

| 学习目标

- * 能够使用C3P0数据库连接池
 - * 导入jar，配置文件，使用某个数据源
 - * 参数了解清楚一点
- * 能够使用Druid数据库连接池
 - * 导入jar，配置文件，使用某个数据源
 - * 参数了解清楚一点
- * 能够了解常见加密算法
 - * 单向加密：MD5，SHA
 - * 对称加密：DES
 - * 非对称加密：RSA
- * 能够给Druid数据源配置文件密码加密
 - * ConfigTools:对私钥进行加密，对公钥进行解密
 - * 配置加密密码
 - * filters
 - * connectionProperties

* 回顾

- * 代理模式
 - * 用户，代理，目标对象（被代理）
 - * 提供额外的功能
- * 分类：静态代理，动态代理（JDK代理（接口代理），子类代理（CGLib代理））
- * 静态代理
 - * 代理对象给目标对象额外增加功能
 - * 代理对象和目标对象有共同接口
 - * 代理对象持有目标对象引用
 - * 运行前，提交准备好的（手工创建代理类或者有其他编译器自动创建代理类）

* 静态代理类：导致代理类很多，增加工作量，提供维护难度

* 动态代理--JDK代理

* 在内存中生成代理类

* API

Object proxy=

```
Proxy.newInstanceProxy(target.getClass().getClassLoader().target.getClass().getInterfaces(),
```

```
    new InvocationHandler(){
```

```
        @Override
```

```
        public Object invoke(Object obj,Method method,Object[] args){
```

```
            // 前后或者出现异常的时候，额外添加的功能
```

```
            method.invoke(target,args);
```

```
        }
```

```
    });
```

* 动态代理--子类代理

* 在内存中生成目标对象子类

* API

* 引入jar：cglib，asm

```
Enhancer enhancer=new Enhancer();
```

```
enhancer.setSuperClass(target.getClass());
```

```
enhancer.setCallback(
```

```
    new MethodInterceptor(){
```

```
        @Override
```

```
        public Object intercept(Object obj,Method method,Object[] args,MethodProxy
```

```
proxy){
```

```
            method.invoke(target,args);
```

```
        }
```

```
    }
```

);

Object proxy=enhancer.create();

* 自定义连接池

* 池：LinkedList

* Connection:close--->关闭数据库链接--->放到池

* 动态代理：代理是Connection的子类

* 能够使用C3P0数据库连接池

* C3P0概述

* C3P0是一个开源的JDBC连接池，它实现了数据源和JNDI绑定，支持JDBC3规范和JDBC2的标准扩展。目前使用它的开源项目有Hibernate、Spring等

```
1 * 使用C3P0
2 * 导入jar包
3     * c3p0-0.9.5.4.jar
4     * c3p0-oracle-thin-extras-0.9.5.4.jar
5     * mchange-commons-java-0.2.15.jar
6 * copy配置文件c3p0-config.xml
7 <?xml version="1.0" encoding="UTF-8"?>
8 <c3p0-config>
9     <!-- C3P0的缺省(默认)配置,
10         如果在代码中“ComboPooledDataSource ds =
11         new ComboPooledDataSource();”
12         这样写就表示使用的是C3P0的缺省(默认)配置信息来创建数据源 -->
13     <default-config>
14         <property name="driverClass">oracle.jdbc.driver.OracleDriver</property>
15         <property name="jdbcUrl">jdbc:oracle:thin:@192.168.1.121:1521/orcl</prc
16         <property name="user">scott</property>
17         <property name="password">tiger</property>
18         <property name="acquireIncrement">5</property>
19         <property name="initialPoolSize">10</property>
20         <property name="minPoolSize">5</property>
21         <property name="maxPoolSize">20</property>
22     </default-config>
23
24     <!-- C3P0的命名配置,
25         如果在代码中“ComboPooledDataSource ds =
```

```

26     new ComboPooledDataSource("MySQL");
27     "这样写就表示使用的是name是MySQL的配置信息来创建数据源 -->
28     <named-config name="MySQL">
29         <property name="driverClass">com.mysql.jdbc.Driver</property>
30         <property name="jdbcUrl">jdbc:mysql://127.0.0.1:3306/test</property>
31         <property name="user">root</property>
32         <property name="password">root</property>
33         <property name="acquireIncrement">5</property>
34         <property name="initialPoolSize">10</property>
35         <property name="minPoolSize">5</property>
36         <property name="maxPoolSize">20</property>
37     </named-config>
38 </c3p0-config>
39
40 * 链接工具类
41 public class ConnectionUtilsC3P0 {
42     public static DataSource dataSource=new ComboPooledDataSource();
43     // 获得链接
44     public static Connection getConn() {
45         try {
46             return dataSource.getConnection();
47         } catch (SQLException e) {
48             e.printStackTrace();
49         }
50         return null;
51     }
52
53     // 释放资源
54     public static void close(Connection con,Statement st,ResultSet rs) {
55         if(rs!=null) {
56             try {
57                 rs.close();
58             } catch (SQLException e) {
59                 e.printStackTrace();
60             }
61         }
62         if(st!=null) {
63             try {
64                 st.close();
65             } catch (SQLException e) {

