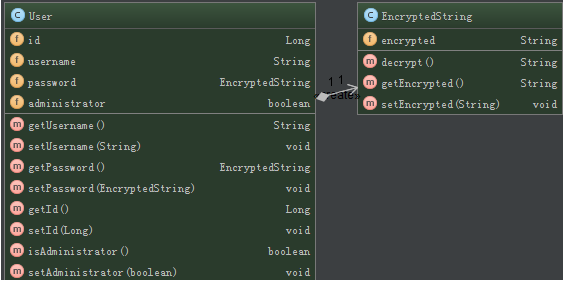
**Mybatis 示例之 复杂(complex)属性(property)**

2014年09月19日 20:22:31 [ljhdfas](https://me.csdn.net/ljhdfas" \t "_blank) 阅读数：226

Mybatis示例专栏：[http://blog.csdn.net/column/details/mybatis-sample.html](http://blog.csdn.net/column/details/mybatis-sample.html" \t "_blank)

Mybatis的复杂属性，Mybatis的这个特点很少被提及，但是有些时候确实又有用。Mybatis的复杂属性指的什么呢？

有如下两个对象：



在User对象中有一个字段password，类型是EncryptedString，这个类只有一个字段encrypted。

Mybatis的复杂属性指的就是User中password这样的属性。如果我们不用复杂属性这个特性来配置User对象，我们能怎么做呢？

在之前的文章中提到过[Association](http://blog.csdn.net/isea533/article/details/20868189)的使用方法（[Mybatis 示例之 Association](http://blog.csdn.net/isea533/article/details/20868189" \t "_blank)），我们可以将password配置为association，使用association的写法（如果看不懂，请先看association的介绍）如下：

**[html]** [view plaincopy](http://blog.csdn.net/isea533/article/details/23978453)[在CODE上查看代码片](https://code.csdn.net/snippets/299024)

1. **<resultMap** type="UserAlias" id="UserResult"**>**
2. **<id** column="id" jdbcType="INTEGER" property="id"**/>**
3. **<result** column="username" jdbcType="VARCHAR" property="username"**/>**
4. **<result** column="administrator" jdbcType="BOOLEAN" property="administrator"**/>**
5. **<association** property="password" javaType="org.apache.ibatis.submitted.complex\_property.EncryptedString"**>**
6. **<result** property="encrypted" column="password" javaType="string"**/>**
7. **</association>**
8. **</resultMap>**
9. <resultMap type="UserAlias" id="UserResult">
10. <id column="id" jdbcType="INTEGER" property="id"/>
11. <result column="username" jdbcType="VARCHAR" property="username"/>
12. <result column="administrator" jdbcType="BOOLEAN" property="administrator"/>
13. <association property="password" javaType="org.apache.ibatis.submitted.complex\_property.EncryptedString">
14. <result property="encrypted" column="password" javaType="string"/>
15. </association>
16. </resultMap>

使用这种方式可以达到我们的目的，但是对于这种只有1个或很少个属性的对象来说，配置一个association未免太麻烦了，如果使用复杂属性配置，会变的很容易。复杂属性写法如下：

**[html]** [view plaincopy](http://blog.csdn.net/isea533/article/details/23978453)[在CODE上查看代码片](https://code.csdn.net/snippets/299024)

1. **<resultMap** type="UserAlias" id="UserResult"**>**
2. **<id** column="id" jdbcType="INTEGER" property="id"**/>**
3. **<result** column="username" jdbcType="VARCHAR" property="username"**/>**
4. **<result** column="password" jdbcType="VARCHAR" property="password.encrypted"**/>**
5. **<result** column="administrator" jdbcType="BOOLEAN" property="administrator"**/>**
6. **</resultMap>**
7. <resultMap type="UserAlias" id="UserResult">
8. <id column="id" jdbcType="INTEGER" property="id"/>
9. <result column="username" jdbcType="VARCHAR" property="username"/>
10. <result column="password" jdbcType="VARCHAR" property="password.encrypted"/>
11. <result column="administrator" jdbcType="BOOLEAN" property="administrator"/>
12. </resultMap>

使用这种复杂属性时，Mybatis会自动创建相应类型的对象，并将查询结果赋值给属性。这种方式明显比上面的要简单很多，当EncryptedString类有多个字段需要赋值时，用association方式的需要在其中增加相应的result，使用复杂对象的只需要在resultMap中继续添加result，property="password.xxx"即可。虽然两种方式都没有问题，建议选择其中一种方式，便于其他开发者统一理解。

除了上面这种复杂属性外，还有一种我们可能已经使用过多次的方式，如下：

**[html]** [view plaincopy](http://blog.csdn.net/isea533/article/details/23978453)[在CODE上查看代码片](https://code.csdn.net/snippets/299024)

1. **<update** id="update" parameterType="UserAlias"**>**
2. UPDATE user SET
3. username = #{username,jdbcType=VARCHAR},
4. password = #{password.encrypted:VARCHAR},
5. administrator = #{administrator,jdbcType=BOOLEAN}
6. WHERE
7. id = #{id:INTEGER}
8. **</update>**
9. <update id="update" parameterType="UserAlias">
10. UPDATE user SET
11. username = #{username,jdbcType=VARCHAR},
12. password = #{password.encrypted:VARCHAR},
13. administrator = #{administrator,jdbcType=BOOLEAN}
14. WHERE
15. id = #{id:INTEGER}
16. </update>

这种通过点(.)访问的方式也是Mybatis支持的复杂属性，可以是很多层的属性。并且支持数组和集合的“[]”下标获取。

不知道有没有注意到上面这个例子有点不同寻常的地方，可能你也发现了，有些地方写的是

**[html]** [view plaincopy](http://blog.csdn.net/isea533/article/details/23978453)[在CODE上查看代码片](https://code.csdn.net/snippets/299024)

1. #{username,jdbcType=VARCHAR}

#{username,jdbcType=VARCHAR}

而有的是

**[html]** [view plaincopy](http://blog.csdn.net/isea533/article/details/23978453)[在CODE上查看代码片](https://code.csdn.net/snippets/299024)

1. #{id:INTEGER}

#{id:INTEGER}

第一种是最常见的方式，第二种是一种更简单的方式，第二种会自动将第一个冒号（:）后面的属性记录成jdbcType，后面还可以继续跟其他的属性。