**Mybatis基础版自动回复机器人案例记录点**

1.在web.xml中建servlet来进行中转控制（为了向列表页面跳转）；

2.页面放在WEB-INF下便于控制和管理（必须从后台走）。如果放在外面会被直接访问到（除非进行拦截\*.jsp）。

3.通过文件夹管理各类文件。

4.注意项目命名规范。

5.添加合理的注释，有助于程序维护。

6.使用预编译SQL语句，后面还要加上其他查找条件。

7.使用“SELECT 列名”提高数据库效率（否则数据库引擎还要去解析列名，例如SELECT \* 这种方式）。

8.Alt+Shift+J 自动添加文档注释

Ctrl+Shift+Y 将大写转换为小写

Ctrl+Shift+J 生成方法注释

9.使用JSTL结合EL表达式

10.Servlet的声明周期（修改了Servlet，则需要重启服务器）。Servlet在Web容器启动的时候只实例化一次（但是在MyEclipse的开发者模式下会自动重新加载）。

11.JSTL课程--《Java遇见HTML-JSP篇》

12.网友说可能需要加这个限定：

<%@ page isELIgnored="false" %>

13.使用String不断拼接新的字符串会产生很多资源垃圾。

14.MySQL语句拼接使用空格，所以在使用JDBC拼接SQL语句的时候一定要注意空格的问题。

15.在涉及到EL表达式的使用时，一定要严格遵守规范，细心谨慎，哪怕是多一个空格都会出错。

16.Servlet一般不直接接触DAO层，中间还有一层来处理业务逻辑。本案例中中间是Service层（也有公司叫做Logic层），简单地将DAO层查到的数据返回到Servlet层，与Servlet层对应。

17.Servlet典型地大致做以下工作：①接收页面参数；②业务需要调用Service；③向页面传值；④页面跳转。

18.Mybatis的工作主要在配置方面。

核心配置文件是G:\Files\JARs\MyBatis\mybatis-3-mybatis-3.4.5\mybatis-3-mybatis-3.4.5\src\test\java\org\apache\ibatis\submitted\complex\_property\Configuration.xml，其中mybatis-3-mybatis-3.4.5是Mybatis提供的源码程序，在下载Mybatis JAR包的同时也下载了。

19.本案例在使用JDBC方式进行数据访问时，DAO层里包含了很多功能：

加载驱动；

获取连接；

执行SQL语句；

将查找到的数据进行封装；

返回数据；

但实际上加载驱动和获取连接都不应该是DAO层的工作（根据之前的经验，可以新建一个DBUtil层来容纳这两个功能），之后后三者才是。

20.对于DAO层，只要能拿到这样一个对象即可：

①对象能与数据库交互；

②对象能执行SQL语句。

这就是Mybatis需要向DAO层提供的对象，这个对象叫做SqlSession对象。

21.SqlSession的作用：

①向SQL语句传入参数；

②执行SQL语句；

③获取执行SQL语句的结果；

④事务的控制。

22.如何得到SqlSession？

①通过配置文件获取数据库连接相关信息；

②通过配置信息构建SqlSessionFactory；

③通过SqlSessionFactory打开数据库会话。

23.（前版本V1.0）DAO层中的Connection，PreparedStatement和ResultSet最后都需要关闭，Mybatis的SqlSession同样也需要关闭。

24.（Point 18）Mybatis不像是JDBC把SQL语句写死，也不采用Java语句动态拼SQL语句，而是采用配置文件的方式来执行SQL操作。

25.Mybatis不仅可以在Web上运行，同样也可以使用main方法进行单元测试。

26.Mybatis封装了SQL操作，其模板是G:\Files\JARs\MyBatis\mybatis-3-mybatis-3.4.5\mybatis-3-mybatis-3.4.5\src\test\java\org\apache\ibatis\submitted\complex\_property\User.xml文件。

27.Mybatis的SQL操作配置文件（本案例的Message.xml）中有唯一id（命名）的SQL语句，并且要求整个项目中的SQL语句id不可相同；所以为了避免麻烦，该配置文件采用命名空间管理机制（类似Java中的包管理机制），不同命名空间下可以有相同的SQL语句id。

28.Message.xml文件中：select标签中SQL语句查找到的列的值会在resultMap标签中找到对应关系并将值赋值给对象的对应的属性；如果select标签中的列名在resultMap中没有找到对应的，那么就不赋值；反过来，那么resultMap的列对应的属性就为空。

29.SqlSession的select方法查找返回为单一的值的数据；selectList方法查找返回为List的数据，并且在Message.xml中配置完成后才可以确定List的泛型是什么类型。

30.Message.xml中<result>标签中的jdbcType属性是指对应的jdbctype类型。jdbctype和其他的数据库数据类型有一种对应关系，是一种数据类型。

31.Message.xml中：<select>标签中的parameterType属性，是指执行SQL语句需要动态传递参数时的参数的类型，如果过类型是java.lang包下的（比如String类型），那么可以直接写属性值为String，即：parameterType="String"；而如果是用户自定义的类，那么需要拼写类的全名（即包含包名的），例如：parameterType="com.imooc.bean.Message"。