```

```

66         e.printStackTrace();
67     }
68 }
69 if(con!=null) {
70     try {
71         con.close();
72     } catch (SQLException e) {
73         e.printStackTrace();
74     }
75 }
76 }
77
78 // 关闭连接
79 public static void closeConn(Connection conn) {
80     close(conn, null, null);
81 }
82
83 // 关闭SQL执行对象
84 public static void closeSt(Statement st) {
85     close(null, st, null);
86 }
87
88 //关闭结果集
89 public static void closeRs(ResultSet rs) {
90     close(null, null, rs);
91 }
92
93 }
94 * UserDaoImpl更改获取链接方式
95 * Connection conn = ConnectionUtilsC3P0.getConn();
96 * 单元测试
97 @Test
98     public void test1() throws SQLException {
99         UserDao userDao=new UserDaoImpl();
100         User user=new User();
101         user.setName("刘备");
102         user.setPassword(MD5Utils.md5("123"));
103         userDao.add(user);
104     }
105 * 假如需要日志的显示

```

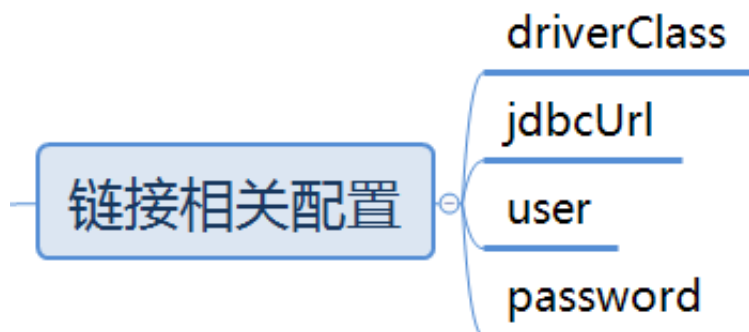
106 * 添加log4j的jar
107 * log4j.properties 配置文件
108

