cas

cas JDBC 登录验证 （接上https）

# 业务需求

在不同公司，可能有很多业务需求或者架构不一样导致我们实现验证的方式不一样，那么cas为我们提供了很多认证的模式（当然也可以[自定义](https://apereo.github.io/cas/5.1.x/installation/Configuring-Custom-Authentication.html)），其中常用的有：

* JDBC认证
* LDAP认证
* Basic认证
* Shiro认证
* Pac4j认证
* MongoDB认证
* Rest认证
* IP黑白名单

还有可能交给第三方认证，例如：微信、QQ、新浪等等

当然也有一些公司或者企业也非常的变态，如：

1. 认证中心不能直接访问账号库，cas也提供功能，可以考虑用[REST认证模块](https://apereo.github.io/cas/5.1.x/installation/Rest-Authentication.html)来处理这个事情
2. 老系统账号唯一性不确定，例如 **组织+账号** 才是唯一值，这时候只能自定义认证器（后面章节会有教程）
3. 业务系统要求返回用户密码（多属性返回/[ClearPass](https://apereo.github.io/cas/5.1.x/integration/ClearPass.html)）

# 1、MD5加密数据库验证（将原来系统提供的用户名和密码注释掉）

## 1、sso-server 添加数据库依赖

<dependencies>  
 <dependency>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-webapp-tomcat</artifactId>  
 <version>${cas.version}</version>  
 <type>war</type>  
 <scope>runtime</scope>  
 </dependency>  
  
  
  
 <!--新增支持jdbc验证-->  
 <dependency>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-support-jdbc</artifactId>  
 <version>${cas.version}</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-support-jdbc-drivers</artifactId>  
 <version>5.1.2</version>  
 </dependency>  
  
  
 <!-- mysql 数据库驱动. -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>6.0.6</version>  
 </dependency>  
  
 </dependencies>

## 2、创建数据 cas，创建表和插入测试数据（密码为md5加密）

| **字段名** | **字段类型** | **备注** |
| --- | --- | --- |
| username | varchar | 登录账号 |
| password | varchar | 密码 |
| expired | int | 过期字段，1为过期，需修改密码 |
| disable | int | 不可用字段，1为不可用，禁用 |
| email | varchar | 邮箱，可不需要 |

/\*  
账号表  
\*/  
CREATE TABLE SYS\_USER (  
 USERNAME VARCHAR(30) PRIMARY KEY,  
 PASSWORD VARCHAR(64) NOT NULL,  
 EMAIL VARCHAR(50),  
 ADDRESS VARCHAR(100),  
 AGE INT,  
 EXPIRED INT,  
 DISABLE INT  
);

---明文MD5数据  
  
/\*123\*/  
INSERT INTO SYS\_USER VALUES ('admin', '202cb962ac59075b964b07152d234b70', 'huang.wenbin@foxmail.com', '广州天河', 24, 0, 0);  
/\*12345678\*/  
INSERT INTO SYS\_USER VALUES ('zhangsan', '25d55ad283aa400af464c76d713c07ad', 'zhangsan@foxmail.com', '广州越秀', 26, 0, 0);  
/\*1234\*/  
/\*锁定用户\*/  
INSERT INTO SYS\_USER VALUES('zhaosi','81dc9bdb52d04dc20036dbd8313ed055', 'zhaosi@foxmail.com', '广州海珠', 25, 0 , 1);  
/\*12345\*/  
/\*不可用\*/  
INSERT INTO SYS\_USER VALUES('wangwu','827ccb0eea8a706c4c34a16891f84e7b', 'wangwu@foxmail.com', '广州番禺', 27, 1 , 0);

## 3、配置数据源 sso-config 中的sso-dev-properties中添加数据源信息（之类将代码全部贴出）

### 1、若密码无加密，调整passwordEncoder.type=NONE

### 2、若密码加密策略为SHA，调整passwordEncoder.encodingAlgorithm=SHA

##  
# CAS Server Context Configuration  
#  
server.context-path=/cas  
server.port=8443  
  
  
#33、3/33、、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  
  
  
#4.jdbc  
  
#Query Database Authentication 数据库查询校验用户名开始  
#查询账号密码sql，必须包含密码字段  
cas.authn.jdbc.query[0].sql=select \* from sys\_user where username=?  
#指定上面的sql查询字段名（必须）  
cas.authn.jdbc.query[0].fieldPassword=password  
#指定过期字段，1为过期，若过期不可用  
cas.authn.jdbc.query[0].fieldExpired=expired  
#为不可用字段段，1为不可用，需要修改密码  
cas.authn.jdbc.query[0].fieldDisabled=disabled  
#数据库方言hibernate的知识  
cas.authn.jdbc.query[0].dialect=org.hibernate.dialect.MySQL5Dialect  
#数据库驱动  
cas.authn.jdbc.query[0].driverClass=com.mysql.jdbc.Driver  
#数据库连接  
cas.authn.jdbc.query[0].url=jdbc:mysql://localhost:3306/cas?useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&useSSL=true  
#数据库用户名  
cas.authn.jdbc.query[0].user=root  
#数据库密码  
cas.authn.jdbc.query[0].password=123456  
#默认加密策略，通过encodingAlgorithm来指定算法，默认NONE不加密,可以自定义密码认证，比如ddkj就是写了一个自定义的认证  
cas.authn.jdbc.query[0].passwordEncoder.type=DEFAULT  
cas.authn.jdbc.query[0].passwordEncoder.characterEncoding=UTF-8  
cas.authn.jdbc.query[0].passwordEncoder.encodingAlgorithm=MD5  
#Query Database Authentication 数据库查询校验用户名结束  
  
