST***);  
 }  
 ***LOGGER***.info("[{}] login is ok", user.getUsername());  
 //成功返回json  
 return user;  
 }  
  
 /\*\*  
 \* 根据请求头获取用户名及密码  
 \*  
 \* @param httpHeaders  
 \* @return  
 \* @throws UnsupportedEncodingException  
 \*/  
 private UserTemp obtainUserFormHeader(HttpHeaders httpHeaders) throws UnsupportedEncodingException {  
 /\*\*  
 \*  
 \* This allows the CAS server to reach to a remote REST endpoint via a POST for verification of credentials.  
 \* Credentials are passed via an Authorization header whose value is Basic XYZ where XYZ is a Base64 encoded version of the credentials.  
 \*/  
 //根据官方文档，当请求过来时，会通过把用户信息放在请求头authorization中，并且通过Basic认证方式加密  
 String authorization = httpHeaders.getFirst("authorization");//将得到 Basic Base64(用户名:密码)  
 String baseCredentials = authorization.split(" ")[1];  
 String usernamePassword = new String(Base64Utils.decodeFromString(baseCredentials), "UTF-8");//用户名:密码  
 ***LOGGER***.debug("login user: {}", usernamePassword);  
 String credentials[] = usernamePassword.split(":");  
 return new UserTemp(credentials[0], credentials[1]);  
 }  
  
 /\*\*  
 \* 解析请求过来的用户  
 \*/  
 private class UserTemp {  
 private String username;  
 private String password;  
  
 public UserTemp(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
 }  
}

## 3、实体类

### 1、解释：@JsonProperty @JsonProperty不仅仅是在序列化的时候有用，反序列化的时候也有用，比如有些接口返回的是json字符串，命名又不是标准的驼峰形式，在映射成对象的时候，将类的属性上加上@JsonProperty注解，里面写上返回的json串对应的名字

public class SysUser {  
 @JsonProperty("id")  
 @NotNull  
 private String username;  
 @JsonProperty("@class")  
 //需要返回实现org.apereo.cas.authentication.principal.Principal的类名接口  
 private String clazz = "org.apereo.cas.authentication.principal.SimplePrincipal";  
 @JsonProperty("attributes")  
 private Map<String, Object> attributes = new HashMap<>();  
  
 @JsonIgnore  
 @NotNull  
 private String password;  
  
 @JsonIgnore  
 //用户是否不可用  
 private boolean disable = false;  
 @JsonIgnore  
 //用户是否过期  
 private boolean expired = false;  
  
 @JsonIgnore  
 //是否锁定  
 private boolean locked = false;  
  
 public boolean isLocked() {  
 return locked;  
 }  
  
 public SysUser setLocked(boolean locked) {  
 this.locked = locked;  
 return this;  
 }  
  
 public boolean isDisable() {  
 return disable;  
 }  
  
 public SysUser setDisable(boolean disable) {  
 this.disable = disable;  
 return this;  
 }  
  
 public boolean isExpired() {  
 return expired;  
 }  
  
 public SysUser setExpired(boolean expired) {  
 this.expired = expired;  
 return this;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public SysUser setPassword(String password) {  
 this.password = password;  
 return this;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public SysUser setUsername(String username) {  
 this.username = username;  
 return this;  
 }  
  
 public String getClazz() {  
 return clazz;  
 }  
  
 public Map<String, Object> getAttributes() {  
 return attributes;  
 }  
  
 public SysUser setAttributes(Map<String, Object> attributes) {  
 this.attributes = attributes;  
 return this;  
 }  
  
 @JsonIgnore  
 public SysUser addAttribute(String key, Object val) {  
 getAttributes().put(key, val);  
 return this;  
 }  
}

## 4、创建service 用来获取用户数据

public class UserRepertory {  
 private Map<String, SysUser> users = new HashMap<>();  
  
 public UserRepertory(Map<String, SysUser> users) {  
 this.users = users;  
 }  
  
 public UserRepertory(SysUser ... users) {  
 for(SysUser user : users) {  
 this.users.put(user.getUsername(), user);  
 }  
 }  
  
 /\*\*  
 \* 根据id获取对应的用户数据  
 \*  
 \* @param id 用户id  
 \* @return  
 \*/  
 public SysUser getUser(String id) {  
 return users.get(id);  
 }  
}

