```
public class DESUtil {

    private static Key key;
    private static String KEY_STR = "myKey";
    private static String CHARSETNAME = "UTF-8";
    private static String ALGORITHM = "DES";

    static {
        try {
            KeyGenerator generator = KeyGenerator.getInstance(ALGORITHM);
            SecureRandom secureRandom =
SecureRandom.getInstance("SHA1PRNG");
            secureRandom.setSeed(KEY_STR.getBytes());
            generator.init(secureRandom);
            key = generator.generateKey();
            generator = null;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * 加密
     * @param str
     * @return
     */
    public static String getEncryptString(String str) {
        BASE64Encoder base64encoder = new BASE64Encoder();
        try {
            byte[] bytes = str.getBytes(CHARSETNAME);
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, key);
            byte[] doFinal = cipher.doFinal(bytes);
            return base64encoder.encode(doFinal);
        } catch (Exception e) {
            // TODO: handle exception
            throw new RuntimeException(e);
        }
    }
```

```
    /**
     * 解密
     * @param str
     * @return
     */
    public static String getDecryptString(String str) {
        BASE64Decoder base64decoder = new BASE64Decoder();
        try {
            byte[] bytes = base64decoder.decodeBuffer(str);
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.DECRYPT_MODE, key);
            byte[] doFinal = cipher.doFinal(bytes);
            return new String(doFinal, CHARSETNAME);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://127.0.0.1:3306/test?characterEncoding=utf-
8&allowMultiQueries=true&serverTimezone=UTC
jdbc.username=WnplV/ietfQ=
jdbc.password=xyHEykQVHqA=
```

```
public class EncryptPropertyPlaceholderConfigurer extends
PropertyPlaceholderConfigurer {
    // 需要解密的字段
    private String[] encryptPropNames = { "jdbc.username", "jdbc.password"
};

    @Override
    protected String convertProperty(String propertyName, String
propertyValue) {
        if (isEncryptProp(propertyName)) {
            // 解密
            String decryptValue = DESUtil.getDecryptString(propertyValue);
            return decryptValue;
        } else {
            return propertyValue;
        }
    }

    private boolean isEncryptProp(String propertyName) {
        for (String encryptpropertyName : encryptPropNames) {
            if (encryptpropertyName.equals(propertyName))
                return true;
        }
        return false;
    }
}
```

```
<!-- <context:property-placeholder location="classpath:*.properties"/> -->

<bean
class="com.light.ac.common.configuration.EncryptPropertyPlaceholderConfigu
rer">
    <property name="locations">
        <list>
            <value>classpath:db.properties</value>
        </list>
    </property>
    <property name="fileEncoding" value="UTF-8"/>
</bean>
```