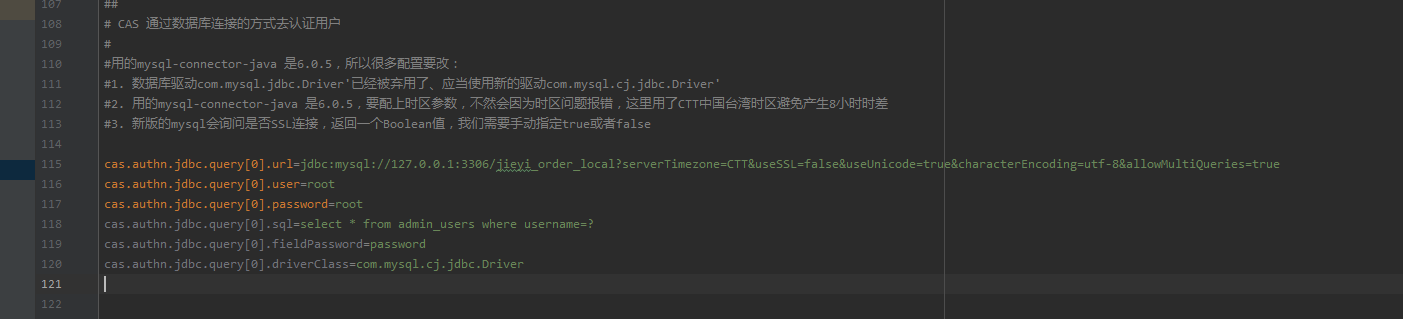
# 简单对接

先导入java的mysql依赖（略）

修改：application.properties

##  
# 添加认证服务  
#  
cas.serviceRegistry.initFromJson=true

# 注释掉写死的用户认证，采用数据库认证的方式  
# cas.authn.accept.users=darbuly::123456



# 高级对接

1. 通过重写passwordEncoder实体类加密逻辑

<https://www.cnblogs.com/gaopengpy/p/12371249.html>

官方说到可以指定一个 passwordEncoder 实体类做密码校验方法。

<https://apereo.github.io/cas/5.2.x/installation/Configuration-Properties.html#jdbc-password-management>

类中的matches方法就是密码认证方法

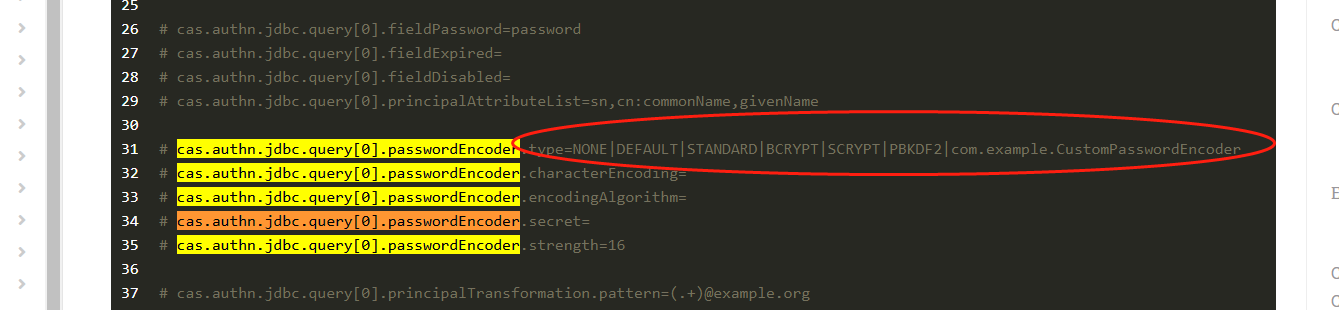
如需采用跟php 的 password\_hash 一致的验证方式，就采用github中这个库就行了

<https://github.com/patrickfav/bcrypt>

import at.favre.lib.crypto.bcrypt.BCrypt;

*/\*\*  
 \* 调用这个方法来判断密码是否匹配  
 \*/*@Override  
public boolean matches(CharSequence rawPassword, String encodePassword) {  
 // 判断密码是否存在  
 if (rawPassword == null) {  
 return false;  
 }  
 //与 php 的 password\_hash() 方法逻辑一致  
 BCrypt.*with*(BCrypt.Version.*VERSION\_2Y*);  
 BCrypt.Result res = BCrypt.*verifyer*().verify(rawPassword.toString().toCharArray(), encodePassword);  
  
 return res.verified;  
}

然后写到配置文件里去：



#默认加密策略，通过encodingAlgorithm来指定算法，默认NONE不加密  
cas.authn.jdbc.query[0].passwordEncoder.type=com.run.cas.CustomPasswordEncoder  
cas.authn.jdbc.query[0].passwordEncoder.characterEncoding=UTF-8  
cas.authn.jdbc.query[0].passwordEncoder.encodingAlgorithm=SHA-256