## 什么是mybatis

mybatis是一个java持久层框架，java中操作关系型 数据库用的是jdbc，mybatis是对jdbc的一个封装。

## Jdbc的预编译



如果你第一次发送sql，因为sql也是语言，要运行也需要编译，而数据库就是编译器和执行器，第一次发sql，会编译，然后把编译完成的东西记录到数据库缓存，下次再发送预编译的语句，发现sql一样，就省略了编译的步骤效率就高了。否则每次数据库都要编译

或者理解为java帮你编译了，数据库直接运行就好，这样效率很高

用到了数据库的预编译，以及数据库的缓存。

## 输出和输入

输入 映射：将值设置到映射对象，或者说设置到数据库，就是输入

输出 映射：就是将数据库的数据取出来，输出到java映射对象，就是输出

## 硬编码

硬编码不便于维护，系统不易维护，只要有硬编码，系统是不利于维护的

## 命名要方便直观

## Eclipse的classpath路径

只要是在项目名称目录，也就是根目录下的类型的文件夹，都是classpath路径，不光是src路径才是对应到classpath，你还可以创其他路径，也可以对应到classpath路径

就相当于classpath

## #{}和${}

他会自动将类型转换成对应的数据库类型传入，如果你用${}你传的什么 就是什么，

比如date类型，你要插入数据库，你用#{}他会自动将字符串转成date，${}他就必须要手动转成date才能传进去，MySQL可以自动转，但是Oracle就不行，反正#的好些，绝对自动转的

#{}

表示一个占位符，向占位符输入参数，mybatis自动进行java类型和jdbc类型的转换。

程序员不需要考虑参数的类型，比如：传入字符串，mybatis最终拼接好的sql就是参数两边加单引号。

#{}接收pojo数据，可以使用OGNL解析出pojo的属性值

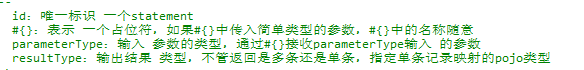
${}

表示sql的拼接，通过${}接收参数，将参数的内容不加任何修饰拼接在sql中。

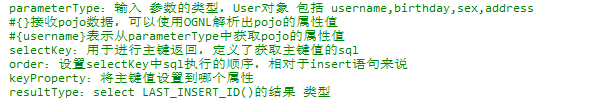
${}也可以接收pojo数据，可以使用OGNL解析出pojo的属性值

缺点：不能防止sql注入。

## Mapper配置



如果是pojo（javabean 实体bean）的就用ognl对象图导航



## 善用模板代码可以很有效的提高工作效率，也就是码代码的效率

## ORM指的是对象关系映射

Hibernate是一个标准的对象关系映射

Mybatis需要程序员自己写sql，所以说是一个不完全的ORM框架

## 需求固定，模型也就稳定了



## Hibernate和mybatis的映射文件

Hibernate为什么要把映射文件放在实体类同包下

因为hibernate的sql基本不会存在优化之类的，所以需求比较固定，对象数据模型稳定，所以放在同包下，也就没什么了，和java文件放在一起无所谓的，反正也不会改，因为hibernate的映射，都是对象与表的映射，基本不会改变的，又没有其他的什么

Mybatis放在其他路径都可以，一般都是类路径

因为mybatis既有映射，还有重要的核心sql语句，为了方便以后的维护，对不同需求可能需要更改，就可以放到一个专门的路径，方便管理，方便和后期的维护

因为mybatis的映射可以根据需求更改，而hibernate一般都是需求固定，而且也不好更改，处理表对象的映射，这个东西一般不会改，而sql更改就不一样了

## 多线程共享成员变量数据

全局的变量，数据，或资源，都是共享的，都是单例的

## 线程不安全的理解

多线程访问，如果以单例方式就不行了（就会出现问题），可能数据会冲突

通俗一点，多线程一般就是多个用户访问同一个资源，这时多个用户操作同一个资源（是单例的，在系统中这个资源只有一个），这个资源是共享的，而每个用户都有不同的数据，这样多个用户操作同一个资源就可能会出现数据冲突，出问题

