Mybatis

2021年7月12日 16:54

MyBatis入门使用

内容	说明	重要程度
MyBatis框架介绍	介绍框架与MyBatis的概念	※※
MyBatis开发流程	详细讲解MyBatis六步开发流程	※※※※※
MyBatis使用细则	讲解MyBatis开发中的各种细节	※※※※※
MyBatis工作流程	讲解MyBatis的内部执行过程	※※※

MyBatis高级特性

内容	说明	重要程度
MyBatis日志管理	MyBaits日志管理配置	※※
动态SQL处理	多条件查询下SQL的动态执行过程	※※※※※
MyBatis缓存机制	介绍MyBatis一二级缓存作用与配置	※※※※※
多表级联查询	配置MyBatis多表级联查询	※※
PageHelper插件	MyBatis分页插件的使用办法	※※※※※

目录

MyBatis进阶目录

SSM的解释

Spring 框架的框架，对系统中的各个对象，进行有效的管理，是框架的框架。所有框架基于spring这个底层框架进行开发；

Spring MVC 替代servelet，提到web层面的开发；

MyBatis简化对数据库的操作；

mybatis的介绍

是java中用于简化数据库操作的框架；框架是可被应用开发者定制的应用骨架；框架是一种规则，保证开发者遵循相同的方式开发程序；框架提倡“不要重复造轮子”，对基础功能进行封装。