32.Mybatis中的OGNL表达式。

OGNL是一种强大的表达式语言，强大之处在于它可以直接支持Java对象的方法。

33.（Point 14）Mybatis动态拼接SQL语句时是不需要注意空格问题的，因为是写在标签里的，最后会自动处理成完整的、正确的SQL语句的。

34.HTML和XML中对Java语言中的一些字符不能直接支持，需要进行转义，比如双引号使用&quto;，&&使用&amp;&amp;。

35.Mybatis在解析时遇到#{command}时，会将其替换成占位符?，并且在其知道属性和列对应关系的前提下会将后面传递的参数替换成占位符，实现动态传递参数。

36.Log4j的使用需要以下两步：①引入JAR包；②引入Log4j的配置文件log4j.properties。

37.如果需要查看第三方类库JAR包的源码时，需要先下载下来JAR包对应的源码工程（只有JAR包下的class文件是无法查看源码的），然后在MyEclipse中为这个JAR包添加Attach Source，操作路径是：项目右键-->Properties-->Java Build Path-->Libraries-->选择需要查看的类库JAR包-->展开有“Source Attachment”-->Edit，选择下载好的源码工程即可。在Referenced Libraries目录下查看源码。

38.在没有框架对Servlet进行封装的情况下，页面上只要有一个请求动作就需要有对应的Servlet以及和Servlet对应的Service。而在本案例中将一些功能相似的Service写在一起（MaintainService.java）。

39.JSP页面上参数传递到Servlet有两种方式：①POST方法；②GET方法。两种方式各有优缺点。

40.Mybatis需要手动提交事务，在手动提交事务之前，框架会认为之前的操作属于一个事务；这不同于JDBC的自动提交事务，Mybatis在封装的时候使用setAutoCommit(false)禁止自动提交事务。

41.Servlet从JSP页面获取的参数只能为String类型（包含单个参数或String数组），所以就涉及到数据类型的转换。

42.Servlet从JSP页面获取的参数是数组（意思是指相同属性的多个值，比如多选框）时，只需要从JSP页面传递多个name相同的参数，在Servlet中使用getParameterValues("name")即可获取那些传递过来的参数放在数组里。

43.不提倡在JSP页面中直接写JavaScript代码，而是使用引入js文件的方式。

44.jQuery框架。

45.Mybatis的核心配置文件并没有在项目的web.xml文件中有什么（读取）配置，而是在获取SqlSession的类中进行了读取。

46.JS中，如果一个变量等于满足以下条件：

var content;

content = ""（空字符串）; content = null; content = undefined; content = 0（数字0）;

那么使用if(content)判断时结果为false。

47.在更改类名时，不要直接修改。选中类名，右键选择“Refactor（重构）”->“Rename”，这样会把相关的引用的名字也改过来，省去很多麻烦和避免错误。

48.Ajax请求。

49.【注意】JDBC中拼接SQL语句时，如果为查找的字段赋予了别名，即结果集中的该字段是别名，那么在使用结果集ResultSet对象获取该字段时，只能通过别名来获取。所以Mybatis中SQL配置文件也是如此，即<result column="COMMAND" jdbcType="VARCHAR" property="command"/>中的column属性值可能需要使用字段别名。

50.【注意】JDBC中拼接SQL语句时，如果字段名前面添加了表名（或者表的别名）用以和同名的字段区分，那么在使用结果集ResultSet对象获取该字段时是不能添加表名前缀的，这样无法获取到值。所以Mybatis中SQL配置文件也是如此，即<result column="COMMAND" jdbcType="VARCHAR" property="command"/>中如果column="a.COMMAND" 这样是不行的。因为JDBC中结果集的字段是不含表名的，例如可以使用ResultSet对象rs.getMetaData().getColumnName(column)来查看所有字段名，是不含表名的。

51.所以面对以上两种情况，有时候SQL配置文件中可能会面临这样的矛盾：相同的字段名如果无法使用表名来区分的话那么就不好区分了，这种情况下只能是必须有字段使用别名了。

52.【其他常用标签】

（1）<where></where>标签

功能①：在需要的时候帮助生成where关键字，完成条件检索。

功能②：去掉where关键字后面的多余的and或or。

（2）<sql id=""></sql>

功能：类似Java中定义常量，将SQL语句中经常用到的部分在这里定义，以后可以直接拿来使用。

使用时：在语句中需要引用的地方插入***<include refid="" />***。

（3）<set></set>标签

功能①：在需要的时候帮助生成set关键字，完成数据表更新。

功能②：去掉语句最后多余的“,”。

（4）<trim></trim>标签

trim标签更具有灵活性，可代替set、where标签。

①：***<trim prefix="where/set/其他" suffix="test"></trim>***

prefix：前缀；suffix：后缀。表示如果需要则在其包含的语句前面或后面输出相应的前缀或后缀。如果设置前缀是where，那么就会发挥相当于where标签的功能。

