**Mybatis**

**目录**

[概述 1](#_Toc27908)

[如何下载Mybatis 1](#_Toc484)

[JDBC的缺陷 2](#_Toc15564)

[结果集映射（resultMap） 2](#_Toc4261)

[Log4j 3](#_Toc29657)

[如何使用Log4j 3](#_Toc10416)

[Lombok 5](#_Toc16066)

[Lombok常用注解 5](#_Toc12025)

[使用方法 6](#_Toc5757)

[动态sql 6](#_Toc2342)

[sql片段 7](#_Toc122)

[缓存 7](#_Toc3011)

**概述**

MyBatis是持久层的框架，Mybatis框架避免了几乎所有的jdbc代码、手动设置参数，以及获取结果集。Mybatis是apache的一个开源项目。mybatis是一个sql映射框架，提供对数据库的操作功能，使用mybatis框架可以让开发人员集中精神写sql就行了，不必关心Connecttion、Statement、ResultSet的创建、销毁、sql的执行。

**如何下载Mybatis**

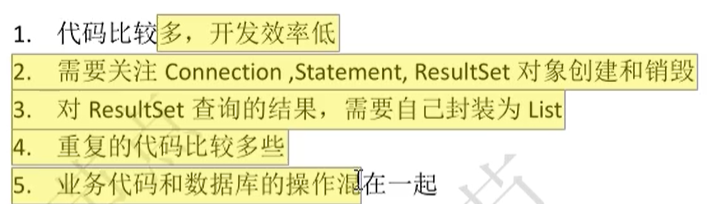
Maven仓库（https://mvnrepository.com/）

<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.7</version>  
</dependency>

Github（https://github.com/mybatis/mybatis-3）

Mybatis3中文文档（https://mybatis.org/mybatis-3/zh/getting-started.html）

**JDBC的缺陷**



Mybatis框架的出现，解决了JDBC的缺陷。Mybatis默认采用JDBC事务管理，使用JDBC提交和回滚事务。Mybatis还可以采用Managed事务管理。数据库中的每一个表都要对应一个实体类。

**结果集映射（resultMap）**

通过结果集映射可以解决因表中的列名与JavaBean的属性名不匹配而导致的赋值失败的问题。



**Log4j**

Log4j是[Apache](https://baike.baidu.com/item/Apache/8512995" \t "https://baike.baidu.com/item/log4j/_blank)的一个开源项目，通过使用Log4j，我们可以控制日志信息输送的目的地是[控制台](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%8F%B0/2438626" \t "https://baike.baidu.com/item/log4j/_blank)、文件、[GUI](https://baike.baidu.com/item/GUI" \t "https://baike.baidu.com/item/log4j/_blank)组件，甚至是套接口服务器、[NT](https://baike.baidu.com/item/NT/3443842" \t "https://baike.baidu.com/item/log4j/_blank)的事件记录器、[UNIX](https://baike.baidu.com/item/UNIX" \t "https://baike.baidu.com/item/log4j/_blank) [Syslog](https://baike.baidu.com/item/Syslog" \t "https://baike.baidu.com/item/log4j/_blank)[守护进程](https://baike.baidu.com/item/%E5%AE%88%E6%8A%A4%E8%BF%9B%E7%A8%8B/966835" \t "https://baike.baidu.com/item/log4j/_blank)等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。最令人感兴趣的就是，这些可以通过一个[配置文件](https://baike.baidu.com/item/%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6/286550" \t "https://baike.baidu.com/item/log4j/_blank)来灵活地进行配置，而不需要修改应用的代码。

**如何使用Log4j**

第一步: 导入log4j依赖

<dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
</dependency>

第二步：在CLASSPATH（类路径）下创建log4j.properties。内容如下:

#将等级为DEBUG的日志信息输出到console和file这两个目的地，console和file的定义在下面的代码  
log4j.rootLogger=DEBUG,console,file  
  