* c3p0 常见配置参数

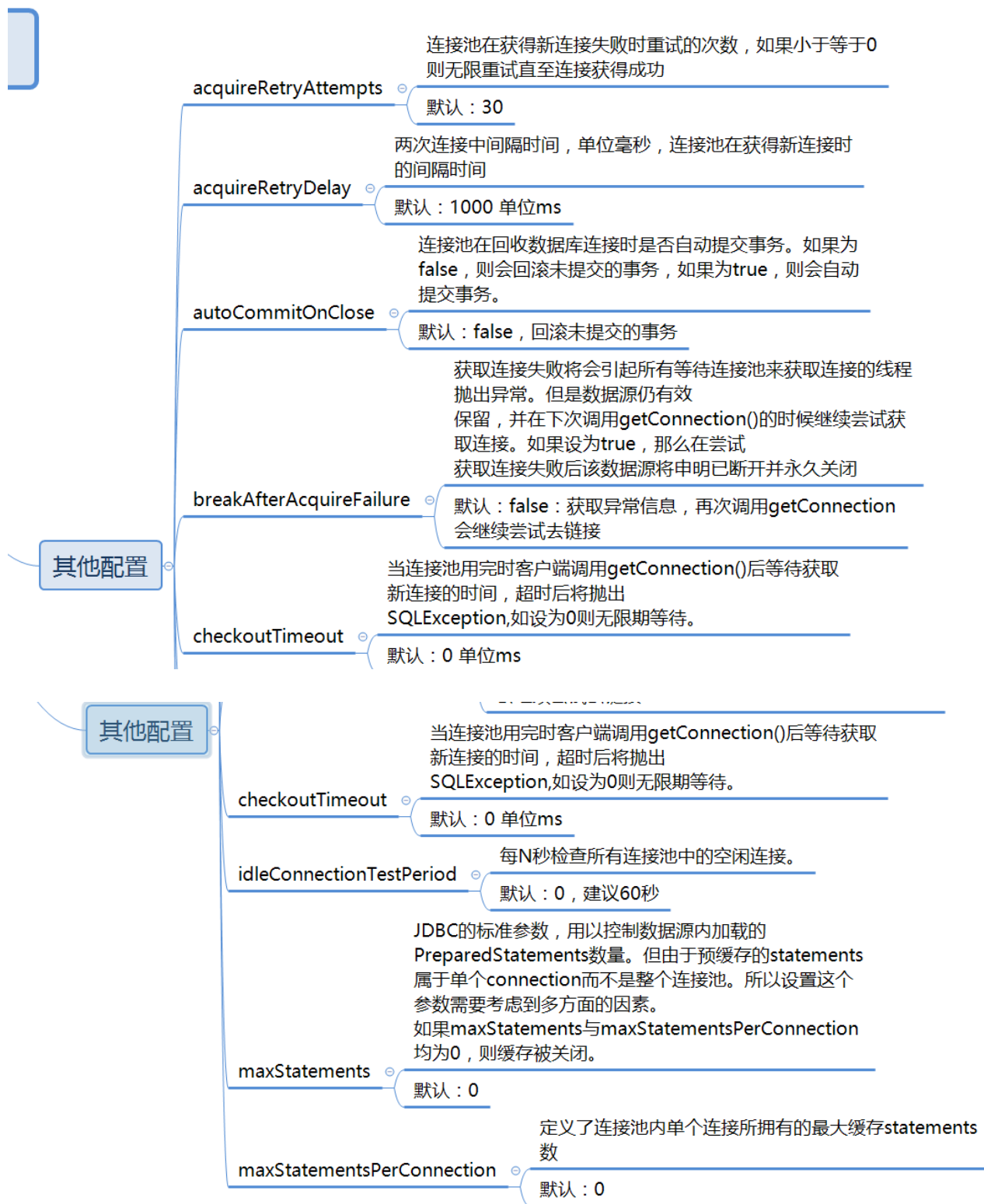
* 基础配置



* 数据库链接配置



* 其他配置



* 能够使用Druid数据库连接池

* Druid 概述

* 阿里巴巴数据库事业部出品，为监控而生的数据库连接池。

* 例如阿里云，也是用druid

* Druid 是一个 JDBC 组件库，包含数据库连接池、SQL Parser 等组件，

* 被大量业务和技术产品使用或集成，经历过最严苛线上业务场景考验，是你值得信赖的技术产品。

```
1 * 使用Druid
2 * 导入jar包
3     * druid-1.0.16.jar
4 * copy配置文件druid.properties
5     driverClassName=oracle.jdbc.driver.OracleDriver
6     url=jdbc:oracle:thin:@192.168.1.121:1521/orcl
7     username=scott
8     password=tiger
9     initialSize=5
10    timeBetweenEvictionRunsMillis=60000
11    minEvictableIdleTimeMillis=300000
12    validationQuery=SELECT 1 from dual
13
14 * ConnectionUtilsDruid 编写
15 public class ConnectionUtilsDruid {
16     public static DruidDataSource dataSource;
17     static {
18         InputStream is = ConnectionUtilsDruid.class.getClassLoader().getResourceAsStream(
19             "druid.properties");
20         Properties props=new Properties();
21         try {
22             props.load(is);
23             dataSource=(DruidDataSource) DruidDataSourceFactory.createDataSource(props);
24         } catch (IOException e) {
25             e.printStackTrace();
26         } catch (Exception e) {
27             e.printStackTrace();
28         }
29     }
30     // 获得链接
31     public static Connection getConn() {
32         try {
33             return dataSource.getConnection();
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37     }
38 }
```



```
37         return null;
38     }
39
40     // 释放资源
41     public static void close(Connection con,Statement st,ResultSet rs) {
42         if(rs!=null) {
43             try {
