mybatis 第一天 mybatis的基础知识

# mybatis框架

## mybatis是什么？

mybatis是一个持久层的框架，是apache下的顶级项目。

mybatis托管到goolecode下，再后来托管到github下(https://github.com/mybatis/mybatis-3/releases)。

mybatis让程序将主要精力放在sql上，通过mybatis提供的映射方式，自由灵活生成（半自动化，大部分需要程序员编写sql）满足需要sql语句。

mybatis可以将向 preparedStatement中的输入参数自动进行输入映射，将查询结果集灵活映射成java对象。（输出映射）

## mybatis框架

mybaits-config.xml（是mybatis的全局配置文件，名称不固定的）

配置了数据源、事务等mybatis运行环境

配置映射文件（配置sql语句）

mapper.xml（映射文件）、mapper.xml、mapper.xml.....

SqlSessionFactory（会话工厂），根据配置文件创建工厂

作用：创建SqlSession

msyql

SqlSession（会话），是一个接口，面向用户（程序员）的接口

作用：操作数据库（发出sql增、删、改、查）

Executor（执行器），是一个接口（基本执行器、缓存执行器）

作用：SqlSession内部通过执行器操作数据库

mapped statement（底层封装对象）

作用：对操作数据库存储封装，包括 sql语句，输入参数、输出结果类型

输入参数类型

java简单类型

hashmap

pojo自定义

输出结果 类型

java简单类型

hashmap

pojo自定义

# 入门程序

## 需求

根据用户id（主键）查询用户信息

根据用户名称模糊查询用户信息

添加用户

删除 用户

更新用户

## 环境

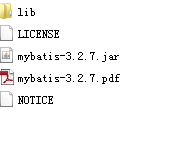
java环境：jdk1.7.0\_72

eclipse：indigo

mysql：5.1

mybatis运行环境（jar包）：

从https://github.com/mybatis/mybatis-3/releases下载，3.2.7版本



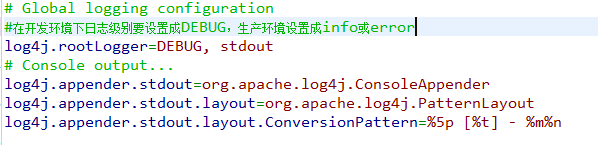
lib下：依赖包

mybatis-3.2.7.jar：核心 包

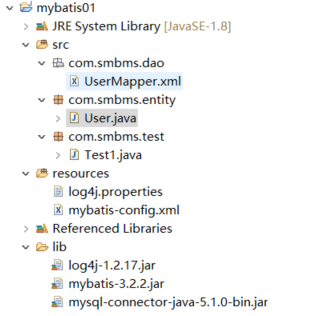
mybatis-3.2.7.pdf，操作指南

加入mysql的驱动包

## log4j.properties



## 工程结构



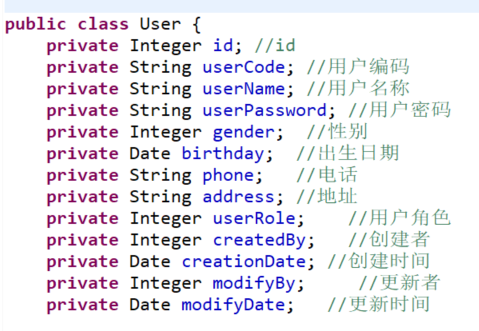
## Mybatis-config.xml

配置mybatis的运行环境，数据源、事务等。



## 根据真实姓名模糊查询用户信息

### 创建实体类

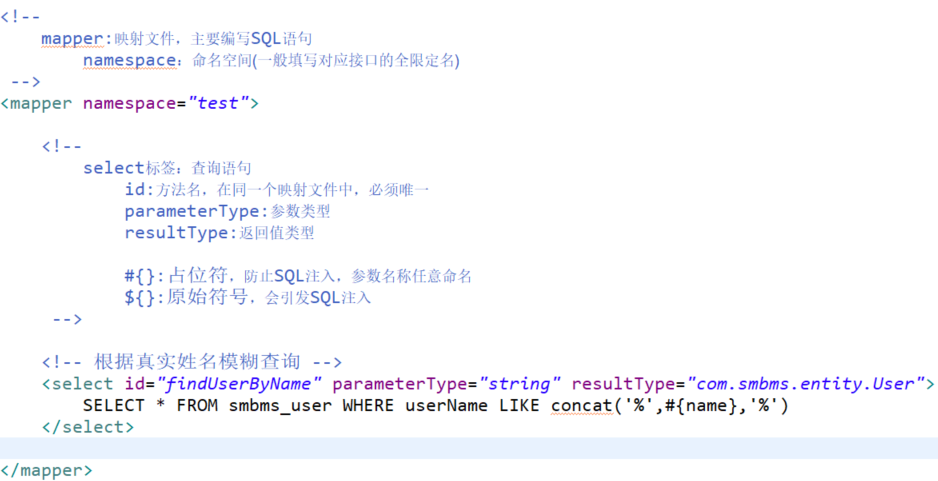


### 映射文件

映射文件命名：

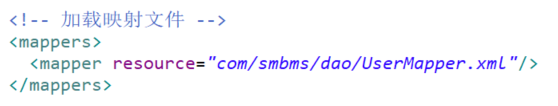
User.xml（原始ibatis命名），mapper代理开发映射文件名称叫XXXMapper.xml，比如：UserMapper.xml、ItemsMapper.xml

在映射文件中配置sql语句。



### 在mybatis-config.xml加载映射文件

在mybatis-config.xml中加载User.xml:



### 程序编写



## 总结

### parameterType

在映射文件中通过parameterType指定输入参数的类型。

### resultType

在映射文件中通过resultType指定输出结果的类型。