MyBatis是优秀的持久层框架；使用XML将SQL与程序解耦，便于维护；学习简单，是JDBC的延伸。

MyBatis开发流程

单元测试与Junit4

一、单元测试

JUnit使用方法

1、引入Junit Jar包或增加Maven依赖

单元测试与Junit4

一、单元测试

单元测试是指对软件中的最小可测试单元进行检查和验证

测试用例是指编写一段代码对已有功能（方法）进行校验

JUnit4是Java中最著名的单元测试工具，主流IDE内置支持

JUnit使用方法

1、引入Junit Jar包或增加Maven依赖

2、编写测试用例验证目标方法是否正确运行

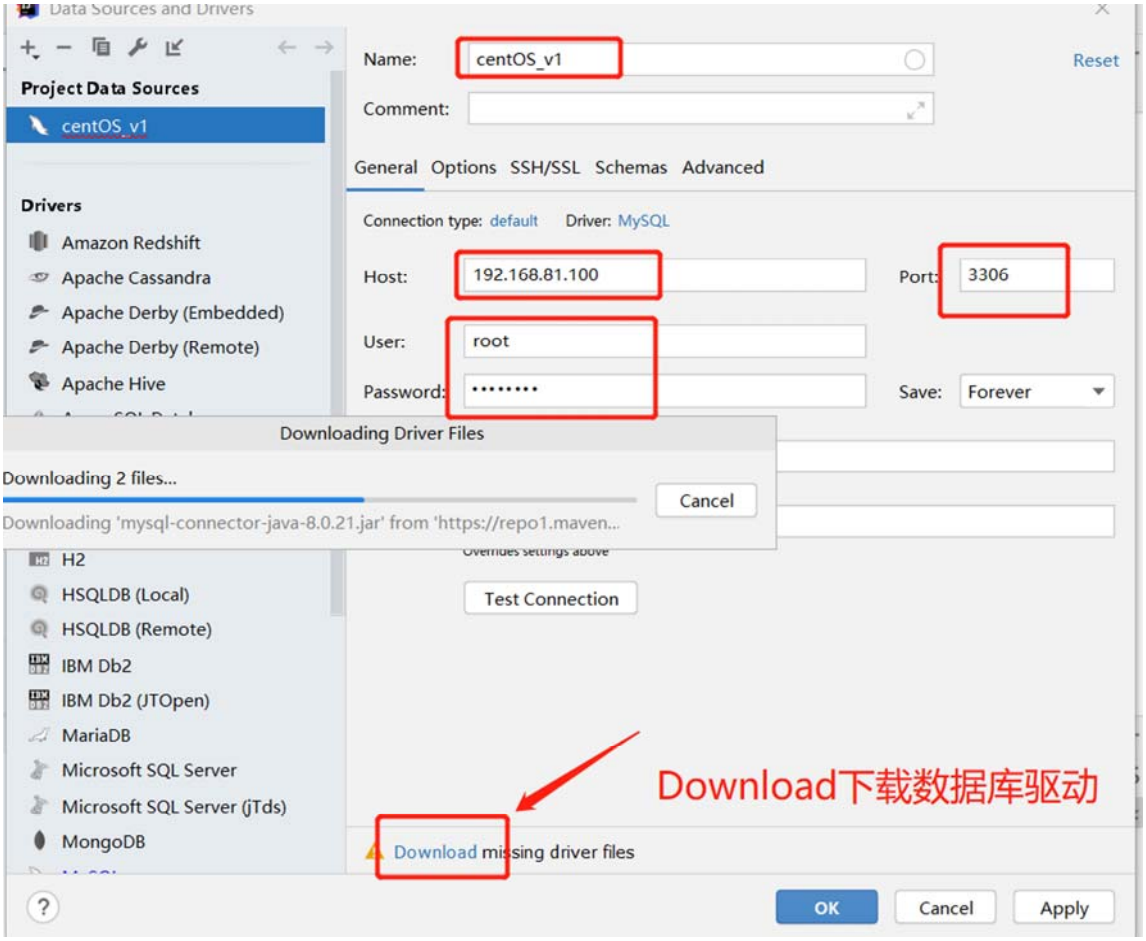
3、在测试用例上增加@Test注解开始单元测试

MyBatis环境配置

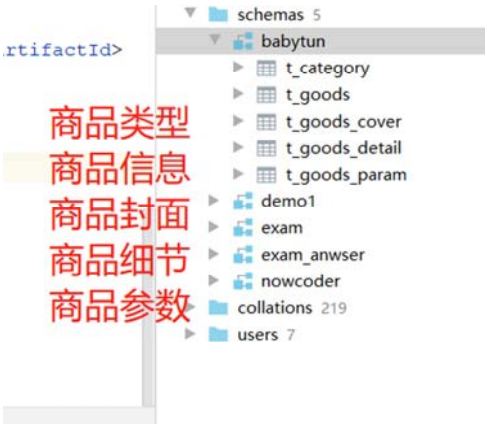
1、创建maven项目

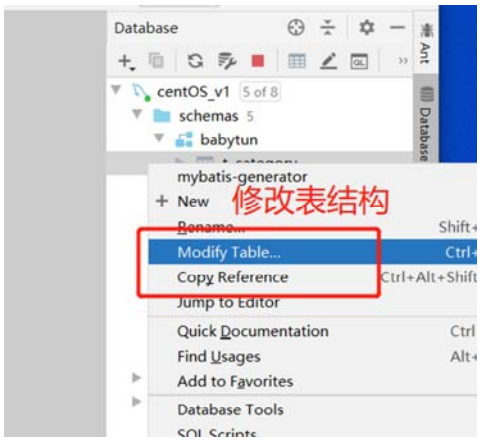
2、导入

1、安装使用idea自带的数据库链接工具



3、导入所需sql脚本





sqlSessionFactory

它是Mybatis的核心对象；用于初始化MyBatis，创建SqlSession对象；保证SqlSessionFactory在应用中全局唯一。其本质上就是加载配置文件，来完成对于MyBatis这个框架的初始化工作。

数据的增删改查有SqlSession对象来完成。

SqlSession是MyBatis操作数据库的核心对象；SqlSession使用JDBC方式与数据库交互。SqlSession对象提供了数据表的CRUD对应方法。

MyBatis环境配置

1、引入mybatis、junit单元测试框架、mysql链接的依赖

```
<dependencies>
<dependency>
<groupId>org.mybatis</groupId>
```

```
<artifactId>mybatis</artifactId>
<version>3.5.1</version>
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.47</version>
</dependency>
<!--java中的单元测试框架，可以测试之前写的mybatis-config.xml文件-->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>RELEASE</version>
<scope>test</scope>
</dependency>
</dependencies>
```

