

# 1 MyBatis简介

## 1.1 MyBatis

MyBatis 本是[apache](http://baike.baidu.com/view/28283.htm)的一个开源项目[iBatis](http://baike.baidu.com/view/628102.htm), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis，实质上Mybatis对ibatis进行一些改进。

MyBatis是一个优秀的持久层框架，它对jdbc的操作数据库的过程进行封装，使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、创建connection、创建statement、手动设置参数、结果集检索等jdbc繁杂的过程代码。

对jdbc的封装框架有哪些：Hibernate,dbutils,jdbcTemplate[spring]，mybatis

原理：Mybatis通过**xml或注解**的方式将要执行的各种statement（statement、preparedStatemnt、CallableStatement）配置起来，并通过java对象和statement中的sql进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射成java对象并返回。

## 1.2 jdbc程序代码

|  |
| --- |
|  |

**存在的问题**

* 数据库连接频繁开启和关闭，会严重影响数据库的性能。
* 代码中存在硬编码，分别是数据库部分的硬编码和SQL执行部分的硬编码。

## 1.3 MyBatis的框架核心

|  |
| --- |
| 1. mybatis配置文件，包括**Mybatis全局配置文件和Mybatis映射文件**，其中全局配置文件配置了数据源、事务等信息；映射文件配置了SQL执行相关的 信息。 2. mybatis通过读取配置文件信息（全局配置文件和映射文件），构造出**SqlSessionFactory，**即会话工厂。 3. 通过SqlSessionFactory，可以创建**SqlSession**即会话。Mybatis是通过SqlSession来操作数据库的。 4. SqlSession本身不能直接操作数据库，它是通过底层的**Executor**执行器接口来操作数据库的。Executor接口有两个实现类，一个是普通执行器，一个是**缓存执行器（默认）**。 5. Executor执行器要处理的SQL信息是封装到一个底层对象**MappedStatement**中。该对象包括：SQL语句、输入参数映射信息、输出结果集映射信息。其中输入参数和输出结果的映射类型包括**HashMap集合对象、POJO对象类型**。 |

# Mybatis入门

## 2.1 环境准备

### 创建数据库表

|  |
| --- |
|  |

### 下载MyBatis

mybaits的代码由github.com管理，下载地址：<https://github.com/mybatis/mybatis-3/releases>

### 创建项目导包

导入下面的包

|  |
| --- |
|  |
|  |

### 添加log4j.properties

Mybatis使用的日志包是log4j的，所以需要添加log4j.properties。

在classpath下创建log4j.properties如下：【文件内容可以从mybatis-3.2.7.pdf中拷贝】

# Global logging configuration

log4j.rootLogger=DEBUG, stdout

# Console output...

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n

日志级别在开发阶段设置成DEBUG，在生产阶段设置成INFO或者ERROR。

## 2.2 开发步骤

|  |
| --- |
| 1. 创建PO（model）类，根据需求创建； 2. 创建全局配置文件SqlMapConfig.xml； 3. 编写映射文件； 4. 加载映射文件，在SqlMapConfig.xml中进行加载； 5. 编写测试程序，即编写Java代码，连接并操作数据库。   思路：   1. 读取配置文件； 2. 通过SqlSessionFactoryBuilder创建SqlSessionFactory会话工厂。 3. 通过SqlSessionFactory创建SqlSession。 4. 调用SqlSession的操作数据库方法。 5. 关闭SqlSession。 |

### 创建PO类

|  |
| --- |
|  |

### 创建SqlMapConfig.xml

在classpath(src)下，创建SqlMapConfig.xml文件【SqlMapConfig.xml（文件头可以从mybatis-3.2.7.pdf文档的2.1.2小节中拷贝）】

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <!-- 配置mybatis的环境信息 -->  <environments default=*"development"*>  <environment id=*"development"*>  <!-- 配置JDBC事务控制，由mybatis进行管理 -->  <transactionManager type=*"JDBC"*></transactionManager>  <!-- 配置数据源，采用dbcp连接池 -->  <dataSource type=*"POOLED"*>  <property name=*"driver"* value=*"com.mysql.jdbc.Driver"*/>  <property name=*"url"* value=*"jdbc:mysql://localhost:3306/mybatis?useUnicode=true&amp;characterEncoding=utf8"*/>  <property name=*"username"* value=*"root"*/>  <property name=*"password"* value=*"123456"*/>  </dataSource>  </environment>  </environments>  </configuration> |