  
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

## 4、开始登录测试 <https://passport.sso.com:8443/cas/login> 用户名和密码 admin/123 登录成功



# 2、加盐处理密码

## 1、加盐数据源配置

#4、加盐处理  
#Encode Database Authentication 开始  
#加密次数  
cas.authn.jdbc.encode[0].numberOfIterations=2  
#该列名的值可替代上面的值，但对密码加密时必须取该值进行处理  
cas.authn.jdbc.encode[0].numberOfIterationsFieldName=  
# 盐值固定列  
cas.authn.jdbc.encode[0].saltFieldName=username  
#静态盐值  
cas.authn.jdbc.encode[0].staticSalt=.  
cas.authn.jdbc.encode[0].sql=select \* from sys\_user\_encode where username=?  
#对处理盐值后的算法  
cas.authn.jdbc.encode[0].algorithmName=MD5  
cas.authn.jdbc.encode[0].passwordFieldName=password  
cas.authn.jdbc.encode[0].expiredFieldName=expired  
cas.authn.jdbc.encode[0].disabledFieldName=disabled  
#数据库连接  
cas.authn.jdbc.encode[0].url=jdbc:mysql://localhost:3306/cas?useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&useSSL=true  
cas.authn.jdbc.encode[0].dialect=org.hibernate.dialect.MySQL5Dialect  
cas.authn.jdbc.encode[0].driverClass=com.mysql.jdbc.Driver  
cas.authn.jdbc.encode[0].user=root  
cas.authn.jdbc.encode[0].password=123456

## 2、测试产生盐

/\*\*  
 \* @author Carl  
 \* @date 2017/9/12  
 \* @since JDK1.7  
 \*/  
public class PasswordSaltTest {  
 private String staticSalt = ".";  
 private String algorithmName = "MD5";  
 private String encodedPassword = "123";  
 private String dynaSalt = "admin\_en";  
 @Test  
 public void test() throws Exception {  
 ConfigurableHashService hashService = new DefaultHashService();  
 hashService.setPrivateSalt(ByteSource.Util.bytes(this.staticSalt));  
 hashService.setHashAlgorithmName(this.algorithmName);  
 hashService.setHashIterations(2);  
 HashRequest request = new HashRequest.Builder()  
 .setSalt(dynaSalt)  
 .setSource(encodedPassword)  
 .build();  
 String res = hashService.computeHash(request).toHex();  
 System.out.println(res);  
 }  
}

|  |
| --- |
| //获得随机盐 |
|  | SecureRandomNumberGenerator secureRandomNumberGenerator = new SecureRandomNumberGenerator(); |
|  | String salt = secureRandomNumberGenerator.nextBytes().toHex(); |
|  | //对密码加密后,将加密后的密码和盐存入对象 |
|  | String password = companyInfo.getPwd(); |
|  | password = new Md5Hash(password, salt).toString(); |

## 3、数据库表创建 和测试数据

/\*  
账号加盐表  
\*/  
CREATE TABLE SYS\_USER\_ENCODE (  
 USERNAME VARCHAR(30) PRIMARY KEY,  
 PASSWORD VARCHAR(64) NOT NULL,  
 EMAIL VARCHAR(50),  
 ADDRESS VARCHAR(100),  
 AGE INT,  
 EXPIRED INT,  
 DISABLED INT  
);;

---加盐数据  
/\*123 可以采用PasswordSaltTest输出值\*/  
INSERT INTO SYS\_USER\_ENCODE VALUES ('admin\_en', 'bfb194d5bd84a5fc77c1d303aefd36c3', 'huang.wenbin@foxmail.com', '江门蓬江', 24, 0, 0);  
INSERT INTO SYS\_USER\_ENCODE VALUES ('zhangsan\_en', '68ae075edf004353a0403ee681e45056', 'zhangsan@foxmail.com', '深圳宝安', 21, 0, 0);  
INSERT INTO SYS\_USER\_ENCODE VALUES ('zhaosi\_en', 'd66108d0409f68af538301b637f13a18', 'zhaosi@foxmail.com', '清远清新', 20, 0, 1);  
INSERT INTO SYS\_USER\_ENCODE VALUES ('wangwu\_en', '44b907d6fee23a552348eabf5fcf1ac7', 'wangwu@foxmail.com', '佛山顺德', 19, 1, 0);

## 4、测试登录 成功，用户名 （admin\_en/123）

