# 02-MyBatis的配置文件详解



## 0、日志

**日志演变：**

1.4之前 没有任何的日志框架

System.out.prinln("")

小明

1.将日志按照级别输入，按照包或者类来输入。

2.将日志输入到文件中，能不能按照日期或者文件大小来进行归档，记录日志同时发送邮件给开发人员

3.自定义格式，让日志更美观

4.性能

log4j 开源 所有的开发人员一起来维护这个框架。 apache 收入。

jdk JUL java.util.logging , 非常多的日志框架Jboss-logging.....

开发slf4j 日志门面 ， 集成其他框架，不实现日志功能

jdk JCL jakarta common logging

基于log4j开发出来一个logback

apache log4j 开发出来来一个log4j

**市面上的日志框架；**

JUL、JCL、Jboss-logging、logback、log4j、log4j2、slf4j....

| 日志门面 （日志的抽象层） | 日志实现 |
| --- | --- |
| ~~JCL（Jakarta Commons Logging）~~ SLF4j（Simple Logging | Log4j JUL（java.util.logging） Log4j2 |

| Facade for Java） ~~jboss-logging~~ | |
| --- | --- |

左边选一个门面（抽象层）、右边来选一个实现；

日志门面： SLF4J； 官方文档：http://www.slf4j.org/

日志实现：Logback； 中文文档：http://www.logback.cn/

怎么在mybatis中实现呢

1.导入pom

```
1
2
3  <!-- log start -->
4  <dependency>
5  <groupId>org.slf4j</groupId>
6  <artifactId>slf4j-api</artifactId>
7  <version>1.7.30</version>
8  </dependency>
9
10
11  <dependency>
12  <groupId>ch.qos.logback</groupId>
13  <artifactId>logback-classic</artifactId>
14  <version>1.2.3</version>
15  </dependency>
16  <!-- log end -->
```

2.添加logback配置文件

```
1 <configuration>
2  <!--appender 追加器 日志以哪种方式进行输出
3  name 取个名字
4  class 不同实现类会输出到不同地方
5  ch.qos.logback.core.ConsoleAppender 输出到控制台
6  -->
7  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
8  <encoder>
9  <!-- 格式 -->
10  <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{100} - %msg%n</pat
tern>
11  </encoder>
12  </appender>
13 <!--cn.tulingxueyuan.mapper-->
```

```
<!--控制跟细粒度的日志级别 根据包\根据类-->
 <logger name="cn.tulingxueyuan.mapper" level="debug"></logger>
 org.apache.ibatis.transaction
 <!--控制所有的日志级别-->
 <root level="error">
 <!-- 将当前日志级别输出到哪个追加器上面 -->
 <appender-ref ref="STDOUT" />
 </root>
</configuration>
```

```
Logger LOGGER= LoggerFactory.getLogger(this.getClass());
/**
 * 日志级别
 * TRACE < DEBUG < INFO < WARN < ERROR。
 * 1 2 3 4 5
 */
@Test
public void test02(){
 LOGGER.trace("跟踪级别");
 LOGGER.debug("调试级别");
 LOGGER.info("信息级别");
 LOGGER.warn("警告级别");
 LOGGER.error("异常级别");
}
```

# 1、全局配置文件详解

在mybatis的项目中，我们发现了有一个mybatis-config.xml的配置文件，这个配置文件是mybatis的全局配置文件，用来进行相关的全局配置，在任何操作下都生效的配置。下面我们要针对其中的属性做详细的解释，方便大家在后续使用的时候更加熟练。