2、对pom.xml文件换源

```
<repositories>
<repository>
<id>aliyun</id>
<name>aliyun</name>
<url>https://maven.aliyun.com/repository/public</url>
</repository>
</repositories>
```

3、编写配置MyBatis-config.xml文件，官方文档里有模板

https://mybatis.org/mybatis-3/zh/configuration.html

```
<?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>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${driver}"/>
        <property name="url" value="${url}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml"/>
  </mappers>
</configuration>
```

4、

编写测试类测试链接

```
@Test
publicvoidtestSqlSessionFactory()throwsIOException{
//利用Reader加载classpath下的mybatis-config.xml核心配置文件
Readerreader=Resources.getResourceAsReader("mybatis-config.xml");
//初始化SqlSessionFactory对象,同时解析mybatis-config.xml文件
SqlSessionFactorysqlSessionFactory=newSqlSessionFactoryBuilder().buil
d(reader);
System.out.println("SessionFactory加载成功");
SqlSessionsqlSession=null;
try{
//创建SqlSession对象,SqlSession是JDBC的扩展类,用于与数据库交互
sqlSession=sqlSessionFactory.openSession();
```

```
//创建数据库连接(测试用)
Connectionconnection=sqlSession.getConnection();
System.out.println(connection);

}catch(Exceptione){
e.printStackTrace();
}finally{
if(sqlSession!=null){
//如果type="POOLED",代表使用连接池,close则是将连接回收到连接池中
//如果type="UNPOOLED",代表直连,close则会调用Connection.close()方法关闭连接
sqlSession.close();
}
}
}
```

初始化工具类

初始化工具类的目的是省去每次都需要创建session

使用工具类后，只需要这一句话就能够完成sqlsessionFatory的初始化及创建新的Session的工作

得到一个全新的sqlSession对象 能够能快的完成数据的增删改查操作。

1、编写工具类

```
/**
*MyBatisUtils工具类,创建全局唯一的SqlSessionFactory对象
*/
publicclassMyBatisUtils{
//保证MyBatis的对象全局唯一的特性//利用static(静态)属于类不属于对象,且全局唯一
privatestaticSqlSessionFactorysqlSessionFactory=null;
//利用静态块在初始化类时实例化sqlSessionFactory
static{
Readerreader=null;
try{
reader=Resources.getResourceAsReader("mybatis-config.xml");
sqlSessionFactory=newSqlSessionFactoryBuilder().build(reader);
}catch(IOExceptione){
e.printStackTrace();
//初始化错误时,通过抛出异常ExceptionInInitializerError通知调用者，类的初始化过程中产生的错误
thrownewExceptionInInitializerError(e);
}

}

/**
*openSession创建一个新的SqlSession对象
*@returnSqlSession对象
*/
//在后续的数据表中，利用SqlSession完成数据表的增删改查操作
publicstaticSqlSessionopenSession(){
returnsqlSessionFactory.openSession();
}

/**
*释放一个有效的SqlSession对象
*@paramsession准备释放SqlSession对象
*/
publicstaticvoidcloseSession(SqlSessionsession){
if(session!=null){
session.close();
}
}

}
```

2、对工具类进行测试

```
@Test
publicvoidtestMyBatisUtils()throwsException{
SqlSessionsqlSession=null;
```

```
//使用工具类后，只需要这一句话就能够完成sqlsessionFatory的初始化及创建新的
Session的工作
//得到一个全新的sqlSession对象，
try{
sqlSession=MyBatisUtils.openSession();
Connectionconnection=sqlSession.getConnection();
System.out.println(connection);
}catch(Exceptione){
throwe;
}finally{
MyBatisUtils.closeSession(sqlSession);
}


}
}
```

