Mybatis_day2

1. 添加修改细节

> 注意: Mybatis执行插入,修改时不允许出现空值,因此在插入修改时,想要使某个列的值可以为空,必须加入jdbcType属性

1.1. 执行修改时

```
<update id="update" parameterType="com.baizhi.entity.User" >
    update t_user set name = #{name},age=#{age,jdbcType=INTEGER} where id = #{id}
</update>
```

如果该值为空请设置jdbcType

1.2. 执行添加时

```
<insert id="save" parameterType="com.baizhi.entity.User">
    insert into t_user values(#{id},#{name,jdbcType=VARCHAR},#{age,jdbcType=INTEGER})
</insert>
```

为空时,指定类型

---

2. 全部的jdbcType属性

为了未来的参考，MyBatis 通过包含的 jdbcType 枚举型，支持下面的 JDBC 类型。

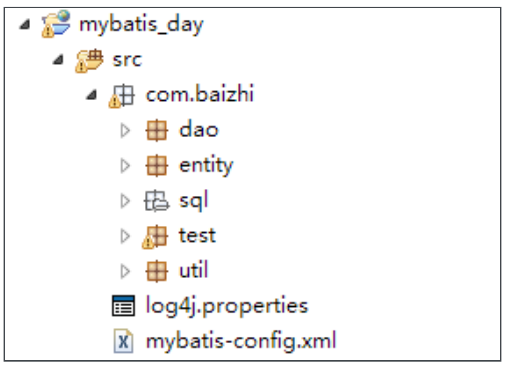
BIT	FLOAT	CHAR	TIMESTAMP	OTHER	UNDEFINED
TINYINT	REAL	VARCHAR	BINARY	BLOB	NVARCHAR
SMALLINT	DOUBLE	LONGVARCHAR	VARBINARY	CLOB	NCHAR
INTEGER	NUMERIC	DATE	LONGVARBINARY	BOOLEAN	NCLOB
BIGINT	DECIMAL	TIME	NULL	CURSOR	

> 注意:以上的jdbcType是mybatis中支持的所有jdbcType,但是根据使用不同的数据库不同的数据类型选择不同的jdbcType是有必要的

---

3. 显示执行sql语句

3.1. 引入log4j.properties



注意:必须放置在项目的src目录中

3.2. 配置log4j.properties

```
log4j.rootLogger=DEBUG, Console //根日志
log4j.appender.Console=org.apache.log4j.ConsoleAppender //控制台展示
log4j.appender.Console.layout=org.apache.log4j.PatternLayout //打印方式
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n //日志格式
log4j.logger.com.baizhi.dao=DEBUG
```

注意:

1.log4j中存在两种日志一种是项目中的根日志,一种是项目中子日志

2.一般在项目中开启子日志即可

3. 日志级别: ERROR、WARN、INFO、DEBUG、TRACE

---

4. Mybatis中的高级查询

4.1. 分页查询

4.1.1. 分页语句

1.排序

2.Rownum起别名,指定范围

3.指定开始记录

4.1.2. 分页语句的实现

注意:分页语句中的计算表达式的写法:

```
select e1.id,e1.name,e1.password from (select e.id,e.name,e.password,rownum r
from (select id,name,password from t_user order by id) e
    where rownum &lt;= #{pageNow}*#{pageSize}) e1 where r > (#{pageNow}-1)*#{pageSize}
```

4.2. 模糊查询

4.2.1. 实现

```
select * from t_user where name like '%'||#{name}||'%'
```

注意:模糊查询必须使用"||"去拼接两个'%'

5. Xml中的特殊字符

在 XML 中，有 5 个预定义的实体引用：

&lt;	<	小于
&gt;	>	大于
&amp;	&	和号
&apos;	'	单引号
&quot;	"	引号

注释：在 XML 中，只有字符 "<" 和 "&" 确实是非法的。大于号是合法的，但是用实体引用来代替它是一个好习惯。

> 注意:在xml中书写大于 小于等以上符号,必须进行转义

---

6. Mybatis中的别名

> 注意:在Mybatis中我们可以给实体起别名来处理,简化开发步骤

6.1. 声明别名

```
<configuration>

    <typeAliases>
        <typeAlias type="zpark.entity.User" alias="user"/>
    </typeAliases>

    <!-- 创建一个环境 -->
    <environments default="dev">
        <environment id="dev">
```

6.2. 使用别名

```
<!-- 管理DAO接口的方法 -->
<insert id="save" parameterType="user">
    insert into t_user values(#{id},#{name},#{bir})
</insert>
```

---

7. MyBatis中动态Sql

7.1. sql复用标签

```
<!-- 将公共的sql片段提取出来 -->
<sql id="userField">
        id,name,bir
</sql>
<select id="queryAll" resultType="zpark.entity.User" >
    select <include refid="userField"/> from t_user
</select>
```

> 注意:sql复用的标签,用来解决sql冗余的问题

7.2. if标签 动态拼接sql

```
<select id="queryUser" resultType="zpark.entity.User">
    select <include refid="userField"/> from t_user
    where
     <if test="name!=null and  name!=''">
        name = #{name}
     </if>
      and bir = #{bir}
</select>
```

注意: if标签判断条件满足执行if中内容 不满足不执行

7.3. where标签

```
<select id="queryUser" resultType="zpark.entity.User">
    select <include refid="userField"/> from t_user
    where
     <if test="name!=null and  name!=''">
        name = #{name}
     </if>
      and bir = #{bir}
</select>
```

注意:当在sql语句中使用if标签时往往会出现如下两种情况:

1.select * from where 条件一 and 条件二,当条件一不满足时sql语句就不是一条正确的语句

2. select * from where 条件一 and 条件二当所有条件不满足时sql语句也不是一条正确的语句

where标签作用:

可以自动去除条件中多余的 and|or 当所有条件不满足时 Where关键字也不会出现在语句中

7.4. choose when otherwise标签

```
<select id="queryUser" resultType="zpark.entity.User">
    select <include refid="userField"/> from t_user
    <where>
        <choose>
            <when test="name!=null and name!=''">
                name = #{name}
            </when>
            <otherwise >
                and bir = #{bir}
            </otherwise>
        </choose>
    </where>
</select>
```

注意: 当满足条件就执行when中语句,不满足条件就执行otherwise中语句

7.5. set标签

```
<update id="update" parameterType="zpark.entity.User">
    update t_user
        <set>
            <if test="name!=null and name!=''">
                name=#{name},
            </if>
            <if test="bir!=null and bir!=''">
                bir=#{bir}
            </if>
        </set>
        where id=#{id}
</update>
```

注意:用来在更新时去掉多余的逗号

7.6. trim标签

```
<update id="update" parameterType="zpark.entity.User">
    update t_user
        <trim prefix="set" suffixOverrides=",">
            <if test="name!=null and name!=''">
                name=#{name},
            </if>
            <if test="bir!=null and bir!=''">
                bir=#{bir},
            </if>
        </trim>
    where id=#{id}
</update>
```

注意:用来自定义sql的拼接方式,很灵活

7.7. foreach遍历标签

```
public void  insertAll(List<User> users);
```

集合类型

每遍历一次加入分号

```
<insert id="insertAll" parameterType="list"  >
    BEGIN
    <foreach collection="list" item="user" separator=";" close=";" >
        insert into t_user values(
        #{user.id},
        #{user.name},
        #{user.age})
    </foreach>
    END;
</insert>
```

遍历别名

最好一次加入分号

注意:用来在mapper文件中,遍历集合信息