44                 rs.close();
45             } catch (SQLException e) {
46                 e.printStackTrace();
47             }
48         }
49         if(st!=null) {
50             try {
51                 st.close();
52             } catch (SQLException e) {
53                 e.printStackTrace();
54             }
55         }
56         if(con!=null) {
57             try {
58                 con.close();
59             } catch (SQLException e) {
60                 e.printStackTrace();
61             }
62         }
63     }
64
65     // 关闭连接
66     public static void closeConn(Connection conn) {
67         close(conn, null, null);
68     }
69
70     // 关闭SQL执行对象
71     public static void closeSt(Statement st) {
72         close(null, st, null);
73     }
74
75     //关闭结果集
76     public static void closeRs(ResultSet rs) {
```

```

77         close(null, null, rs);
78     }
79
80 }
81 * UserDaoImpl更改获取链接方式
82 * Connection conn = ConnectionUtilsDruid.getConnection();
83 * 使用单元测试
84 * 假如需要日志的显示
85     * 添加log4j的jar
86     * log4j.properties 配置文件

```

* Druid数据库连接池配置

配置这个属性的意义在于，如果存在多个数据源，监控的时候可以通过名字来区分开来。

name ⊖ 默认："DataSource-"
+System.identityHashCode(this)

initialSize ⊖ 初始化时建立物理连接的个数。初始化发生在显示调用init方法

默认：0

maxActive ⊖ 最大连接池数量

默认：8

minIdle ⊖ 最小空闲连接数

maxWait ⊖ 获取连接时最大等待时间，单位毫秒

是否缓存preparedStatement，也就是PSCache。
PSCache对支持游标的数据库性能提升巨大，比如说oracle。

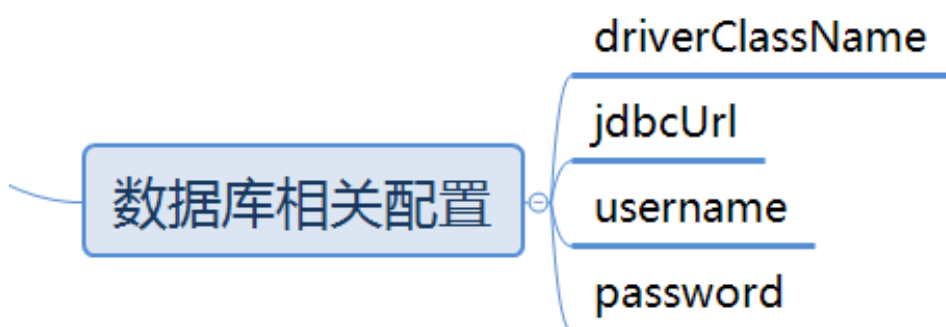
poolPreparedStatements ⊖ 在mysql5.5以下的版本中没有PSCache功能，建议关闭掉。5.5及以上版本有PSCache，建议开启。

默认值：false

要启用PSCache，必须配置大于0，当大于0时，
poolPreparedStatements自动触发修改为true。
在Druid中，不会存在Oracle下PSCache占用内存过多的问题，