以上是对工具类链接的测试。

MyBatis实现数据查询操作

1、查询步骤

①创建实体类（Entity）

②创建Mapper XML （用来说明当前SQL语句是什么）

③在②中，增加编写<select>SQL标签

④在核心的Mybatis-config.xml文件中开启驼峰命名映射

⑤在核心的Mybatis-config.xml文件中新增<mapper>的声明

⑥SqlSession调用select方法来执行select语句

MyBatis数据写入

前置知识：数据库事务；数据库事务是保证数据操作完整性的基础操作

所有从客户端发来的增删改，都会先被发送到Mysql中的事务日志中，在事务日志中体现出新增数据，当客户端向mysql发起了commit的提交命令时，事务日志才会向数据表写入新增数据。当全部数据写入数据表中后，事务日志将会被清空。

假设数据一，二执行成功，数据三未执行成功，客户端会产生一个rollback回滚的命令，当前事务日志表中的所有数据都会被清除，同时也不会放入数据表中。只有当所有数据都执行成功，并且由客户端发起commit提交命令，才将所有的数据提交到数据表中。

MyBatis写操作包括三种　插入insert　更新update　删除delete 是不需要书写resultType或者resultMap。

insert数据插入

将Goods对象中的数据，带入到insert into的新增语句中，设置值的部分则使用#{属性名字}，来进行指代。

selectKey与useGeneratedKeys的区别

selectKey用于插入数据后，将最新的主键值进行返回；
useGeneratedKeys

两者的区别（显示与隐示范）：

selectKey标签需要明确编写获取最新主键的SQL语句

useGeneratedKeys属性会自动根据驱动生成对应SQL语句

selectKey适用于所有的关系型数据库

useGeneratedKeys只支持“自增主键”类型的数据库 不支持的DB2,oracle

总结：

selectKey标签是通用方案，适用于所有数据库，但编写麻烦

useGeneratedKeys属性只支持“自增主键”类型的数据库，使用简单

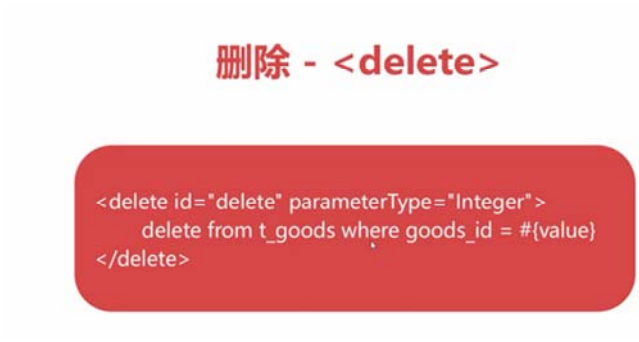
