# SpringBoot的MyBatis工程配置

2021年10月6日 21:50

## 以ResultfulDemoMyBatis为例

application.yaml

```yaml
management:
  endpoints:
    web:
      exposure:
        include: "*"

server:
  port: 8082

spring:
  boot:
    admin:
      client:
        url: http://localhost:8081
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://172.16.3.5:3306/oomall_demo?serverTimezone=GMT%2B8
    username: demouser
    password: 123456
    type: com.alibaba.druid.pool.DruidDataSource
    initialization-mode: always
    schema: classpath:schema.sql
    data: classpath:data.sql
    druid:
      #初始化时建立物理连接的个数
      initial-size: 3
      #最小连接池数量
      min-idle: 3
      #最大连接池数量
      max-active: 100
      #获取连接时最大等待时间
      max-wait: 60000
      #配置监控页面访问登录名及密码
      stat-view-servlet:
        login-username: admin
        login-password: 123456
        enabled: true
        url-pattern: /druid/*

      filter:
        stat:
          #是否开启慢sql查询监控
          log-slow-sql: true
          #慢SQL执行时间
          slow-sql-millis: 100
      validation-query: select 1
    sql-script-encoding: UTF-8

mybatis:
  #domain对象的包
  type-aliases-package: cn.edu.xmu.restfuldemo.model
  #mapper.xml所在的位置
  mapper-locations: classpath:cn.edu.xmu.restfuldemo.mapper/*.xml
  #自动将SQL中查出来的带下划线的字段，转换为驼峰标志，再去匹配类中的属性
  configuration:
    #输出产生的SQL
    #log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
```

新增的一块，用于数据库连接

连接驱动，因为我们用的是jdbc，所以这里使用jdbc驱动
需要事先在pom中引入jdbc依赖（mysql-connector-java）

我们不是直接去连接数据库，而是利用一个连接池
连接池会提前建立好一些连接，在使用SQL时就不需要每使用一次就建立和关闭一次连接，而是直接从连接池中拿一个连接过来，然后去运行SQL语句

schema：建表sql语句
data：初始化数据sql语句

如果连接数量超过最大连接数，就需要设置一个最大等待时间

超过100ms的SQL语句会被记录到日志中去
（这里我们使用logback）

## logback日志文件配置

（之前的log4j配置也可以作为参考）

logback.xml

logback.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="false">

    <property name="LOG_HOME" value="logs" />

    <!-- 控制台输出 -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <!--格式化输出：%d表示日期，%thread表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n是换行符-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
```

这个用来定义日志文件的输出目录
现在我们是定义在当前目录下的logs目录，如果我们需要部署到服务器上，就需要修改这个配置项，修改为服务器的某一个目录下

在日志文件中最主要的是Appender，Appender是用来输出日志的
第一个Appender是控制台输出，即我们在Console上看到的内容
中间我们定义了输出格式和输出所用的字符集

logback.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="false">

    <property name="LOG_HOME" value="logs" />

    <!-- 控制台输出 -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <!--格式化输出：%d表示日期，%thread表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n是换行符-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
    </appender>
    <!-- 按照每天生成日志文件 -->
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--日志文件输出的文件名-->
            <FileNamePattern>${LOG_HOME}/restfuldemo.log.%d{yyyy-MM-dd}.log</FileNamePattern>
            <!--日志文件保留天数-->
            <MaxHistory>30</MaxHistory>
        </rollingPolicy>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <!--格式化输出：%d表示日期，%thread表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n是换行符-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <!--日志文件最大的大小-->
        <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <MaxFileSize>10MB</MaxFileSize>
        </triggeringPolicy>
    </appender>

    <appender name="DruidFILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${LOG_HOME}/log_druid_slow_sql.log</file>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- 归档的日志文件的路径，例如今天是2013-12-21日志，当前写的日志文件路径为file节点指定，可以将此文件与file指定文件路径设置为
            不同路径，从而将当前日志文件或归档日志文件置不同的目录。
            而2013-12-21的日志文件在由fileNamePattern指定。%d{yyyy-MM-dd}指定日期格式，%i指定索引 -->
            <fileNamePattern>${LOG_HOME}/restfuldemo.log-druid_slow_sql-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
            <!-- 除按日志记录之外，还配置了日志文件不能超过2M，若超过2M，日志文件会以索引0开始，
            命名日志文件，例如log-error-2013-12-21.0.log -->
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>20MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>
        <!-- 追加方式记录日志 -->
        <append>true</append>
        <!-- 日志文件的格式 -->
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %logger Line:%-3L - %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <!-- 此日志文件只记录级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>ERROR</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>

    <logger name="com.alibaba.druid.filter.stat.StatFilter" level="ERROR">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="DruidFILE" />
    </logger>

    <!-- 日志输出级别 -->
    <logger name="cn.edu.xmu.restfuldemo" level = "INFO"/>
    <root level="ERROR">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

这个用来定义日志文件的输出目录
现在我们是定义在当前目录下的logs目录，如果我们需要部署到服务器上，就需要修改这个配置项，修改为服务器的某一个目录下

在日志文件中最主要的是Appender，Appender是用来输出日志的
第一个Appender是控制台输出，即我们在Console上看到的内容
中间我们定义了输出格式和输出所用的字符集

第二个Appender是输出到文件的Appender，主要是按照时间原则生成在系统这一级别的日志文件
它是按照每天的方式来生成日志文件，保留三十天
triggeringPolicy定义了日志文件的大小，超过规定大小则开一个新文件

第三个Appender是关于Druid的文件的输出
这里我们定义了file标签，是为了记录目前我们看到的所有慢SQL
所以这个file会把当前正在记录的慢SQL写在日志文件中

这里我们为第三个Appender增加了一个filter标签
level表示这个Appender只记录ERROR级别的

logger标签
用name属性指定某个文件或文件夹输出的日志级别，并通过Appender指定日志的输出格式
这里我们指定Druid中的慢SQL输出到控制台和Druid日志中

用root为默认的日志输出做定义
这里我们定义默认的日志文件会输出到控制台和文件中

## pom文件中新加的几项

```xml
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper-spring-boot-starter</artifactId>
    <version>1.2.13</version>
</dependency>
```

page-helper用于分页
当我们要查询多条语句时，这个依赖可以帮助我们分页，这样就不会从数据库中一次拿太多数据

```xml
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.20</version>
</dependency>
```

jdbc的驱动

```xml
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.1.2</version>
</dependency>
```

MyBatis需要用到的依赖

```xml
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid-spring-boot-starter</artifactId>
    <version>1.1.22</version>
</dependency>
```

Druid需要用到的依赖