Des加密算法

> des加密算法，是一个对称的加密算法，目前被广泛应用，所以打算写一个demo。

```java
package com.dao;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import java.io.IOException;
import java.security.SecureRandom;

/**
 * Created by linSir on 2017/6/22.des加密算法
 */
public class Test {

    public static void main(String args[]) throws IOException {


        //加密
        //byte[] result = Test.encrypt(str.getBytes(),password);
        //BASE64Encoder base64encoder = new BASE64Encoder();
        //String encode=base64encoder.encode(result);

        String miwen = "ZraEmkLPeVT1CBGRpcbXTfVRhUWt6riMMh8UoWcVEClwLcCRuJoMmZW+IS5MYshasXVUu1VIFeqE\
                "ySjMvDvu4z6GxUR7BVq95mfILIT6kvCLW1rvgJoZGlkXzDW7R+n8R/POzu61cfKejnMnW0HiRmsK\n" +
                "CNLB3zf0KYfB5H0x0+GUtXQmtQyG0x5tQSyHSWOdQVyEj7mYFw4h6uFhN94ifgZq8ohpUduWZBgU\n" +
                "EN3B4akKt8+oPQPFv1GvrFucOmrfDpyTy+YuLZZ0nlPA5AYTa2TnC++ZPPo62XW4O2EZ0qGXcuO1\n" +
                "3zHfq8mmtdQ7DbGN2JIBNLL/EN97o7pHRkVNbB9/eHElf37MghHZWTUfIlvRtSTwaWkW3IR2aWzj\n" +
                "GQXrdqErVUdcTvLH2fGnInYU6XAtwJG4mgyZG6OZZ89Yg9iOcWG4GruJvFEa/UQNDmbS+vyvWpP/\n" +
                "75zOiDos5s5yeJUcUaJt+SkUR7z5yr7bbK/DHkS5aEvfNI/nL4Z4DrGN++9Uzv34XD4ZTg0csEuL\n" +
                "96+LAUKED43iaJUo6wruiZ/7KmpvP5p3ii5p03Z1ymscmTlqUTZ55YFBCz3dZg8OSGIlKj+7uaYF\n" +
                "umweL38ksAtVL1wjgWMVF+9oYUie/jf6+mAdmvwiACoGu71ZziWc4tz1UPb27Qx4Qf0h/nItAkuT\n" +
                "yLK6+Hx0+GQ2weK2q95kgf8zUs1igyhu1VdMGHbp/Ma3DyJIo6wPgwRlpFedCq0/w7ECGGPHfLUb\n" +
                "eNmBK3nCqqN7TABiLfHfzR8mBjjMcJQ1MtGGwZB6H6zAGkcSEQqHgsTbnG6t8GvS06t9eepMn6VG\n" +
                "7X+dS4LUS9LpIZ/OgNwxvyxd3vw/dKn9u0OLgvJRGv1EDQ5m0t80qIo6RxHCLmnTdnXKaxxFIhNG\n" +
                "caq19CPEthsSFziTHlX/qM5g/yCwbN9+qClQ0z5VI/ZGUAcs9Cz3WjimPGKNyLa+AKgUE7dh4sFr\n" +
                "vQGrlRxRom35KRRRd/VE9Goz3EAcQQ1NhiDMYobeoH0as5XkG3hTF2zZyfn/QJZnNwh4GxCLkZPS\n" +
                "VKFdg0bpAy3irJouw+IG69DUewM4W4a1u7h8i76pCLLxP5gIYqKqKgm97itSQe8ZV3qbG9gNxMrg\n" +
                "aUQ+fCE3TvsP07RdLW9Dn6Mazxnq7wnw9X2Qj9+sTl+hLLKhL+ZlHIJk5wvvmTz6OikBCmYmdEZb\n" +
                "Q1Prg0CgHVrFy4JOc9rTCmLnieHBG/xVwI5AOp42NUOk/Ycc7EIuzQ4tKEGS7RjmcpKEUMkog5c5\n" +
                "k693mGsn2VUQNeRPmfrcN7Ra+L18fvKMs1ESEjrUR/GpHwg6UcCBfBh8r/B5bYdoV2ik02liSVzX\n" +
                "Kt5vzA3ZjC5mvkF/RJJUoCUa6j3xqznjJhmABsN23gOcRh8RRWb8VGI2xD8ErYwuZr5Bf0SSTyTL\n" +
                "p6dacHJIz1u6+TEr9OfinHoMzBwXETOkAnPOt/YdwGw8/MM41w==";

        BASE64Decoder base64decoder = new BASE64Decoder();
        byte[] encodeByte = base64decoder.decodeBuffer(miwen);


        //直接将如上内容解密
        try {
            byte[] decryResult = Test.decrypt(encodeByte, "");
            System.out.println("解密后: " + new String(decryResult));
        } catch (Exception e1) {
            e1.printStackTrace();
        }

    }

    /**
     * 加密
     */
    public static byte[] encrypt(byte[] datasource, String password) {
        try {
            SecureRandom random = new SecureRandom();
            DESKeySpec desKey = new DESKeySpec(password.getBytes());
            //创建一个密匙工厂，然后用它把DESKeySpec转换成
            SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
            SecretKey securekey = keyFactory.generateSecret(desKey);
            //Cipher对象实际完成加密操作
            Cipher cipher = Cipher.getInstance("DES");
            //用密匙初始化Cipher对象
            //用密匙初始化Cipher对象
            cipher.init(Cipher.ENCRYPT_MODE, securekey, random);
            //现在，获取数据并加密
            //正式执行加密操作
            return cipher.doFinal(datasource);
        } catch (Throwable e) {
            e.printStackTrace();
        }
        return null;
    }

    /*
     * 解密
     */
    private static byte[] decrypt(byte[] src, String password) throws Exception {
        // DES算法要求有一个可信任的随机数源
        SecureRandom random = new SecureRandom();
        // 创建一个DESKeySpec对象
        DESKeySpec desKey = new DESKeySpec(password.getBytes());
        // 创建一个密匙工厂
        SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
        // 将DESKeySpec对象转换成SecretKey对象
        SecretKey securekey = keyFactory.generateSecret(desKey);
        // Cipher对象实际完成解密操作
        Cipher cipher = Cipher.getInstance("DES");
        // 用密匙初始化Cipher对象
        cipher.init(Cipher.DECRYPT_MODE, securekey, random);
        // 真正开始解密操作
        return cipher.doFinal(src);
    }
}
```