maxOpenPreparedStatements ⊖ 可以把这个数值配置大一些，比如说100

配置



* 能够了解常见加密算法

* 参考Java安全技术研究_change.pptx

- 1 * 加密分类
- 2 * 单向加密：MD5，SHA
- 3 * MD5Util
- 4 * SHAUtil
- 5 * 对称加密：DES
- 6 * DesUtil
- 7 * 非对称加密：RSA

```
8      * RSAUtil
9
10 * MD5Util测试
11 @Test
12 public void test1() {
13     String md5=MD5Util.md5("123");
14     System.out.println(md5);
15 }
16 结果:
17 202cb962ac59075b964b07152d234b70
18
19 * SHAUtil测试
20 @Test
21 public void test2() {
22     String sha = SHAUtil.SHAEncode("123");
23     System.out.println(sha);
24 }
25 * 结果
26 40bd001563085fc35165329ea1ff5c5ecbdbbeef
27
28 * DesUtil 测试
29 @Test
30 public void test3() throws Exception {
31     // 加密
32     String key="abc123de";
33     String data="亮哥教育";
34     String encrypt = DesUtil.encrypt(data, key);
35     System.out.println(encrypt);
36
37     // 解密
38     String decrypt = DesUtil.decrypt(encrypt, key);
39     System.out.println(decrypt);
40 }
41 结果:
42 9mRRlpow4FARsL3gI1hMbw==
43 亮哥教育
44
45 * RSAUtil测试
46 @Test
47 public void test5() throws Exception {
```

```

48 // 产生一对密钥
49 Map<String, Object> genKeyPair = RSAUtil.genKeyPair();
50 // 获取公钥
51 String publicKey = RSAUtil.getPublicKey(genKeyPair);
52 // 获取私钥
53 String privateKey = RSAUtil.getPrivateKey(genKeyPair);
54 System.out.println("publicKey:");
55 System.out.println(publicKey);
56 System.out.println("privateKey:");
57 System.out.println(privateKey);
58 // 用公钥加密，私钥解密
59 byte[] data="亮哥教育".getBytes();
60 byte[] encryptByPublicKey = RSAUtil.encryptByPublicKey(data, publicKey);
61 System.out.println("公钥加密后的数据:");
62 System.out.println(new String(encryptByPublicKey));
63 byte[] decryptByPrivateKey = RSAUtil.decryptByPrivateKey(encryptByPublicKey);
64 System.out.println("私钥解密后的数据:");
65 System.out.println(new String(decryptByPrivateKey));
66 System.out.println("-----");
67 // 用私钥加密，公钥解密
68 byte[] encryptByPrivateKey = RSAUtil.encryptByPrivateKey(data, privateKey);
69 System.out.println("私钥加密后的数据:");
70 System.out.println(new String(encryptByPrivateKey));
71 byte[] decryptByPublicKey = RSAUtil.decryptByPublicKey(encryptByPrivateKey);
72 System.out.println("公钥解密后的数据:");
73 System.out.println(new String(decryptByPublicKey));
74 }
75 * 结果
76 publicKey:
77 MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDbEU8AU//S1mZPLWNoksgKBsJlepEqHtro/Wjd
78 QY42c2T3rLXJ1J4YFcTWFks8hEyTGxSZ2BAJJH2UvjN690ZYNq+Brmn41QFLNXMAWpDE1jqZtRgc
79 r7jvjfkLRmH+g5M3FD6C7gV0yCNcAg7cHTPyPufSEoqUNg0aMEvTJGpy3wIDAQAB
80 privateKey:
81 MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwgGJdAgEAAoGBANsRTwBT/9LWZk8tY2iSyAoGwmV6
82 kSoe2uj9aN1BjjZzPestcnUnhgUK1Z+SzyETJMBfJnYEAKkfZS+M3r3Rlg2r4GuafjVAUs1cwBa
83 kMTW0pm1GBYvu0+N+QtGYf6DkzcUPoLuBXTII1wCDtdwM/I+59ISipQ2DRowS9MkanLfAgMBAAEC
84 gYBBh4fuPTJFS0UHYjheS6ny9dqqGVMCDbLgyIfLUDxIMuPGua2HRe9dCsSkGzJCXoududOrcb+
85 NSArOkmb+uPbAbH4jJJU+k6g1U0fJ77/WyRhic5r055aecU9mvg84ZScdFUoE3m0+c4np/RlAjMv
86 q43+pHi0AyRUVHgSIBIoQQJBA048YoGfHxUMaFtZ5zmdew7s/ZRVN1eGXgFVEk5E8NmWRljyDqNu
87 i/2ipiiNn01Zqra9cdTtDApBUsv2Z4fCbTkCQQDrZwa/lDAGonjD33pTRqWPqFT01TduHX2lgfno