比如：这个线程准备执行一个更新语句，还没更新，另一个线程来了，因为单例的，资源只有一个，数据共享的，所以他就会覆盖那个资源中的sql数据，执行后面的线程的操作

一个线程就可以看做一个用户吧

多个线程去访问，每个方法体中的内存空间是互不影响的

## Sqlsession的使用范围

不要定义为成员变量（全局变量），因为全局的都是共享的，上面说了，共享的话，就变成了单例的了，这样多线程访问就会有问题，数据可能冲突

Sqlsession最佳应用范围在方法体内，在方法内定义局部变量（来）使用sqlsession

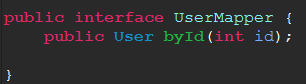
## 抛异常的机制

异常抛出，是抛给子类的，谁实现你或者继承李，就会得到你那个父类的所有，包括异常，异常也就抛给你了，你来处理异常

如果一个方法抛了异常，那么调用这个方法的时候也就会抛异常，也就是谁用，谁是子类就哪个处理

## 一个方法return以后，后面的语句就会失效，不会再执行

## 形参

这个id就是一个形参

实际参数指的是有确切值的参数，不一定是初始化的，只要有确定的值就是实参

引用是指，将一个对象的，或者其他的东西赋给他，不是具体的值，而是带有值的变量，或参数，带有实参的变量的引用

## 子类不能抛出比父类更多或者更大的异常

比如如果父类没有抛异常，那么你也不能有异常，接口的实现类是子类，接口也可以理解为父类

只是说不能抛，但是你可以抓

## 单例的解释

单例就是在整个系统中，整个运行过程中，只有一个这个对象，只new了一次，是所有用户都只有一个，而且这一个是相同个的，只new了一次

## 一对多，多对多的关联关系

子表指向父表，也就是外键引用主键，外键表指向主键表

一般多的一方，都是维护外键的表为多的一方，也就是子表为多的一方

如：用户和订单，一个用户可以创建多个订单，订单引用用户，订单为外键为子表，他是多的一方

只要维护的有外键，那么这个表就有可能成为多的一方，成为一个子表，否则他就一定是一的一方

## 什么东西是属于存在内存的

就像一级缓存就是将查到的存入缓存（也就是内存），他内部又是map，所以，只要是java中的域对象，或者说可以存值的对象，都是将数据存到内存中的，其他的都是存到磁盘的

就是这么简单的理解就好了

## 并发处理能力

通过分布式管理可以提高并发处理能力

如果提高数据库的高并发问题，就可以使用缓存，将数据存入缓存（内存）不从数据库（数据库的文件也是保存在磁盘上的 所以也可以说不从磁盘取）取，直接从内存取，就可以减少数据库的压力，从内存取就很快，也就不会占用数据库的并发数量，从而提高数据库的高并发能力

## Mybatis和Spring的整合 mapper的开发方式

最原始的dao开发方式，需要创建实现类，实现类中用sqlsession.selectone()之类的实现查询，里面的参数就是mapper.xml中的statement，

以前得到mapper的代理对象，是通过sqlsession.getmapper()传入一个接口，就会生成这个接口对应的代理对象，得到的也就是一个代理对象，用接口接收，就是这个接口的代理对象，他就会根据接口的返回值动态生成代理对象，代理对象中也是用的sqlsession.selectone()之类的实现查询，因为如果自己写实现类，模板都是重复的，为了方便，就自动生成了

整合以后，将mapper配置到spring，通过spring得到bean，mybatis就会自动生成代理对象，得到的bean就是一个代理的对象，用接口接收/引用，就是这个接口的代理对象,目前需要自己创建Spring容器，真正开发中由web.xml中的监听器创建容器，然后直接用就行了，直接注入，注入后直接就可以用了，控制层直接就可以访问这个东西，他就会得到里面的注入的对象，直接就可以用，不用getbean，才调用方法，只要有对象就能用