**官方说明：**

MyBatis 的配置文件包含了会深深影响 MyBatis 行为的设置和属性信息。 配置文档的顶层结构如下：

- configuration（配置）
  - properties（属性）
  - settings（设置）

- typeAliases（类型别名）
- typeHandlers（类型处理器）
- objectFactory（对象工厂）
- plugins（插件）
- environments（环境配置）
  - environment（环境变量）
    - transactionManager（事务管理器）
    - dataSource（数据源）
- databaseIdProvider（数据库厂商标识）
- mappers（映射器）

mybatis-config.xml

```
<?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>
 <!--引入外部配置文件，类似于Spring中的property-placeholder
 resource:从类路径引入
 url:从磁盘路径或者网络路径引入
 -->
 <properties resource="db.properties"></properties>
 <!--用来控制mybatis运行时的行为，是mybatis中的重要配置-->
 <settings>
 <!--设置列名映射的时候是否是驼峰标识-->
 <setting name="mapUnderscoreToCamelCase" value="true"/>
 </settings>
 <!--typeAliases表示为我们引用的实体类起别名，默认情况下我们需要写类的完全限定名
 如果在此处做了配置，那么可以直接写类的名称,在type中配置上类的完全限定名，在使用的时候可以忽略大小写
 还可以通过alias属性来表示类的别名
 -->
 <typeAliases>
<!-- <typeAlias type="cn.tulingxueyuan.bean.Emp" alias="Emp">
</typeAlias>-->
 <!--如果需要引用多个类，那么给每一个类起别名肯定会很麻烦，因此可以指定对应的包名，那么默认用的是类名-->
 <package name="cn.tulingxueyuan.bean"/>
```

```
</typeAliases>
<!--
在实际的开发过程中，我们可能分为开发环境，生产环境，测试环境等等，每个环境的配置可以是不一样的
environment就用来表示不同环境的细节配置，每一个环境中都需要一个事务管理器以及数据源的配置
我们在后续的项目开发中几乎都是使用spring中配置的数据源和事务管理器来配置，此处不需要研究
-->
<!--default:用来选择需要的环境-->
<environments default="development">
<!--id:表示不同环境的名称-->
<environment id="development">
<transactionManager type="JDBC"/>
<!--配置数据库连接-->
<dataSource type="POOLED">
<!--使用${}来引入外部变量-->
<property name="driver" value="${driverClassname}"/>
<property name="url" value="${url}"/>
<property name="username" value="${username}"/>
<property name="password" value="${password}"/>
</dataSource>
</environment>
</environments>
<!--
在不同的数据库中，可能sql语句的写法是不一样的，为了增强移植性，可以提供不同数据库的操作实现
在编写不同的sql语句的时候，可以指定databaseId属性来标识当前sql语句可以运行在哪个数据库中
-->
<databaseIdProvider type="DB_VENDOR">
<property name="MySQL" value="mysql"/>
<property name="SQL Server" value="sqlserver"/>
<property name="Oracle" value="orcl"/>
</databaseIdProvider>

<!--将sql的映射文件适用mappers进行映射-->
<mappers>
<!--
指定具体的不同的配置文件
class:直接引入接口的全类名，可以将xml文件放在dao的同级目录下，并且设置相同的文件名称，同时可以使用注解的方式来进行相关的配置
```

```
60   url:可以从磁盘或者网络路径查找sql映射文件
61   resource:在类路径下寻找sql映射文件
62   -->
63  <!-- <mapper resource="EmpDao.xml"/>
64   <mapper resource="UserDao.xml"/>
65   <mapper class="cn.tulingxueyuan.dao.EmpDaoAnnotation"></mapper>-->
66   <!--
67   当包含多个配置文件或者配置类的时候，可以使用批量注册的功能，也就是引入对应的包，而不是具体的配置文件或者类
68   但是需要注意的是，
69   1、如果使用的配置文件的形式，必须要将配置文件跟dao类放在一起，这样才能找到对应的配置文件.
70   如果是maven的项目的话，还需要添加以下配置，原因是maven在编译的文件的时候只会编译java文件
71   <build>
72   <resources>
73   <resource>
74   <directory>src/main/java</directory>
75   <includes>
76   <include>**/*.xml</include>
77   </includes>
78   </resource>
79   </resources>
80   </build>
81
82   2、将配置文件在resources资源路径下创建跟dao相同的包名
83   -->
84   <package name="cn.tulingxueyuan.dao"/>
85   </mappers>
86  </configuration>
```