```

```

88 GxIHDY1zxN1N2+D21bqF7eymlcaiXY1r2eiLltICrGHYXnjXAkEArte+a+c+RpJenHy0yIIcNx7T
89 VGPeQ+wnXc7zqmHKadmHJ4wu1h8xuqn+TFD0Ey61rXwH6P84EOa00puVi/tTcQJBALJV1b40j61H
90 ErhJKFHIW3dtLhdhJCIOW25MXnfhcjHXpocG3R1n8zDsT+dAjSTSm03OcNo89jgx5R3TXEySF3cC
91 QGEQWk2MVX0TRZUQCUNAURAz14h5yfny7oUEgcfSvSt4kCCWFCPjkaxtsYflsK0iOK18tW9rZiJP
92 dH8bYDIEZM8=
93 公钥加密后的数据:
94 ••禡•蜥•?, 侏龠&•u?8?•銳檢擲掟U/ W?•吉•跣%0, 筵mY?•?•?2沃&, ?••??{擗?••?••+咧鉦f•□
95 私钥解密后的数据:
96 亮哥教育
97 -----
98 私钥加密后的数据:
99 [K?•蔬'r]•苙U•NI••?"o?•鳩E亮>$穗柁Ka癢渤*w冥•J□+;C•□2g*?•傒梓龍塔o跟A耿g敕O>?爆
100 公钥解密后的数据:
101 亮哥教育
102

```

* 能够给Druid数据源配置文件密码加密

```

1  * Druid 加密的工具类ConfigTools测试
2  * 注意文件的编码: UTF-8, 否则无法加密中文
3  * 提供了私钥加密, 公钥解密, 没有提供公钥加密, 私钥解密的方法
4  @Test
5      public void test6() throws Exception {
6          String[] genKeyPair2 = ConfigTools.genKeyPair(1024);
7          // 获取私钥
8          String privateKey = genKeyPair2[0];
9          // 获取公钥
10         String publicKey = genKeyPair2[1];
11         System.out.println("publicKey:");
12         System.out.println(publicKey);
13         System.out.println("privateKey:");
14         System.out.println(privateKey);
15         // 用私钥加密, 公钥解密
16         String data="tiger";
17         String encrypt = ConfigTools.encrypt(privateKey, data);
18         System.out.println("私钥加密后的数据:");
19         System.out.println(encrypt);

```

```
20     String decrypt=ConfigTools.decrypt(publicKey, encrypt);
21     System.out.println("公钥解密后的数据:");
22     System.out.println(decrypt);
23 }
24 结果:
25 publicKey:
26 MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCFIAncXk6he3vSKS42mnLqgoWwOEvM6K4jxHKm2kA
27 privateKey:
28 MIICdgIBADANBgkqhkiG9w0BAQEFAASCAMAwggJcAgEAAoGBAJ8gCdxetqF7e9IpLjaacuqChZY4S8z
29 私钥加密后的数据:
30 a3XspqUwY5b0Ya4yoYJAK89m/enZfv4MerDoLM7Zpapw9XwbzNHnd0EGxB12r7piuInSNfG39q5sDg+
31 公钥解密后的数据:
32 tiger
33
34 * 温馨提示:
35     * 记得保存好自己的私钥
36 * 修改配置文件
37 driverClassName=oracle.jdbc.driver.OracleDriver
38 url=jdbc:oracle:thin:@192.168.1.121:1521/orcl
39 username=scott
40 password=a3XspqUwY5b0Ya4yoYJAK89m/enZfv4MerDoLM7Zpapw9XwbzNHnd0EGxB12r7piuInSNf
41 #url=jdbc:mysql://localhost:3306/test
42 #username=root
43 #password=root
44 initialSize=5
45 timeBetweenEvictionRunsMillis=60000
46 minEvictableIdleTimeMillis=300000
47 validationQuery=SELECT 1 from dual
48 filters=config
49 connectionProperties=config.decrypt=true;config.decrypt.key=MIGfMA0GCSqGSIb3DQE
50
51 * 测试
52     * 可以使用
53
```