### 映射文件

在classpath下，创建sqlmap文件夹。在sqlmap目录下，创建User.xml映射文件。

**【Mybatis的映射文件头（可以从mybatis-3.2.7.pdf文件中拷贝）】**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <!--  namespace：命名空间，它的作用就是对SQL进行分类化管理，可以理解为SQL隔离  注意：使用mapper代理开发时，namespace有特殊且重要的作用  -->  <mapper namespace=*"test"*>  <!--  [id]：statement的id，要求在命名空间内唯一  [parameterType]：入参的java类型  [resultType]：查询出的单条结果集对应的java类型  [#{}]： 表示一个占位符?  [#{id}]：表示该占位符待接收参数的名称为id。注意：如果参数为简单类型时，#{}里面的参数名称可以是任意定义  -->  <select id=*"findUserById"* parameterType=*"int"* resultType=*"com.gyf.domain.User"*>  SELECT \* FROM USER WHERE id = #{id}  </select>  </mapper> |

### 配置文件加载映射文件

|  |
| --- |
|  |

### 测试类

|  |
| --- |
|  |

## 2.3 更多案例讲解

### 模糊查询用户信息

|  |
| --- |
| <!--  [${}]：表示拼接SQL字符串  [${value}]：表示要拼接的是简单类型参数。  注意：  1、如果参数为简单类型时，${}里面的参数名称必须为value  2、${}会引起SQL注入，一般情况下不推荐使用。但是有些场景必须使用${}，比如order by ${colname}  -->  <select id=*"findUserByName"* parameterType=*"String"* resultType=*"com.gyf.domain.User"*>  SELECT \* FROM USER WHERE username like '%${value}%'  </select> |
|  |

### 插入用户信息

|  |
| --- |
|  |
|  |

### 删除用户

|  |
| --- |
|  |
|  |

### 更新用户

|  |
| --- |
|  |
|  |

### 主键返回之MySQL自增主键

思路：

* MySQL自增主键，是指在insert之前MySQL会自动生成一个自增的主键。
* 我们可以通过MySQL的函数获取到刚插入的自增主键:

LAST\_INSERT\_ID()

* 这个函数是在insert语句之后去调用。

|  |
| --- |
| <insert id=*"insertUser"* parameterType=*"com.gyf.domain.User"*>  <!--  [selectKey标签]：通过select查询来生成主键  [keyProperty]：指定存放生成主键的属性  [resultType]：生成主键所对应的Java类型  [order]：指定该查询主键SQL语句的执行顺序，相对于insert语句  [last\_insert\_id]：MySQL的函数，要配合insert语句一起使用 -->  <selectKey keyProperty=*"id"* resultType=*"int"* order=*"AFTER"*>  SELECT LAST\_INSERT\_ID()  </selectKey>  <!-- 如果主键的值是通过MySQL自增机制生成的，那么我们此处不需要再显示的给ID赋值 -->  INSERT INTO USER (username,sex,birthday,address)  VALUES(#{username},#{sex},#{birthday},#{address})  </insert> |

### 主键返回之MySQL自增UUID

|  |
| --- |
| <insert id=*"insertUser"* parameterType=*"com.gyf.domain.User"*>  <selectKey keyProperty=*"id"* resultType=*"String"* order=*"BEFORE"*>  SELECT UUID()  </selectKey>  INSERT INTO USER (username,sex,birthday,address)  VALUES(#{username},#{sex},#{birthday},#{address})  </insert> |

### ORCLE主键

SELECT user\_seq.nextval() FROM dual

### 小结：

|  |
| --- |
| **parameterType和resultType**  parameterType指定输入参数的java类型，可以填写别名或Java类的全限定名。  resultType指定输出结果的java类型，可以填写别名或Java类的全限定名。  **#{}和${}**  #{}：相当于预处理中的占位符？。  #{}里面的参数表示接收java输入参数的名称。  #{}可以接受HashMap、POJO类型的参数。  当接受简单类型的参数时，#{}里面可以是value，也可以是其他。  #{}可以防止SQL注入。  ${}：相当于拼接SQL串，对传入的值不做任何解释的原样输出。  ${}会引起SQL注入，所以要谨慎使用。  ${}可以接受HashMap、POJO类型的参数。  当接受简单类型的参数时，${}里面只能是value。  **selectOne和selectList**  selectOne：只能查询0或1条记录，大于1条记录的话，会报错：  selectList：可以查询0或N条记录 |

## 2.4 MyBatis的Dao编写 【一般不用，有更多好方式】

### dao

|  |
| --- |
|  |
|  |

### 测试

|  |
| --- |
|  |

## 2.5 MyBatis的Dao编写【mapper代理方式实现】

Mapper代理的开发方式，程序员只需要编写mapper接口（相当于dao接口）即可。Mybatis会自动的为mapper接口生成动态代理实现类。

**不过要实现mapper代理的开发方式，需要遵循一些开发规范。**

### 开发规范

|  |
| --- |
| 1. mapper接口的全限定名要和mapper映射文件的namespace的值相同。 2. mapper接口的方法名称要和mapper映射文件中的statement的id相同； 3. mapper接口的方法参数只能有一个，且类型要和mapper映射文件中statement的parameterType的值保持一致。 4. mapper接口的返回值类型要和mapper映射文件中statement的resultType值或resultMap中的type值保持一致；   通过规范式的开发mapper接口，可以解决原始dao开发当中存在的问题：   * 模板代码已经去掉； * 剩下去不掉的操作数据库的代码，其实就是一行代码。这行代码中硬编码的部分，通过第一和第二个规范就可以解决。 |

