Universe Scientific Publishing
Computer System Network and Telecommunications,计算机系统网络和电信(1) 2019,3
ISSN: 2661-3719(Print) ISBN:2661-3727(Online)

浅析 mybatis 在 java 开发中的应用

蒋 超 鲍正德 李晨曦
四川大学锦城学院 计算机软件学院 四川成都 611731

DOI: 10.18686/jsjxt.v1i3.1276

【摘要】mybatis 是一款性能优良的 JavaEE 持久层框架,因此在大型的 JavaEE 开发中,开发者通常会选用它来作为开发项目的持久层框架;本文通过自动回复机器人的案例介绍了 mybatis 在项目持久层开发中的基本应用和工作原理。本案例使用 mybatis 框架完成了两个基本功能:页面消息的自动回复和后台页面的信息维护;通过本文的项目案例,读者可以了解到 mybatis 作为持久层框架会有很多益处:基本的 ORM 实现、SQL 语句的统一管理、动态 SQL 语句的实现。

【关键词】java 持久层 mybatis jdbc

1 引言

时至今日,java 语言仍是互联网领域内最热门的程序设计语言。但是在大型的 java 项目开发中,如果开发者直接通过传统的 JDBC 技术进行数据库的相关操作会产生大量的重复代码,而且步骤繁琐,这样会降低开发人员的工作效率。所以在 Java 的企业开发中,开发者经常会使用将 JDBC 进行了完整封装的持久层框架来开发持久层,而 mybatis 就是一款性能优良的持久层框架。[2]

2 mybatis 的基本介绍

2.1 持久层

在一般的大型软件开发中,开发人员通常会在开发项目中设立一个专门与数据库交互的层次来实现数据的持久化,这个层次一般被称为持久层。[2]

2.1 mybatis

MyBatis 作为一款开发者首选的持久层框架,传统的 jdbc 被封装于其中。对比于传统 jdbc,mybatis 有很多优势:对 SQL 语句的统一管理、动态 SQL 语句的拼接、ORM 映射等等。

3 mybatis 与 jdbc 的比较

3.1 jdbc 的介绍

由于 jdbc 繁琐的访问数据库流程,在实际的项目开发中,开发人员一般会选用对 jdbc 进行封装的持久层框架来与数据库交互。相比于传统 jdbc 的繁琐的操作数据库的步骤,Mybatis 框架支持定制化 SQL、存储过程、高级映射,避免手动和 JDBC 代码设置参数及获取结果集等诸多功能。[1]

3.2 使用 jdbc 与数据库交互的一个简单案例

这个案例是后面自动回复机器人项目中的一个功能板块(根据前端页面的多个关键字段来查询指定的信息并返回结果给前端页面)。这里使用了 jdbc 来完成这个查询功能以便介绍 jdbc 操作数据库的一般流程:

第一步:先加载 mysql 的驱动程序并获得本案例对应的数据库;第二步:sql 语句的准备(初始化 SQL 查询语句,再利用 java 代码将相关 SQL 条件语句与查询参数放置在 SQL 查询语句后面);[3]第三步:调用 SQL 语句获得返回结果并将返回的数据拼装成所需对象;第四步:关闭相关资源。在本案例中,相关实体类与数据库表的设计可以参考本文后面项目的图表设计。

```
//加载驱动与获取数据库连接
Class.forName("com.mysql.jdbc.Driver");
Connection conn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/JC
//sql语句的准备
Statement stmt =(Statement) conn.createStatement();
StringBuilder sql=new StringBuilder("SELECT * FROM message");
//SQL语句的执行
rs = stmt.executeQuery(sql.toString());
//结果集类型的转换
while(rs.next())
        {
            Message message=new Message();
            messagelist.add(message);
            message.setId(rs.getString("ID"));
            message.setCommand(rs.getString("COMMAND"));
            message.setDescription(rs.getString("DESCRIPTION"));
            message.setContent(rs.getString("CONTENT"));
        }
return messagelist;
```

图 1 jdbc 实现信息查询功能的代码段

3.3 jdbc 与 mybatis 的对比

在上述的简单案例中,传统的 jdbc 与数据库交互的流程有许多的缺陷:

第一点,在 jdbc 中,数据源的获取与释放过于频

ISSN: 2661-3719(Print) ISBN:2661-3727(Online)

繁,开发者也不能方便地配置数据源信息;在 mybatis 中,开发者能够以配置文件的方式在核心配置文件中方便地配置数据源信息。[4]

第二点,在 jdbc 中,SQL 语句与程序代码未能分离,不便于程序代码的维护;在 mybatis 框架中,开发者能够一并地将 SQL 语句配置在映射文件中,这样便可使得开发者可以专注于业务逻辑而不用耗费精力在 SQL 语句上;

第三点,在 jdbc 中,因为程序代码与 SQL 语句并未分离,开发者对 SQL 语句拼接的难度很大;

第四点,在上述 jdbc 实例中,开发者需要从结果集 ResultSet 中取出结果并转换为指定类型;在 Mybatis 中,开发者可以通过配置文件将数据库表对象与 javabean 进行映射,无需额外的类型转换操作;

Mybatis 的诸多优势在后面的具体项目代码中会一一体现,这里就不在赘述。

4 mybatis 框架运行的基本原理

4.1 mybatis 中的核心对象与相应的配置文件:

核心配置文件:在该文件中,开发者主要配置了项目的数据源、映射文件的路径。[4]

Mapper. xml:在映射文件中,主要放置了 ORM 映射的相关配置以及所有的 SQL 语句;

SqlSessionFactory:读者从这个对象的名字就可以了解到,它是 sqlsession 实例的生产者。当每次项目启动到项目结束的运行期间,该对象提供了整个项目需要使用的全部 sqlsession。在项目启动时,该对象由数据源配置类通过读取 mybatis 的核心配置文件所创建,在该对象被创建完成后,开发者就可以通过它获取源源不断的 sqlsession,进而通过 sqlsession 来与数据库交互。一般来讲,开发的项目若涉及到多个数据库,便需要获取多个 sqlsessionfactory。

SqlSession:项目的数据库操作都通过该对象执行。

Executor:Executor 是 sqlsession 里面的一个子对象,在每次的 SQL 会话执行过程中,都由它来完成具体的数据库相应操作。

MappedStatement: MappedStatemet 是 executor 对象里面的一个参数。该参数用来对映射文件中 SQL 语句中的所有信息进行组装,为 Executor 对象具体的数据库操作做准备。

4.2 Mybatis 的执行流程图:

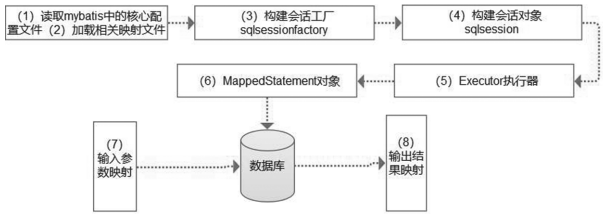

图 2 Mybatis 的执行原理图

5 mybatis 在自动回复机器人项目开发中的应用

5.1 整个项目的结构

(1)项目结构图:

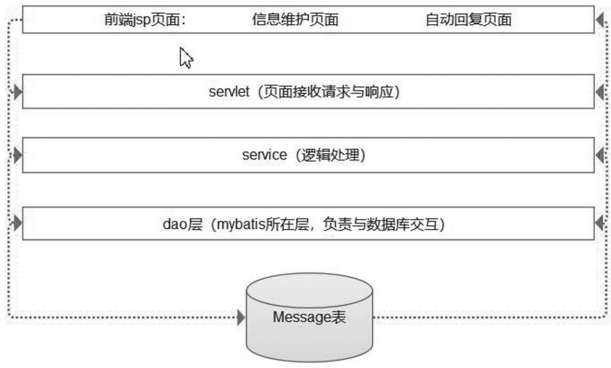

图 3 项目的结构图

(2) 项目的功能介绍

由于自动回复机器人项目案例的侧重点是介绍 mybatis 框架在项目持久层开发中的具体应用,所以基于 dao 层的接口规划,笔者大体上可以将自动回复机器人案例划分为两个功能板块:

第一功能块:相应 servlet 接收到前端信息管理页面的请求并向下传到 dao 的接口调用类,然后 dao 层对数据库相应表做增删改查操作,并将操作后得到的数据集逐级送回到前端页面;[5]

第二功能块:相应 servlet 收到前面回复页面请求的关键字并向下传给 dao 层接口调用类,之后 dao 层对数据库的 Message 表做查询操作并将查询结果返回给前端回复页面。[5]

5.2 项目的具体实现

(1)相关实体类设计、数据库表结构设计、配置文件内容:

Message 类设计:

ISSN: 2661-3719(Print) ISBN:2661-3727(Online)

Message Bean		
id	int	编号
title	string	标题
description	string	描述
content	string	内容

图 4 项目实体类设计

数据库表结构设计:

Message table		
ID	int	编号 (主键)
title	varchar	标题
description	varchar	描述
content	varchar	内容

图 5 项目数据库表的设计

Configrutaion. xml 文件配置:下图的内容是本项目案例中的 mybatis 配置文件的内容:包含了项目的数据源和本项目映射文件路径的编写。

```
<configuration>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC">
        <property name="" value=""/>
      </transactionManager>
      <dataSource type="UNPOOLED">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/JC"/>
        <property name="username" value="root"/>
         <property name="password" value="123456"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="config/Message.xml"/>
  </mappers>
</configuration>
```

图 6 mybatis 核心配置文件内容

Mapper. xml 文件中重要信息的配置:

bean 对象与 message 表的映射:

```
<mapper namespace="dao.IMessage">
  <resultMap type="bean.Message" id="MessageResult">
    <id column="ID" jdbcType="INTEGER" property="id"/>
    <result column="COMMAND" jdbcType="VARCHAR" property="command"/>
    <result column="DESCRIPTION" jdbcType="VARCHAR" property="description"/>
    <result column="CONTENT" jdbcType="VARCHAR" property="content"/>
  </resultMap>
```

图 7 实体类与数据库表的映射配置

信息维护页面信息查询对应的 SQL 语句:使用 OGNL 表达式做了 SQL 语句的拼接:

```
<select id="querymessagelist"  parameterType="bean.Message" resultMap="MessageResult">
  SELECT * FROM message
  <if test="command!=null">and COMMAND=#{command}</if>
  <if test="description!=null">and DESCRIPTION like '%'#{description}'%'</if>
</select>
```

图 8 信息查询的 SQL 语句

信息维护页面信息的批量删除 SQL 语句:使用 OGNL 表达式做了相应 id 的遍历:

```
<delete id="deletebatch" parameterType="java.util.List">
    delete from n=message where ID IN(
    <foreach collection="list" item="item" separator=",">
        #{item}
    </foreach>)
</delete>
```

图 9 信息删除的 SQL 语句

……,后面类似的 SQL 语句不再一一赘述。

(2)数据源获取类的设计、持久层的接口设计:

数据源获取类:读取配置文件构建 sqlsession-factory,得到 sqlsession 并返回给调用对象:

```
public class DBAccess {
    public SqlSession getsqlsession() throws IOException
    {
        //读取核心配置文件
        Reader reader=Resources.getResourceAsReader("config/Configuration.xml");
        //构建sqlsessionfactory
        SqlSessionFactory sessionfactory=new SqlSessionFactoryBuilder().build(reader)
        //构建sqlsession
        SqlSession sqlsession=sessionfactory.openSession();
        return sqlsession;
    }
```

图 10 数据源获取类的设计

dao 层接口:基于前面项目的两大功能板块与配置文件中编写的 SQL 语句,在 dao 层的接口设计中,做了如下几个函数的设计:

```
public interface IMessage {

    //信息维护页面的查询方法
    public List<Message> querymessagelist(Message message);
    //自动回复页面的查询方法
    public Message query_one(String command);
    //维护页面的信息插入
    public void insert(Message message);
    //维护页面的信息修改
    public void modify(Message message);
    //维护页面的信息删除
    public void deletebatch(List<Integer> ids);
}
```