数据的更新与删除操作

更新

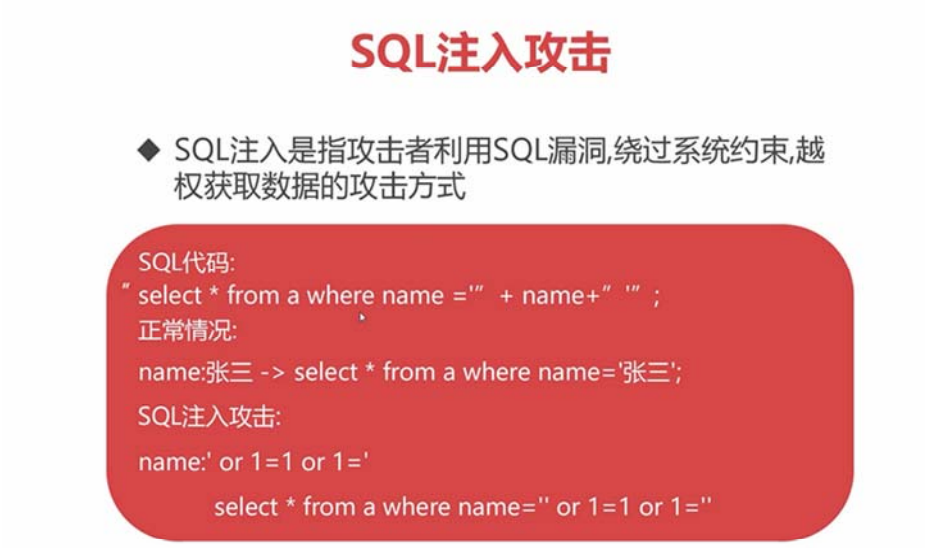
1、获取原始的数据（根据主键ID获取得到原始的数据）

2、对原始的数据进行修改，进行更新的操作

3、提交事务

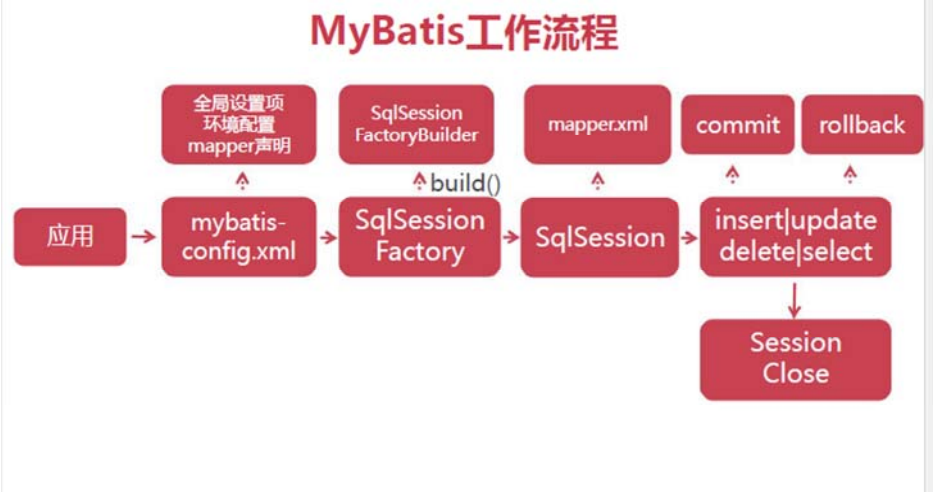


预防sql注入攻击



预编译传值会对特殊字符做转义处理，可以有效预防sql注入。

MyBatis的工作流程



1、完成mybatis的初始化工作；

2、构建SqlSessionFactory，需要保证全局唯一。

高级特性

MyBatis的日志管理

日志：日志文件是用于记录系统操作事件的记录文件或文件集合，日志保存历史数据,是诊断问题以及理解系统活动的重要依据

java中的日志

通过统一的日志门面，屏蔽了底层复杂的实现。底层的切换，可以由日志门面自动自动完成迁移工作。工作项目中，logback是目前的主流。MyBatis通过SLF4J来支持logback。

日志的打印及配置

MyBatis动态SQL

MyBatis的二级缓存

缓存：缓冲存储的意思，用于数据优化，提高程序执行效率的有效方式。

一级缓存默认开启,缓存范围SqlSession会话；二级缓存手动开启,属于范围Mapper Namespace；

二级缓存运行规则

1、二级开启后默认所有查询操作均使用缓存；

2、写操作commit提交时对该namespace缓存强制清空；

3、配置useCache=false可以不用缓存；

4、配置flushCache=true代表强制清空缓存；

MyBatis中的二级缓存，可以跨越两个不同的session得到相同hashcode的值。

一级缓存的生存周期仅局限于一个session会话中，当session创建开启时，产生一级缓存，当session会话关闭后，一级缓存也随之被清空。同时，在同一个session中，提交一个commit操作，也会清空一级缓存。

开启二级缓存

```
<!--     开启二级缓存，使用最近最久未使用算法-->
    <cache eviction="LRU" flushInterval="600000" size="512" readOnly="true"/>
```