②：***<trim prefix="where" prefixOverrides="and/or">***

相当于上面where标签的功能②。

***<trim prefix="set" suffixOverrides=",">***

相当于上面set标签的功能②。

（5）从多个条件中选择其中一个（分支）：

***<choose>***

***<when test=""></when>***

***<when test=""></when>***

***<when test=""></when>***

***<otherwise></otherwise>***

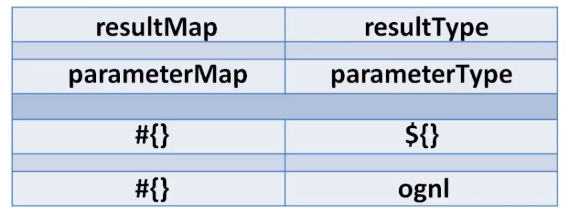
***</choose>***

相当于Java中的if...else if...else或者switch...case语句。

（6）一对多查询时，在父表中查询到一条数据时关联子表中的多条数据，在SQL配置文件中使用<collection property="" resultMap="" />标签；而现在反过来，查询到子表一条数据时想去管理父表中的数据，那么使用***<association property="" resultMap="" />***。



53.【容易混淆的概念】



（1）xxxMap与xxxType

xxxMap就得需要映射配置，而xxxType则是指类型，需要指定Java类。parameterMap是为了配合老版本而存在的，实际使用不多。

resultMap是指向结果集列名和实体类属性的映射配置；而如果使用resultType代替resultMap，那么是不需要进行映射配置的，因为系统会自动匹配，这时候SQL语句中的结果集列名只要与实体类中属性名相同（不区分大小写）即是映射完成。

但resultMap也有一个好处在于可以通过设置typeHandler来解决数据库与Java数据类型不匹配的问题，这是resultType无法实现的。

（2）#{}与${}

***SELECT ID,COMMAND,DESCRIPTION,CONTENT FROM message WHERE***

***COMMAND=#{command}***

这里在执行时会将#{command}替换为参数占位符?，PreparedStatement会将?再替换为真正传递过来的参数。也就是说进行了SQL语句预编译的过程。

**预编译的优点有：效率高，防止SQL注入等。**

***SELECT ID,COMMAND,DESCRIPTION,CONTENT FROM message WHERE COMMAND=${command}***

这里在执行时会将#{command}直接替换为传递过来的command参数。

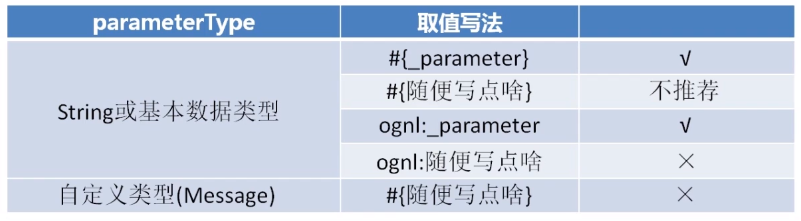
从以上角度来看，使用#{}更多，但是**两者的真正区别在于是否经过预编译过程**，${}是没有预编译的。例如需要对查找结果集进行排序时：

***SELECT ID,COMMAND,DESCRIPTION,CONTENT FROM message WHERE***

***COMMAND=#{command} ORDER BY ${command}***

这个时候如果在最后使用#{command}，那么预编译后就会被替换为?，这样是无法进行排序的，而使用${command}是可以的。

（3）#{}取值与OGNL中取值方式



54.【常见问题解析】

（1）获取自增值主键值

***<insert id="insertCommand" useGeneratedKeys="true" keyProperty="id" parameterType="com.imooc.bean.Message">***

***INSERT INTO command (COMMAND,DESCRIPTION) VALUES (#{command},#{description})***

***</insert>***

***useGeneratedKeys="true"***表示使用自增主键，所以在传递参数的时候主键对应的属性id是空的；

***keyProperty="id"***表示将插入数据后查询到的主键值放到属性id中。

（2）乱码

在本案例中要关注以下地方编码的要求（实际应用中还会有其他地方）：

①文件本身的编码方式；

②JSP页面的编码：

***<%@ page language="java" pageEncoding="utf-8" contentType="text/html;charset=utf-8" %>***

或者在HTML中

***<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>***

③Servlet接收页面传值时设置编码；

④如果以get方式提交中文，Tomcat中也需要设置；

⑤在与数据库交互时，Mybatis配置数据库连接时：

***<property name="url" value="jdbc:mysql://127.0.0.1:3306/micro\_message?useUnicode=true&amp;characterEncoding=utf8"/>***

⑥数据库安装时要求设置数据库的编码；

⑦新建数据库和数据表时的编码。

（3）找不到namespace.id的异常效果。

如果在Mybatis核心配置文件没有引入SQL操作配置文件或者SqlSession对象调用select方法时namespace.id名称写错，都会抛出这样的异常。

（4）SQL语法错误。

（5）不要过度使用${}。