# 02、Mybatis SQL映射文件详解

MyBatis 的真正强大在于它的语句映射，这是它的魔力所在。由于它的异常强大，映射器的 XML 文件就显得相对简单。如果拿它跟具有相同功能的 JDBC 代码进行对比，你会立即发现省掉了将近 95% 的代码。MyBatis 致力于减少使用成本，让用户能更专注于 SQL 代码。

SQL 映射文件只有很少的几个顶级元素（按照应被定义的顺序列出）：

- `cache` – 该命名空间的缓存配置。

- `cache-ref` – 引用其它命名空间的缓存配置。
- `resultMap` – 描述如何从数据库结果集中加载对象，是最复杂也是最强大的元素。
- ~~`parameterMap` – 老式风格的参数映射。此元素已被废弃，并可能在将来被移除！请使用行内参数映射。文档中不会介绍此元素。~~
- `sql` – 可被其它语句引用的可重用语句块。
- `insert` – 映射插入语句。
- `update` – 映射更新语句。
- `delete` – 映射删除语句。
- `select` – 映射查询语句。

在每个顶级元素标签中可以添加很多个属性，下面我们开始详细了解下具体的配置。

## 1、insert、update、delete元素

| 属性 | 描述 |
| --- | --- |
| id | 在命名空间中唯一的标识符，可以被用来 |
| parameterType | 将会传入这条语句的参数的类全限定名或 的，因为 MyBatis 可以通过类型处理器 出具体传入语句的参数，默认值为未设置 |
| parameterMap | 用于引用外部 parameterMap 的属性， 行内参数映射和 parameterType 属性。 |
| flushCache | 将其设置为 true 后，只要语句被调用， 级缓存被清空，默认值：（对 insert、u true。 |
| timeout | 这个设置是在抛出异常之前，驱动程序等 的秒数。默认值为未设置（unset）（依 |
| statementType | 可选 STATEMENT，PREPARED 或 CA MyBatis 分别使用 Statement，Prepar CallableStatement，默认值：PREPAR |
| useGeneratedKeys | （仅适用于 insert 和 update）这会令 getGeneratedKeys 方法来取出由数据库 如：像 MySQL 和 SQL Server 这样的关 自动递增字段），默认值：false。 |
| keyProperty | （仅适用于 insert 和 update）指定能唯 MyBatis 会使用 getGeneratedKeys 的 selectKey 子元素设置它的值，默认值： 生成列不止一个，可以用逗号分隔多个属 |
| keyColumn | （仅适用于 insert 和 update）设置生成 某些数据库（像 PostgreSQL）中，当主 的时候，是必须设置的。如果生成列不止 多个属性名称。 |
| databaseId | 如果配置了数据库厂商标识（database 会加载所有不带 databaseId 或匹配当前 如果带和不带的语句都有，则不带的会被 |

```xml
<!--如果数据库支持自增可以使用这样的方式-->
   <insert id="insertUser" useGeneratedKeys="true" keyProperty="id">
      insert into user(user_name) values(#{userName})
   </insert>
   <!--如果数据库不支持自增的话，那么可以使用如下的方式进行赋值查询-->
   <insert id="insertUser2" >
       <selectKey order="BEFORE" keyProperty="id" resultType="integer">
          select max(id)+1 from user
       </selectKey>
       insert into user(id,user_name) values(#{id},#{userName})
    </insert>
```

SQL映射文件内容过多 笔记将分别记录于--《MyBatis基于XML的详细使用》

## 面试题：

### 1. Mybatis都有哪些Executor执行器？它们之间的区别是什么？

### 2. ORM是什么？

### 3. 为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？

### 4. Mybatis之Mapper接口的实现原理