日志记录

MyBatis 通过使用内部日志工厂提供日志信息。内部日志工厂会将日志信息委派给以下日志实现之一

- SLF4J
- Apache Commons Logging
- Log4j 2
- Log4j（自 3.5.9 起弃用）
- JDK 日志记录

所选的日志解决方案基于内部 MyBatis 日志工厂的运行时自省。MyBatis 日志工厂将使用它找到的第一个日志实现（按上述顺序搜索实现）。如果 MyBatis 未找到上述任何实现，则会禁用日志记录。

许多环境将 Commons Logging 作为应用程序服务器类路径的一部分（好的示例包括 Tomcat 和 WebSphere）。了解在这些环境中，MyBatis 会将 Commons Logging 用作日志实现非常重要。在 WebSphere 这样的环境中，这意味着你的 Log4J 配置将被忽略，因为 WebSphere 提供了自己的 Commons Logging 专有实现。这可能会非常令人沮丧，因为它看起来 MyBatis 正在忽略你的 Log4J 配置（事实上，MyBatis 正在忽略你的 Log4J 配置，因为 MyBatis 会在这些环境中使用 Commons Logging）。如果你的应用程序在类路径中包含 Commons Logging 的环境中运行，但你更愿意使用其他日志实现，则可以通过在 mybatis-config.xml 文件中添加设置来选择不同的日志实现，如下所示

```
<configuration>
  <settings>
    ...
    <setting name="logImpl" value="LOG4J"/>
    ...
  </settings>
</configuration>
```

有效值包括 SLF4J、LOG4J、LOG4J2、JDK_LOGGING、COMMONS_LOGGING、STDOUT_LOGGING、NO_LOGGING 或实现 `org.apache.ibatis.logging.Log` 并获取字符串作为构造函数参数的完全限定类名。

你还可以通过调用以下方法之一来选择实现

```
org.apache.ibatis.logging.LogFactory.useSlf4jLogging();
org.apache.ibatis.logging.LogFactory.useLog4JLogging();
org.apache.ibatis.logging.LogFactory.useLog4J2Logging();
org.apache.ibatis.logging.LogFactory.useJdkLogging();
org.apache.ibatis.logging.LogFactory.useCommonsLogging();
org.apache.ibatis.logging.LogFactory.useStdOutLogging();
```

如果你选择调用其中一个方法，则应在调用任何其他 MyBatis 方法之前这样做。此外，这些方法仅当该实现可在运行时类路径中时才会切换到请求的日志实现。例如，如果你尝试选择 Log4J2 日志记录，而 Log4J2 在运行时不可用，则 MyBatis 将忽略使用 Log4J2 的请求，并使用其用于发现日志实现的常规算法。

SLF4J、Apache Commons Logging、Apache Log4J 和 JDK Logging API 的具体内容超出了本文档的范围。但是，以下示例配置应该可以帮助你入门。如果你想了解有关这些框架的更多信息，可以从以下位置获取更多信息

- SLF4J
- Apache Commons Logging
- Apache Log4j 2.x
- JDK 日志记录 API

日志记录配置

要查看 MyBatis 日志记录语句，您可以在包、映射器完全限定类名、命名空间或完全限定语句名称上启用日志记录。

同样，您执行此操作的方式取决于所使用的日志记录实现。我们将展示如何使用 SLF4J(Logback) 执行此操作。配置日志记录服务仅仅是包含一个或多个额外配置文件（例如 `logback.xml`）以及有时一个新的 JAR 文件的问题。以下示例配置将使用 SLF4J(Logback) 作为提供程序配置完整的日志记录服务。共有 2 个步骤。

步骤 1：添加 SLF4J + Logback JAR 文件

因为我们正在使用 SLF4J(Logback)，所以我们需要确保其 JAR 文件可供我们的应用程序使用。要使用 SLF4J(Logback)，您需要将 JAR 文件添加到应用程序类路径中。

对于 Web 或企业应用程序，您可以将 `logback-classic.jar`、`logback-core.jar` 和 `slf4j-api.jar` 添加到 `WEB-INF/lib` 目录中，或者对于独立应用程序，您可以简单地将其添加到 JVM `-classpath` 启动参数中。

如果您使用 maven，则可以通过在 `pom.xml` 上添加以下设置来下载 jar 文件。

```
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.x.x</version>
</dependency>
```

