隐藏用户名：

需求：

保留用户名前面一位，后面一位，中间统一用2个星号代替。

方案：

1，实体类中，声明一个新的变量，将用户名转换为隐藏方式；

页面显示隐藏的变量； | ---写一个公共方法；

2，js的处理；

js中，获取页面变量名，然后替换，最后显示到页面控件；

3，jsp的处理，

在页面，要显示的用户名字段，进行隐藏处理，然后显示；

最终：

默认，使用1；

不够的，使用下js处理；

或使用jsp的处理；

-----------------

实现：

公共类：

**public** **class** StrinUtils {

**public** **static** String stringSecretTo(String str) {

StringBuilder sb2 = **new** StringBuilder(str);

sb2 .replace(1, sb2.length()-1, "\*\*");

**return** sb2.toString();

}

}

实体类中：

**private** String userNameSecret; //声明一个新变量；并，只写get；

**public** String getUserNameSecret() {

userNameSecret = **this**.getUserName(); //getUserName(),获得本实体类中的用户名；

//userNameSecret = **super**.getUsername(); | username在继承类中的；

**if**(userNameSecret.length()>2){

userNameSecret = StrinUtils.*stringSecretTo*(userNameSecret);

}

**return** userNameSecret;

}

页面：

${transfer.userNameSecret}

-----------------

url中的用户名：不能隐藏，只能另外加解密处理；

//一般不需要，因为一般都会使用用户id进行传递；

1，公共类，

//加密字符串

**public** **static** String stringEncryptEn(String strEn) {

//字符串转换为字节

**byte**[] bt = strEn.getBytes();

//base64加密 ---只能加密字节；

Base64 be = **new** Base64();

String str = be.*encode*(bt);

//System.out.println(bt);

//System.out.println(str);

**return** str;

}

//解密字符串

**public** **static** String stringEncryptDe(String strDe) {

//base64解密---解密字符串为字节

Base64 bd = **new** Base64();

**byte**[] bt2 = bd.*decode*(strDe);

//字节转换为字符串

String str2 = **new** String(bt2);

//System.out.println(bt2);

//System.out.println(str2);

**return** str2;

}

2，实体类，

**private** String userNameEncrypt ; //声明一个新变量；并，只写get；

**public** String getUserNameEncrypt() {

userNameEncrypt = **this**.getUsername();

userNameEncrypt = StrinUtils.stringEncryptEn(userNameEncrypt);

**return** userNameEncrypt;

}

3，页面，

${transfer.userNameEncrypt}

4，接收用户名的action中：

userName = StrinUtils.*stringEncryptDe*(userName); //解密用户名

///////////////////////////////////////////////////