[**#**](https://pig4cloud.com/#jasypt-%E7%9A%84%E8%A7%A3%E5%86%B3%E6%96%B9%E6%A1%88) **jasypt 的解决方案**

1. Maven依赖

<dependency>

<groupId>com.github.ulisesbocchio</groupId>

<artifactId>jasypt-spring-boot-starter</artifactId>

<version>1.16</version>

</dependency>

1. 配置

jasypt:

encryptor:

password: foo #根密码

3 调用JAVA API 生成密文

@RunWith(SpringJUnit4ClassRunner.class)

@SpringBootTest(classes = PigAdminApplication.class)

public class PigAdminApplicationTest {

@Autowired

private StringEncryptor stringEncryptor;

@Test

public void testEnvironmentProperties() {

System.out.println(stringEncryptor.encrypt("lengleng"));

}

}

或者直接使用JAVA 方法调用 （不依赖 spring 容器）

/\*\*

\* jasypt.encryptor.password 对应 配置中心 application-dev.yml 中的密码

\*/

@Test

public void testEnvironmentProperties() {

System.setProperty(JASYPT\_ENCRYPTOR\_PASSWORD, "lengleng");

StringEncryptor stringEncryptor = new DefaultLazyEncryptor(new StandardEnvironment());

//加密方法

System.out.println(stringEncryptor.encrypt("123456"));

//解密方法

System.out.println(stringEncryptor.decrypt("saRv7ZnXsNAfsl3AL9OpCQ=="));

}

4 配置文件中使用密文

spring:

datasource:

password: ENC(密文)

xxx: ENC(密文)

5 [其他非对称等高级配置参考](https://github.com/ulisesbocchio/jasypt-spring-boot)

[**#**](https://pig4cloud.com/#%E6%80%BB%E7%BB%93) **总结**

1. Spring Cloud Config 提供了统一的加解密方式，方便使用，但是如果应用配置没有走配置中心，那么加解密过滤是无效的；依赖JCE 对于低版本spring cloud的兼容性不好。
2. jasypt 功能更为强大，支持的加密方式更多，但是如果多个微服务，需要每个服务模块引入依赖配置，较为麻烦；但是功能强大 、灵活。
3. 个人选择 jasypt