1、用Map集合给SQL映射文件中的SQL语句传值

```
--parameterType
    1、--该属性是专门给SQL语句传值的

    2、--该属性可以采用：
        JavaBean
        简单类型
        Map集合
        ...

    3、--前面例子使用过JavaBean和简单类型给SQL语句传值，下面测试用Map集合给SQL语句传值

    4、--什么情况需要使用Map集合传值而不能用JavaBean传值？
    当JavaBean不够用的时候我们才用Map集合给sql语句传值
        一般情况下，一个表对应一个JavaBean，两张表就需要在实体类包中创建两个JavaBean。
        当sql查询语句中是多表联合查询，条件中要传的值有的是A表中的，有的是B表中的，这样跨表的情况下，就没有合
        适的JavaBean来传值。
        如果我们为了这条SQL语句新建一个存储所需要传值的JavaBean，然后创建对象传值，成本太高了。
        这时我们只需要创建一个Map集合，往集合添加数据时将其key命名为需要传值的表中字段一致即可。
            select
                e.*
            from
                emp e
            join
                dept d
            on
                e.deptno=d.deptno
            where
                d.dname="RESEARCH" and e.job="MANAGER";
        如上面的语句中条件中的两个参数是两个表中的，用Map集合进行传值

        5、--使用log4j组件将执行的SQL语句打印到控制台
            log4j：
                logger for java，一个第三方组件，为Java语言准备的日志工具
            很多框架都集成了该组件，如spring、springMVC、mybatis、hibernate等，该组件专门用来记录日志
        --开发中不需要精通log4j，只需要把相关配置和相关jar导入即可
        第一步：
            在类的根路径下新建一个log4j.properties的配置文件，具体里面写什么，直接百度，不同框架的执行的sql
            语句输出到控制台的配置不一样。这里我们是mybatis使用log4j，所以搜索 "mybatis log4j sql",如果是
            spring，就是"spring log4j sql"。
            注意：高版本即log4j2不再使用properties文件，用的是xml文件配置，mybatis用的是log4j1.x.x

            log4j.rootLogger=DEBUG, Console
            log4j.appender.Console=org.apache.log4j.ConsoleAppender
            log4j.appender.Console.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n
log4j.logger.org.apache=INFO
```

第二步：

引入log4j的jar包：log4j-1.2.17.jar

然后直接运行程序即可。

创建模块项目d-mybatis-parameterType-map，添加jdbc.properties、config.xml、SqlMapper.xml，测试时类MyBatisParameterMapTest01.java以及sql语句打印的日志配置文件log4j.properties，不用新建实体类JavaBean。具体代码如下

jdbc.properties

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mysqlstudy
jdbc.username=root
jdbc.password=rong195302
```

config.xml

```
<?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>

    <!--引入独立的配置文件-->
    <!--mybatis中的resource默认从项目的类路径查找-->
    <properties resource="jdbc.properties"/>

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

SqlMapper.xml

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

```
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="test">

    <!--使用Map集合给sql语句传值，占位符中的字段名要和Map集合中的key属性名一致-->
    <!--所以我们给Map集合中添加数据时key属性名一般与表中的字段名一致，javabean传值也一样-->
    <!--
    下面四种种写法都可以
    <insert id="save" parameterType="java.util.Map">
    <insert id="save" parameterType="java.util.HashMap">
    <insert id="save" parameterType="map">
    -->
    <insert id="save" parameterType="Map">
        insert into t_user
            (loginName,loginPwd,realName)
        values
            (#{loginName},#{loginPwd},#{realName})
    </insert>


</mapper>
```

MyBatisParameterMapTest01.java

```
package com.mybatis;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class MyBatisParameterMapTest01 {

    public static void main(String[] args) {

        SqlSession sqlSession = null;
        try {

            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(Resources.getResourceAsStream("mybatis-
config.xml"));

            sqlSession = sqlSessionFactory.openSession();

            //创建Map集合，往集合中添加数据
            Map<String,String> map = new HashMap<>();
            map.put("loginName","6666666@mail.com");
            map.put("loginPwd","8888888");
            map.put("realName","中国大使馆");

            //然后用该集合往sql语句中传值,执行SQL语句
            int count = sqlSession.insert("save",map);
```

```
            System.out.println("往数据库中添加了 "+count+" 条数据");
            sqlSession.commit();
        } catch (IOException e) {
            if (sqlSession != null) {
                sqlSession.rollback();
            }
            e.printStackTrace();
        }finally {
            sqlSession.close();
        }

    }

}
```

log4j.properties

```
log4j.rootLogger=DEBUG, Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n
log4j.logger.org.apache=INFO
```