#控制台输出的相关设置  
log4j.appender.console = org.apache.log4j.ConsoleAppender  
log4j.appender.console.Target = System.out  
log4j.appender.console.Threshold=DEBUG  
log4j.appender.console.layout = org.apache.log4j.PatternLayout  
log4j.appender.console.layout.ConversionPattern=[%c]-%m%n  
  
#文件输出的相关设置  
log4j.appender.file = org.apache.log4j.RollingFileAppender  
log4j.appender.file.File=./log/lun.log  
log4j.appender.file.MaxFileSize=10mb  
log4j.appender.file.Threshold=DEBUG  
log4j.appender.file.layout=org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern=[%p][%d{yy-MM-dd}][%c]%m%n  
  
#日志输出级别  
log4j.logger.org.mybatis=DEBUG  
log4j.logger.java.sql=DEBUG  
log4j.logger.java.sql.Statement=DEBUG  
log4j.logger.java.sql.ResultSet=DEBUG  
log4j.logger.java.sql.PreparedStatement=DEBUG

第三步：在mybatis主配置文件中设置Log4j为日志的实现

<settings>  
 <!-- log4j日志工厂的实现 -->  
 <setting name="logImpl" value="LOG4J"/>  
</settings>

**Lombok**

Lombok项目是一个Java库，它会自动插入编辑器和构建工具中，Lombok提供了一组有用的注释，用来消除Java类中的大量样板代码。仅五个字符(@Data)就可以替换数百行代码从而产生干净，简洁且易于维护的Java类。

**Lombok常用注解**

@Setter ：注解在类或字段，注解在类时为所有字段生成setter方法，注解在字段上时只为该字段生成setter方法。

@Getter ：使用方法同上，区别在于生成的是getter方法。

@ToString ：注解在类，添加toString方法。

@EqualsAndHashCode： 注解在类，生成hashCode和equals方法。

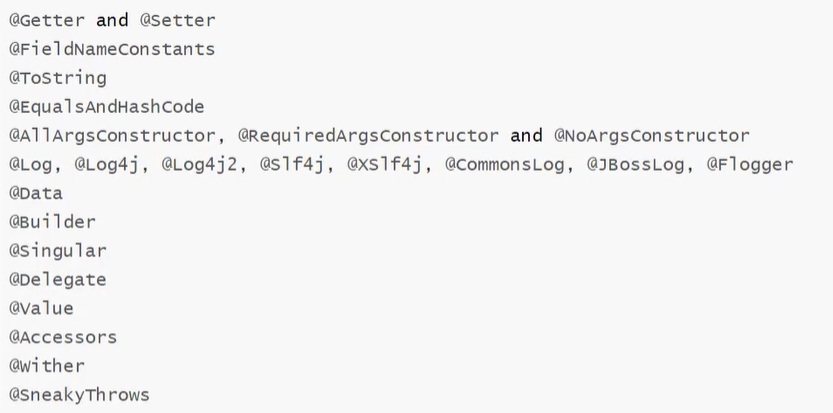
@NoArgsConstructor： 注解在类，生成无参的构造方法。

@RequiredArgsConstructor： 注解在类，为类中需要特殊处理的字段生成构造方法，比如final和被@NonNull注解的字段。

@AllArgsConstructor： 注解在类，生成包含类中所有字段的构造方法。

@Data： 注解在类，生成setter/getter、equals、canEqual、hashCode、toString方法，如为final属性，则不会为该属性生成setter方法。

@Slf4j： 注解在类，生成log变量，严格意义来说是常量。



**使用方法**

第一步：在IDAE中安装lombok插件

第二步：在项目中导入lomnbok依赖

注: 不建议使用，知道有这么一个插件即可。

**动态sql**

动态 SQL 是 MyBatis 的强大特性之一。如果你使用过 JDBC 或其它类似的框架，你应该能理解根据不同条件拼接 SQL 语句有多痛苦，例如拼接时要确保不能忘记添加必要的空格，还要注意去掉列表最后一个列名的逗号。利用动态 SQL，可以彻底摆脱这种痛苦。动态sql就是根据不同的条件生成不同的sql，动态sql，本质上还是sql语句，只是我们可以在sql语句的基础上，增加一些条件判断逻辑。