### #{}和${}

#{}表示一个占位符号，#{}接收输入参数，类型可以是简单类型，pojo、hashmap。

如果接收简单类型，#{}中可以写成value或其它名称。

#{}接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性...的方式获取对象属性值。

${}表示一个拼接符号，会引用sql注入，所以不建议使用${}。

${}接收输入参数，类型可以是简单类型，pojo、hashmap。

如果接收简单类型，${}中只能写成value。

${}接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性...的方式获取对象属性值。

### selectOne和selectList

selectOne表示查询出一条记录进行映射。如果使用selectOne可以实现使用selectList也可以实现（list中只有一个对象）。

selectList表示查询出一个列表（多条记录）进行映射。如果使用selectList查询多条记录，不能使用selectOne。

如果使用selectOne报错：

org.apache.ibatis.exceptions.TooManyResultsException: Expected one result (or null) to be returned by selectOne(), but found: 4

# mybatis开发dao的方法

## SqlSession使用范围

### SqlSessionFactoryBuilder

通过SqlSessionFactoryBuilder创建会话工厂SqlSessionFactory

将SqlSessionFactoryBuilder**当成一个工具类使用即可**，不需要使用单例管理SqlSessionFactoryBuilder。

在需要创建SqlSessionFactory时候，只需要new一次SqlSessionFactoryBuilder即可。

### SqlSessionFactory

通过SqlSessionFactory创建SqlSession，使用单例模式管理sqlSessionFactory（工厂一旦创建，使用一个实例）。

将来mybatis和spring整合后，使用单例模式管理sqlSessionFactory。

### SqlSession

SqlSession是一个面向用户（程序员）的接口。

SqlSession中提供了很多操作数据库的方法：如：selectOne(返回单个对象)、selectList（返回单个或多个对象）、。

SqlSession是线程不安全的，在SqlSesion实现类中除了有接口中的方法（操作数据库的方法）还有数据域属性。

**SqlSession最佳应用场合在方法体内，定义成局部变量使用。**

## mapper代理方法（只需要mapper接口（相当 于dao接口））

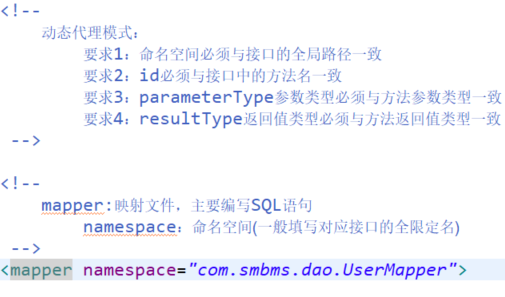
### 思路（mapper代理开发规范）

程序员还需要编写mapper.xml映射文件

程序员编写mapper接口需要遵循一些开发规范，mybatis可以自动生成mapper接口实现类代理对象。

开发规范：

1、在mapper.xml中namespace等于mapper接口地址

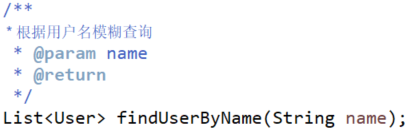


2、mapper.java接口中的方法名和mapper.xml中statement的id一致

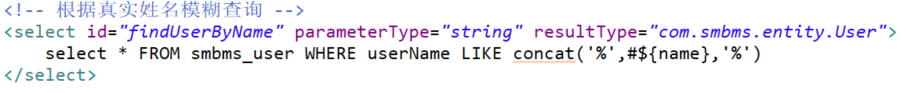
3、mapper.java接口中的方法输入参数类型和mapper.xml中statement的parameterType指定的类型一致。

4、mapper.java接口中的方法返回值类型和mapper.xml中statement的resultType指定的类型一致。

### mapper.java



### mapper.xml



### 在mybatis.xml中加载mapper.xml



### 测试



# mybatis-config.xml

mybatis的全局配置文件SqlMapConfig.xml，配置内容如下：

properties（属性）

settings（全局配置参数）

typeAliases（类型别名）

typeHandlers（类型处理器）

objectFactory（对象工厂）

plugins（插件）

environments（环境集合属性对象）

environment（环境子属性对象）

transactionManager（事务管理）

dataSource（数据源）

mappers（映射器）

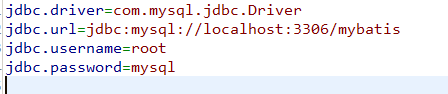
## properties属性

需求：

将数据库连接参数单独配置在db.properties中，只需要在mybatis中加载db.properties的属性值。

在SqlMapConfig.xml中就不需要对数据库连接参数硬编码。

将数据库连接参数只配置在db.properties中，原因：方便对参数进行统一管理，其它xml可以引用该db.properties。



在sqlMapConfig.xml加载属性文件：



properties特性：

注意： MyBatis 将按照下面的顺序来加载属性：

* 在 properties 元素体内定义的属性首先被读取。
* 然后会读取properties 元素中resource或 url 加载的属性，它会覆盖已读取的同名属性。
* 最后读取parameterType传递的属性，它会覆盖已读取的同名属性。

建议：

不要在properties元素体内添加任何属性值，只将属性值定义在properties文件中。

在properties文件中定义属性名要有一定的特殊性，如：XXXXX.XXXXX.XXXX

## settings全局参数配置

mybatis框架在运行时可以调整一些运行参数。

比如：开启二级缓存、开启延迟加载。。

全局参数将会影响mybatis的运行行为。

详细参见“学习资料/mybatis-settings.xlsx”文件







## typeAliases（别名）重点

### 需求

在mapper.xml中，定义很多的statement，statement需要parameterType指定输入参数的类型、需要resultType指定输出结果的映射类型。

如果在指定类型时输入类型全路径，不方便进行开发，可以针对parameterType或resultType指定的类型定义一些别名，在mapper.xml中通过别名定义，方便开发。

### mybatis默认支持别名

|  |  |
| --- | --- |
| 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

#### 单个别名定义

引用别名：

#### 批量定义别名（常用）

## typeHandlers（类型处理器）

mybatis中通过typeHandlers完成jdbc类型和java类型的转换。

通常情况下，mybatis提供的类型处理器满足日常需要，不需要自定义.

mybatis支持类型处理器：

|  |  |  |
| --- | --- | --- |
| 类型处理器 | **Java**类型 | **JDBC**类型 |
| BooleanTypeHandler | Boolean，boolean | 任何兼容的布尔值 |
| ByteTypeHandler | Byte，byte | 任何兼容的数字或字节类型 |
| ShortTypeHandler | Short，short | 任何兼容的数字或短整型 |
| IntegerTypeHandler | Integer，int | 任何兼容的数字和整型 |
| LongTypeHandler | Long，long | 任何兼容的数字或长整型 |
| FloatTypeHandler | Float，float | 任何兼容的数字或单精度浮点型 |
| DoubleTypeHandler | Double，double | 任何兼容的数字或双精度浮点型 |
| BigDecimalTypeHandler | BigDecimal | 任何兼容的数字或十进制小数类型 |
| StringTypeHandler | String | CHAR和VARCHAR类型 |
| ClobTypeHandler | String | CLOB和LONGVARCHAR类型 |
| NStringTypeHandler | String | NVARCHAR和NCHAR类型 |
| NClobTypeHandler | String | NCLOB类型 |
| ByteArrayTypeHandler | byte[] | 任何兼容的字节流类型 |
| BlobTypeHandler | byte[] | BLOB和LONGVARBINARY类型 |
| DateTypeHandler | Date（java.util） | TIMESTAMP类型 |
| DateOnlyTypeHandler | Date（java.util） | DATE类型 |
| TimeOnlyTypeHandler | Date（java.util） | TIME类型 |
| SqlTimestampTypeHandler | Timestamp（java.sql） | TIMESTAMP类型 |
| SqlDateTypeHandler | Date（java.sql） | DATE类型 |
| SqlTimeTypeHandler | Time（java.sql） | TIME类型 |
| ObjectTypeHandler | 任意 | 其他或未指定类型 |
| EnumTypeHandler | Enumeration类型 | VARCHAR-任何兼容的字符串类型，作为代码存储（而不是索引）。 |

## mappers（映射配置）

### 通过resource加载单个映射文件

### 通过mapper接口加载单个mapper

按照上边的规范，将mapper.java和mapper.xml放在一个目录 ，且同名。

### 批量加载mapper(推荐使用)