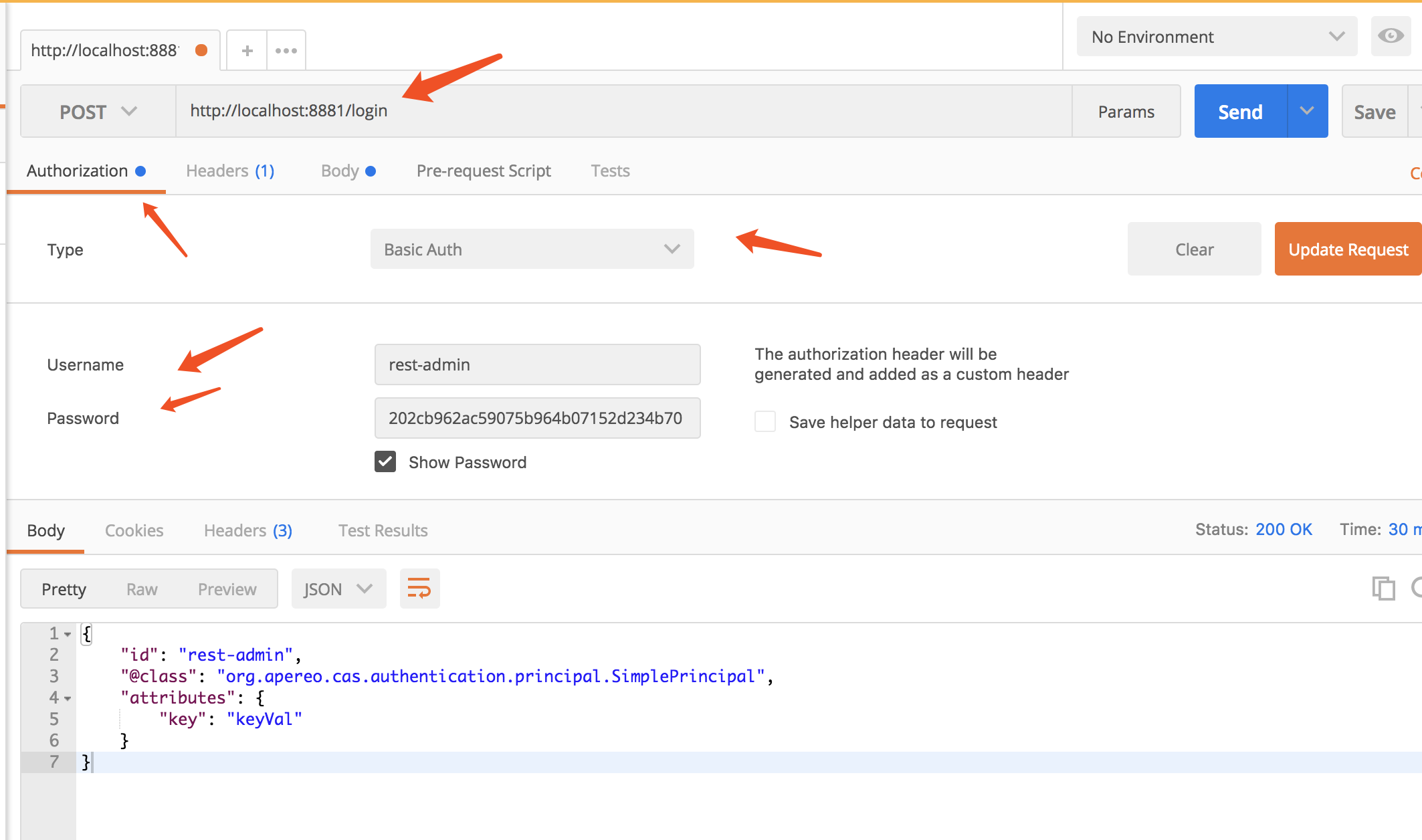
## 5、启动tomcat添加用户数据

@Configuration  
public class ApplicationConfig {  
 @Bean  
 public UserRepertory userRepertory() {  
 SysUser admin = new SysUser().setUsername("rest-admin").setPassword("202cb962ac59075b964b07152d234b70").addAttribute("key", "keyVal");  
 SysUser test = new SysUser().setUsername("rest-test").setPassword("202cb962ac59075b964b07152d234b70").addAttribute("test", "testVal");  
 SysUser locked = new SysUser().setUsername("rest-locked").setPassword("202cb962ac59075b964b07152d234b70").setLocked(true);  
 SysUser disable = new SysUser().setUsername("rest-disable").setPassword("202cb962ac59075b964b07152d234b70").setDisable(true);  
 SysUser expired = new SysUser().setUsername("rest-expired").setPassword("202cb962ac59075b964b07152d234b70").setExpired(true);  
 return new UserRepertory(admin, test, locked, disable, expired);  
 }  
}

## 6、启动tomcat项目 ，postman访问测试，是否成功

### 配置端口为8081

spring:  
 application:  
 name: auth rest client  
server:  
 port: 8881  
  
logging:  
 level:  
 com:  
 carl: DEBUG



{

"id": "rest-admin",

"@class": "org.apereo.cas.authentication.principal.SimplePrincipal",

"attributes": {

"key": "keyVal"

}

}

# 7、sso-config 中配置sso-dev.properties cas 使用rest登录

#REST 认证开始  
cas.authn.rest.uri=http://localhost:8881/login  
cas.authn.rest.passwordEncoder.type=DEFAULT  
cas.authn.rest.passwordEncoder.characterEncoding=UTF-8  
cas.authn.rest.passwordEncoder.encodingAlgorithm=MD5  
#REST 结束

# 8、sso-server 添加rest依赖

<!--restj认证-->  
<dependency>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-support-rest-authentication</artifactId>  
 <version>${cas.version}</version>  
</dependency>

# 9、开始启动sso-rest-client sso-config sso-server

## 1、浏览器中访问 <https://passport.sso.com:8443/cas/login>

### 1、使用锁定账户登录 rest-locked/123



## 2、正常访问成功 rest-admin/123