步骤 2：配置 Logback

配置 Logback 很简单。假设您要启用此映射器的日志记录

```
package org.mybatis.example;
public interface BlogMapper {
  @Select("SELECT * FROM blog WHERE id = #{id}")
  Blog selectBlog(int id);
}
```

创建一个名为 `logback.xml` 的文件，如下所示，并将其放在您的类路径中

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>

  <appender name="stdout" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%5level [%thread] - %msg%n</pattern>
    </encoder>
  </appender>

  <logger name="org.mybatis.example.BlogMapper">
    <level value="trace"/>
  </logger>
  <root level="error">
    <appender-ref ref="stdout"/>
  </root>

</configuration>
```

上述文件将导致 SLF4J(Logback) 报告 `org.mybatis.example.BlogMapper` 的详细日志记录，而仅报告应用程序其余类的错误。

如果您想以更精细的级别调整日志记录，则可以针对特定语句而不是整个映射器文件启用日志记录。以下行将仅为 `selectBlog` 语句启用日志记录

```
<logger name="org.mybatis.example.BlogMapper.selectBlog">
  <level value="trace"/>
</logger>
```

相反，您可能希望为一组映射器启用日志记录。在这种情况下，您应将映射器所在的根包添加为记录器

```
<logger name="org.mybatis.example">
  <level value="trace"/>
</logger>
```

有些查询可以返回巨大的结果集。在这些情况下，您可能希望查看 SQL 语句，但不要查看结果。为此，SQL 语句记录在 DEBUG 级别（JDK 日志记录中的 FINE），结果记录在 TRACE 级别（JDK 日志记录中的 FINER），因此，如果您想查看语句但不想查看结果，请将级别设置为 DEBUG。

```
<logger name="org.mybatis.example">
  <level value="debug"/>
</logger>
```

但是，如果您不使用映射器接口而是使用此类映射器 XML 文件，该怎么办？

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.ac.cn//DTD Mapper 3.0//EN"
  "https://mybatis.ac.cn/dtd/mybatis-3-mapper.dtd">
<mapper namespace="org.mybatis.example.BlogMapper">
  <select id="selectBlog" resultType="blog">
    select * from Blog where id = #{id}
  </select>
</mapper>
```

在这种情况下，您可以通过为命名空间添加记录器（如下所示）来启用整个 XML 文件的日志记录

```
<logger name="org.mybatis.example.BlogMapper">
  <level value="trace"/>
</logger>
```

或针对特定语句

```
<logger name="org.mybatis.example.BlogMapper.selectBlog">
  <level value="trace"/>
</logger>
```

是的，您可能已经注意到，为映射器接口或 XML 映射器文件配置日志记录时没有区别。

如果您正在使用 SLF4J 或 Log4j 2，MyBatis 将使用标记 `MYBATIS` 调用它。

`logback.xml` 文件中的其余配置用于配置附加程序，这超出了本文档的范围。但是，您可以在 Logback 网站上找到更多信息。或者，您也可以简单地对其进行试验，看看不同的配置选项有什么效果。

Log4j 2 的配置示例

```
<!-- pom.xml -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.x.x</version>
</dependency>
```

```
<!-- log4j2.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<Configuration xmlns="https://logging.apache.ac.cn/log4j/2.0/config">

  <Appenders>
    <Console name="stdout" target="SYSTEM_OUT">
      <PatternLayout pattern="%5level [%t] - %msg%n"/>
    </Console>
  </Appenders>

  <Loggers>
```

```
    <Logger name="org.mybatis.example.BlogMapper" level="trace"/>
    <Root level="error" >
      <AppenderRef ref="stdout"/>
    </Root>
  </Loggers>

</Configuration>
```

Log4j 的配置示例

```
<!-- pom.xml -->
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

```
# log4j.properties
log4j.rootLogger=ERROR, stdout

log4j.logger.org.mybatis.example.BlogMapper=TRACE

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

JDK 日志记录的配置示例

```
# logging.properties
handlers=java.util.logging.ConsoleHandler
.level=SEVERE

org.mybatis.example.BlogMapper=FINER

java.util.logging.ConsoleHandler.level=ALL
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter
java.util.logging.SimpleFormatter.format=%1$tT.%1$tL %4$s %3$s - %5$s%6$s%n
```