2、用Map集合存储查询结果集

```
--resultType
    1、--该属性是专门用来指定查询结果集的封装数据类型

    2、--该属性可以采用：
        JavaBean
        简单类型
        Map集合
        ...

    3、--一定要注意：无论是什么类型，什么情况，resultType属性都不能省略，并且只有select语句才有该属性

    4、--前面例子使用过JavaBean封装查询结果集，下面测试用简单数据类型和Map集合封装查询结果集数据
    resultType="Map"的作用：
    就是告诉mybatis在执行完sql查询语句后，将查询结果集中的每一条记录，以该条记录中的列名为key，该条记录中的值
    为value，添加到一个Map集合中，就是说底层的方法执行完该查询sql语句后返回的是查询结果集记录条数个Map集合，
    然后下面java语句：
    List<Map> list = sqlSession.selectList("getNameAndDptByJob","MANAGER");
    就会把这些个Map集合存到List集合中
    resultType="JavaBean类型"的作用：
    告诉mybatis在执行完sql查询语句后，将查询结果集中的每一条记录，将该条记录与JavaBean中属性字段名一致的列名
    的值赋值给JavaBean对象中的对应字段，返回的是一个个JavaBean对象，存储到List集合中
```

创建项目e-mybatis-resultType-map，各文件代码如下：

mybatis-config.xml

```
<?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>

    <!--引入独立的配置文件-->
    <!--mybatis中的resource默认从项目的类路径查找-->
    <properties resource="jdbc.properties"/>

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

SqlMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="test">
    <!--将查询的数据用简单类型封装保存-->
    <select id="getAllName" resultType="String">
        select
            ename
        from
            emp
    </select>

    <select id="getByEname" resultType="Map">
        <!--封装查询数据到Map集合中是以查询结果的列名为key，而列名是可以起别名的-->
        select
            e.ename,e.sal,d.dname
        from
            emp e
        join
           dept d
        on
            e.deptno = d.deptno
        where
            e.ename = #{ename}
```

```
    </select>

    <select id="getNameAndDptByJob" resultType="Map">
        select
            e.ename,d.dname
        from
            emp e
        join
           dept d
        on
            e.deptno = d.deptno
        where
            e.job = #{job}
    </select>
</mapper>
```

jdbc.properties

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mysqlstudy
jdbc.username=root
jdbc.password=rong195302
```

log4j.properties

```
log4j.rootLogger=DEBUG, Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n
log4j.logger.org.apache=INFO
```

MyBatisResultTypeTest01.java

```
package com.mybatis.test;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.util.List;
import java.util.Map;

public class MyBatisResultTypeTest01 {

    public static void main(String[] args) {

        SqlSession sqlSession = null;

        try {

            //获取核心配置文件信息的SqlSessionFactory对象
```

```
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(Resources.getResourceAsStream("mybatis-
config.xml"));

            //开启事务
            sqlSession = sqlSessionFactory.openSession();

            //do work()
            //查询emp表中的所有员工姓名，用String类型封装数据
//              List<String> empList1 = sqlSession.selectList("getAllName");
//              for (String ename:
//                   empList1) {
//                  System.out.println(ename);
//              }

            //查询员工"SMITH"的信息，显示名字、薪资和部门名称
            //这里使用Map集合封装查询结果集和String类型给sql语句传值
            //注意是selectOne()方法
            Map<String,String> smithMap =
sqlSession.selectOne("getByEname","SMITH");
            System.out.println(smithMap);
            System.out.println("==========================================");

            //查询出所有工作岗位为MANAGER的员工，要求显示员工姓名和部门名称
            //这里使用Map集合封装查询结果集和String类型给sql语句传值
            //因为没有合适的javabean封装查询结果集，所以泛型中填Map
                List<Map> list =
sqlSession.selectList("getNameAndDptByJob","MANAGER");
            System.out.println(list.toString());

            //提交事务
            sqlSession.commit();
        } catch (IOException e) {
            //出错，回滚事务
            if (sqlSession != null) {
                sqlSession.rollback();
            }
            e.printStackTrace();
        }finally {
            //关闭资源
            if (sqlSession != null) {
                sqlSession.close();
            }
        }

    }

}
```