eviction是缓存的清除策略,当缓存对象数量达到上限后,自动触发对应算法对缓存对象清除

1.LRU – 最近最久未使用:移除最长时间不被使用的对象。

O1 O2 O3 O4 .. O512

14 99 83 1 893

2.FIFO – 先进先出:按对象进入缓存的顺序来移除它们。

3.SOFT – 软引用:移除基于JVM中的垃圾收集器状态和软引用规则的对象。

4.WEAK – 弱引用:更积极的移除基于垃圾收集器状态和弱引用规则的对象。

5.LFU -最近最少被使用

flushInterval 代表间隔多长时间自动清空缓存，单位毫秒，600000毫秒 = 10分钟。

Size 缓存存储上限，用于保存对象或集合（1个集合算一个对象)的数量上限。

Readonly 设置为true，代表返回只读缓存，每次从缓存取出的是缓存对对象本身，这种执行效率较高。

设置为false，代表每次取出的是缓存对象的“副本”，每一次取出的对象都是不同的，这种安全性较高。

在其他子标签中

useCache="true" 是否使用缓存。全局查询时，不推荐将其放入缓存。一下子返回很多数据的查询，缓存命中率也是比较低的，也不推荐放入缓存中。

```
    <select id="selectAll" resultType="com.mybatis.entity.Goods" useCache="false">
        select * from t_goods order by goods_id desc limit 10
    </select>
```

```
<!--完成数据插入操作-->
    <insert id="insert" parameterType="com.mybatis.entity.Goods" flushCache="true">
        INSERT INTO t_goods(title,sub_title,original_cost,current_price,discount,is_free_delivery,cate
        VALUES(#{title},#{subTitle},#{originalCost},#{currentPrice},#{discount},#{isFreeDelivery},#{ca
<!--AFTER代表执行完上述insert语句后，开始执行下面的语句
last_insert_id()为mysql自带的，用于获取当前链接最后产生的id号-->
        <selectKey resultType="Integer" keyProperty="goodsId" order="AFTER">
            select last_insert_id()
        </selectKey>
    </insert>
```

在执行完sql语句后，立即强制执行清空缓存

```
    <select id="selectGoodsMap" resultType="java.util.LinkedHashMap" flushCache="true">
        select g.* , c.category_name from t_goods g , t_category c
        where g.category_id = c.category_id
    </select>
```

执行完sql后，缓存被强制清空，清空后，该sql的执行结果同样不在缓存中

OneToMany对象关联查询

实体关系分析

ManyToOne对象关联查询

多对一对象关联查询

分页插件PageHelper

官方文档为： https://pagehelper.github.io/docs/howtouse/

```
<!--
    plugins在配置文件中的位置必须符合要求，否则会报错，顺序如下：
    properties?, settings?,
    typeAliases?, typeHandlers?,
    objectFactory?,objectWrapperFactory?,
    plugins?,
    environments?, databaseIdProvider?, mappers?
-->
<plugins>
    <!-- com.github.pagehelper为PageHelper类所在包名 -->
    <plugin interceptor="com.github.pagehelper.PageInterceptor">
        <!-- 使用下面的方式配置参数，后面会有所有的参数介绍 -->
        <property name="param1" value="value1"/>
        </plugin>
</plugins>
```

1. `helperDialect`：分页插件会自动检测当前的数据库链接，自动选择合适的分页方式。 你可以配置`helperDialect`属性来指定分页插件使用哪种方言。配置时，可以使用下面的缩写值：
   `oracle`,`mysql`,`mariadb`,`sqlite`,`hsqldb`,`postgresql`,`db2`,`sqlserver`,`informix`,`h2`,`sqlserver2012`,`derby`

5. `reasonable`：分页合理化参数，默认值为`false`。当该参数设置为 `true` 时，`pageNum<=0` 时会查询第一页，`pageNum>pages`（超过总数时），会查询最后一页。默认`false`时，直接根据参数进行查询。

不同数据库的分页实现原理

不同数据库的分页实现原理

MySQL分页

```
select * from table limit 10,20;
```

Oracle

```
select t3.* from (
    select t2.*, rownum as row_num from (
        select * from table order by id asc
    ) t2 where rownum<=20
) t3
where t2.row_num>11
```

SQL Server 2000

```
select top 3 * from table
where
    id not in
    (select top 15 id from table)
```

SQL Server 2012+

```
select * from table order by id
    offset 4 rows fetch next 5 rows only
```

MyBatis的批处理