图 11 dao 层接口的设计

(3)相应接口功能的实现

相应接口功能的实现(这里就以信息页面的批量删除为实例):首先利用数据源类获得 sqlsession;dao 层的“interface”再通过会话对象的“getmapper”方法获取与之对应的“mapper. class”;最后通过持久层接口执行相应 SQL 语句并进行事务提交:

```
public void deletebatch(List<Integer> ids) throws IOException
{
    //获取数据源类
    DBAccess db=new DBAccess();
    //通过数据源类取得sqlsession
    SqlSession sqlsession = db.getsqlsession();
    //接口与映射标签的匹配
    IMessage imessage=sqlsession.getMapper(IMessage.class);
    //SQL语句的执行
    imessage.deletebatch(ids);
    //提交事务
    sqlsession.commit();
    //资源关闭
    sqlsession.close();
}
```

图 12 信息删除功能的实现

5.3 项目总结

本项目案例的前端部分涉及到两个 jsp 页面;后台有三个纵向部分。在控制层,采用了 servlet 来接收前端页面的请求并返回相应结果集;逻辑层在本

ISSN: 2661-3719(Print) ISBN:2661-3727(Online)

案例中存在的意义不大;在项目的 dao 层,本项目使用了 mybatis 来完成与数据库的交互。[5]在持久层的设计中,采用了面向接口编程的方式来实现接口与相关映射标签的匹配,进而利用调用接口的形式来执行 sql 语句。

上述机器人自动回复项目案例介绍了 mybatis 框架运行的基本原理,并使用了面向接口编程、动态 SQL 语句等重要技术来完成整个持久层与数据库交互的功能板块实现。

6 mybatis 的优缺点与应用场景

6.1 Mybatis 的利与弊:

在 Java 开发中,相比于其他 dao 层框架,Mybatis 有许多优势:

简单易学,对于许多 Java 开发新手而言,mybatis 用作 dao 层框架是一个不错的决定;

灵活:在 mybatis 框架中,一切 SQL 语句都被放置于 XML 文件中,SQL 语句与逻辑代码的分开提高了项目的开发效率;在 jdbc 中,SQL 语句的拼接相当繁琐,而在 mybatis 中,SQL 语句的拼接十分容易实现;

消除了 SQL 与程序代码的耦合:mybatis 处于持久层的位置,专注于数据库的交互,不会参与业务逻辑,这样使得项目的体系更加明确,有利于开发效率的提高;SQL 语句放置于配置文件中这一举措使得开发者可以专注于 java 面向对象的开发,而不用耗费过多精力在 SQL 语句上。

同样实现相同的访问数据库动作,对比于传统的 jdbc,mybatis 能够缩减一半以上代码量。

mybatis 的不足之处:

在 mybatis 中,SQL 语句统一在 xml 文件中配置,配置文件中巨量的 SQL 语句编写有些繁琐;

mybatis 框架侧重于 SQL 语句,这样会使得数据库的移植性变得很差。

6.2 Mybatis 的应用场景:

对比于其他的持久层框架,mybatis 更侧重于 SQL,它使得用户更多地通过 SQL 语句来实现 ORM 功能。这种模式虽然灵活,但在开发一个相对成熟的项目中会降低开发效率,所以企业如果使用 Mybatis,需要根据开发项目的实际情况对 Mybatis 进行再次的封装,让数据库访问愈加简洁和适用,从而提高开发效率。[2]

【参考文献】

[1]何军,陈倩怡. Vue+Springboot+Mybatis 开发消费管理系统[J]. 电脑编程技巧与维护,2019(02):87-88+102.

[2]荣艳冬. 关于 Mybatis 持久层框架的应用研究[J]. 信息安全与技术,2015,6(12):86-88.

[3]李伟超. 利用 JavaJDBC 进行数据库访问[J]. 民营科技,2014 年 04 期.

[4]管才路. 基于 Java 的 Mybaits 生成持久层配置文件[J]. 电子科技与软件工程,2018,(22),139.

[5]贺雪梅. web 应用开发中的 SSM 框架设计[J]. 电子世界,2019,(01),206.