### 编写步骤

### 第一步：【】

重新写个UserMapper配置文件和定义mapper映射文件UserMapper.xml（内容同Users.xml，**除了namespace的值**），放到新创建的目录mapper下。

|  |
| --- |
|  |

### 第二步：【添加映射配置文件】

|  |
| --- |
|  |

### 第三步：测试

|  |
| --- |
|  |

# 3 全局配置文件其它配置

## 3.1 properties数据库文件配置

### 在src下配置个db.properties文件

|  |
| --- |
|  |

### 修改全局的配置文件

|  |
| --- |
|  |

## 3.2 setting【了解】

|  |
| --- |
|  |

### 具体配置详解

|  |
| --- |
|  |

## 3.3 typeAliases

别名是使用是为了在映射文件中，更方便的去指定参数和结果集的类型，不再用写很长的一段全限定名。

### mybatis支持的别名

|  |  |
| --- | --- |
| 别名 | 映射的类型 |
| \_byte | byte |
| \_long | long |
| \_short | short |
| \_int | int |
| \_integer | int |
| \_double | double |
| \_float | float |
| \_boolean | boolean |
| string | String |
| byte | Byte |
| long | Long |
| short | Short |
| int | Integer |
| integer | Integer |
| double | Double |
| float | Float |
| boolean | Boolean |
| date | Date |
| decimal | BigDecimal |
| bigdecimal | BigDecimal |

### 自定义别名

|  |
| --- |
|  |

## mappers

### <mapper resource=’’/>

使用相对于类路径的资源

如：<mapper resource="sqlmap/User.xml" />

### <mapper url=’’/> 【不用】

使用完全限定路径

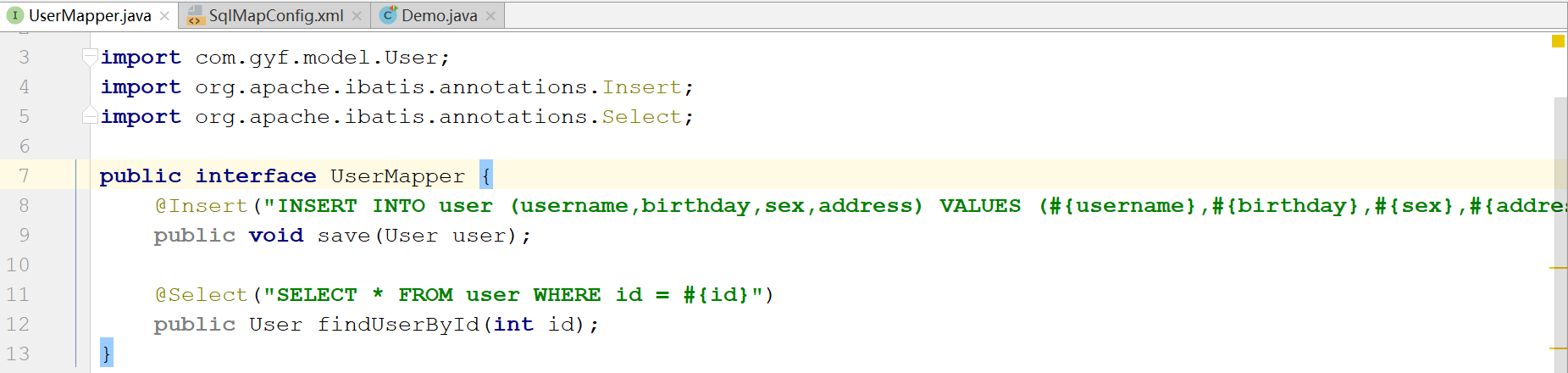
如：<mapper url="file:///D:\workspace\_spingmvc\mybatis\_01\config\sqlmap\User.xml" />