**sql片段**

有时候，我们可以将出现频率较高的sql语句抽取出来，方便复用。

第一步: 使用sql标签抽取公共部分

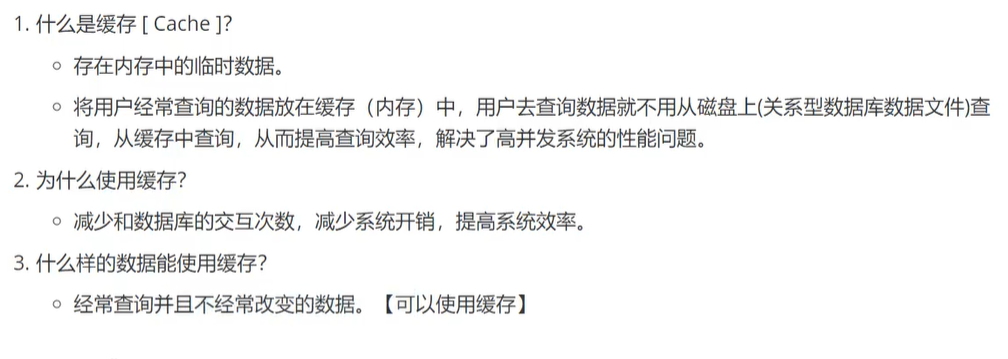
<sql id="if-author-title">  
 <if test="author != null">  
 author = #{author}  
 </if>  
 <if test="title != null">  
 and title = #{title}  
 </if>  
</sql>

第二步：在需要使用sql片段的地方，使用include标签引用即可。

<select id="queryBlogByIf" parameterType="map" resultType="blog">  
 select \* from blog  
 <where>  
 <include refid="if-author-title" />  
 </where>  
</select>

注意: 最好基于单表来定义sql片段。sql片段中最好不要包含<where>、<set>等标签。

**缓存**







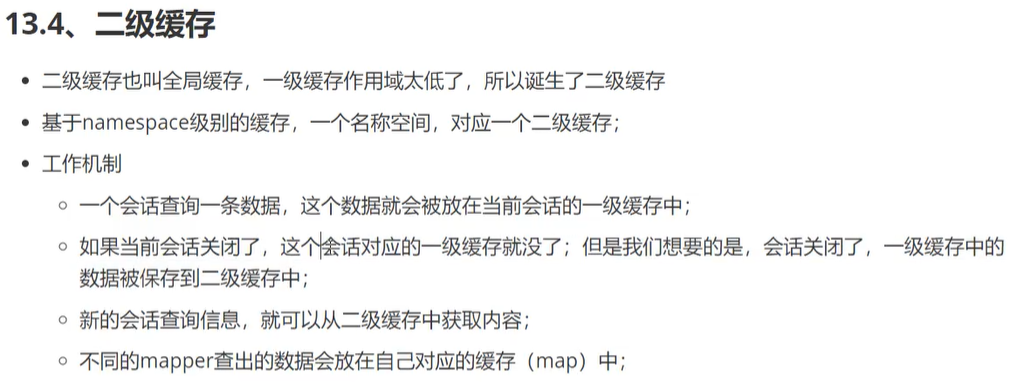
导致本地缓存失效的几种情况

1. 查询内容不同
2. 增删改操作，会改变数据库中原来的数据，所以为了保持数据的一致性，会

重新刷新缓存。

1. 清空本地缓存，即sqlSession.clearCache();

总结：一级缓存是默认开启的，只对同一次sqlSession中查询到的数据进行缓存，即被缓存的数据，只在同一次sqlSession中有效，作用域小。



开启二级缓存后的查询顺序：开启二级缓存后,再执行查询语句,会先从二级缓存中找要查询的数据,二级缓存中没有找到，再从被本地缓存中找，如果本地缓存中也没查找到,说明这条查询语句之前没被执行(新的查询语句)，因此没被缓存，

所以最终从数据库中找要查询的数据。图解如下:

