# mybatis中#和$的区别

动态 sql 是 mybatis 的主要特性之一，在 mapper 中定义的参数传到 xml 中之后，在查询之前 mybatis 会对其进行动态解析。mybatis 为我们提供了两种支持动态 sql 的语法：#{} 以及 ${}。

1. #传入的数据都是有类型的，$传入的数据是直接显示在sql中。
2. #{}的参数替换是发生在DBMS中的，及#进行的预处理操作，而${}则发生在动态解析过程中，直接拼接的sql。
3. #方式能够很大程度防止sql注入，$方式无法防止sql注入。
4. $方式一般用于传入数据库对象，例如传入表名，以及order by 后的排序字段。
5. 一般能用#的就不用$。

Mybatis本身说明：

String Substitution

By default, using the #{} syntax will cause MyBatis to generate PreparedStatement properties and set the values safely against the PreparedStatement parameters (e.g. ?). While this is safer, faster and almost always preferred, sometimes you just want to directly inject a string unmodified into the SQL Statement. For example, for ORDER BY, you might use something like this:

ORDER BY ${columnName}

Here MyBatis won't modify or escape the string.

NOTE It's not safe to accept input from a user and supply it to a statement unmodified in this way. This leads to potential SQL Injection attacks and therefore you should either disallow user input in these fields, or always perform your own escapes and checks.

从上文可以看出：

1. 使用#{}格式的语法在mybatis中使用Preparement语句来安全的设置值，执行sql类似下面的：

PreparedStatement ps = conn.prepareStatement(sql);

ps.setInt(1,id);

这样做的好处是：更安全，更迅速，通常也是首选做法。

2. 不过有时你只是想直接在 SQL 语句中插入一个不改变的字符串。比如，像 ORDER BY，你可以这样来使用：

ORDER BY ${columnName}

此时MyBatis 不会修改或转义字符串。

这种方式类似于：

Statement st = conn.createStatement();

ResultSet rs = st.executeQuery(sql);

这种方式的缺点是： 以这种方式接收从用户输入的内容并提供给语句中不变的字符串是不安全的，会导致潜在的 SQL 注入攻击，因此要么不允许用户输入这些字段，要么自行转义并检验。