### <mapper class=’’/>

使用mapper接口的全限定名

如：<mapper class="cn.gyf.mybatis.mapper.UserMapper"/>

也可使用注解开发，把xml文件删除



**注意：此种方法要求mapper接口和mapper映射文件要名称相同，且放到同一个目录下**；

### <package name=’’/>（推荐）

注册指定包下的所有映射文件

如：<package name="cn.gyf.mybatis.mapper"/>

**注意：此种方法要求mapper接口和mapper映射文件要名称相同，且放到同一个目录下**；

# Mybatis的映射文件

## 4.1 输入映射ParameterType

指定输入参数的java类型，可以使用别名或者类的全限定名。它可以接收**简单类型,POJO对象、HashMap**。

### 传递简单类型

根据用户ID查询用户信息。

|  |
| --- |
|  |

### 传递POJO对象

|  |
| --- |
|  |

### 传递POJO包装对象

开发中通过pojo传递查询条件 ，查询条件是综合的查询条件，不仅包括用户查询条件还包括其它的查询条件（比如将用户购买商品信息也作为查询条件），这时可以使用包装对象传递输入参数。

**需求**

综合查询用户信息，需要传入查询条件复杂，比如（用户信息、订单信息、商品信息）。

vo:键值对对象，相对于kv

po:persist object 持久化对象

pojo:简单的java对象

entity:实体

#### 定义POJO包装类

|  |
| --- |
|  |

#### 修改UserMapper.java

|  |
| --- |
|  |

#### 修改UsrMappler.xml

|  |
| --- |
|  |

#### 测试

|  |
| --- |
|  |

### 传递Map对象

|  |
| --- |
| 修改UserMapper.java |
|  |
| 修改UserMapper.xml |
|  |
| 测试 |
|  |

## 4.2 输出映射 resultType/resultMap

### resultType

使用resultType进行结果映射时，查询的列名和映射的pojo属性名完全一致，该列才能映射成功。

如果查询的列名和映射的pojo属性名全部不一致，则不会创建pojo对象；

如果查询的列名和映射的pojo属性名有一个一致，就会创建pojo对象。

#### 输出简单类型

当输出结果只有一列时，可以使用ResultType指定简单类型作为输出结果类型。

**案例：输出一个count(\*)**

|  |
| --- |
| **UserMapper.java** |
|  |
| **UserMapper.xml** |
|  |
| 测试 |
|  |

#### 输出POJO单个对象

|  |
| --- |
|  |
|  |

#### 输出POJO列表

|  |
| --- |
|  |
|  |

**总结：**

输出单个pojo对象和pojo列表时，mapper映射文件中的resultType的类型是一样的，mapper接口的方法返回值不同。

同样的mapper映射文件，返回单个对象和对象列表时，mapper接口在生成动态代理的时候，会根据返回值的类型，决定调用selectOne方法还是selectList方法。

### resultMap

如果查询出来的列名和属性名不一致，通过定义一个**resultMap将列名和pojo属性名**之间作一个映射关系。

1. 定义resultMap

2、使用resultMap作为statement的输出映射类型

|  |
| --- |
| UserMapper.java |
|  |
| UserMapper.xml |
|  |
| 测试 |
|  |

# 动态SQL

## 5.1 if和where

* If标签：作为判断入参来使用的，如果符合条件，则把if标签体内的SQL拼接上。

**注意：用if进行判断是否为空时，不仅要判断null，也要判断空字符串‘’；**

* Where标签：会去掉条件中的第一个and符号。

|  |
| --- |
|  |

## 5.2 SQL片断

Mybatis提供了SQL片段的功能，可以提高SQL的**可重用性**。

|  |
| --- |
|  |

## 5.3 foreach 遍历

### 案例：查询指定id的用户

【SELECT \* FROM `user` where id in (31,32,33); 】

|  |
| --- |
| UserQueryVO.java |
|  |
| UserMapper.xml |
|  |
| 测试： |
|  |

### 另一种遍历，参数直接传入id集合【自己练】

|  |
| --- |
|  |

# mybatis与hibernate的区别【面试题】

## Mybatis技术特点：

好处：

1. 通过直接编写SQL语句，可以直接对SQL进行性能的优化；
2. 学习门槛低，学习成本低。只要有SQL基础，就可以学习mybatis，而且很容易上手；
3. 由于直接编写SQL语句，所以灵活多变，代码维护性更好。

缺点：

1. 不能支持数据库无关性，即数据库发生变更，要写多套代码进行支持，移植性不好。
   1. Mysql:limit
   2. Oracle:rownum
2. 需要编写结果映射。

## Hibernate技术特点：

好处：

1. 标准的orm框架，程序员不需要编写SQL语句。
2. 具有良好的数据库无关性，即数据库发生变化的话，代码无需再次编写。
   1. 以后,mysql数据迁移到oracle，只需要改方言配置

缺点:

1. 学习门槛高，需要对数据关系模型有良好的基础，而且在设置OR映射的时候，需要考虑好性能和对象模型的权衡。
2. 程序员不能自主的去进行SQL性能优化。

## Mybatis应用场景：

需求多变的互联网项目，例如电商项目。

## Hibernate应用**场**景：

需求明确、业务固定的项目，例如OA项目、ERP项目等。