实验三：动态 SQL

一 、实验目的：通过 mybatis 提供的各种标签方法实现动态拼接 sql

二 、预习要求：预习 if、choose、 when、where 等标签的用法

三、实验内容：

(一)根据性别和名字查询用户

(二)使用 if 标签改造 UserMapper.xml

（三）使用 where 标签进行改造 UserMapper.xml

（四）使用 Sql 片段改造 UserMapper.xml

(五)利用 foreach 标签实现根据多个 id 查询用户信息

四、实验方法和步骤：

(一)根据性别和名字查询用户

查询 sql：SELECT id, username, birthday, sex, address FROM `user` WHERE sex = 1 AND username LIKE '%张%'

①Mapper.xml 文件

UserMapper.xml 配置 sql，如下：

```
<!-- 根据条件查询用户 -->
<select id="queryUserByWhere" parameterType="com.haust.pojo.User"
resultType="com.haust.pojo.User">
    SELECT id, username, birthday, sex, address FROM `user`
    WHERE sex = #{sex} AND username LIKE
    '%${username}%'
</select>
```

②Mapper 接口

编写 Mapper 接口，如下：

```
    /**
     * 根据条件查询用户
     *
     * @param user
     * @return
```

```
     */
    List<User> queryUserByWhere(User user);
```

③测试方法

在 UserMapperTest 添加测试方法，如下：

```
@Test
public void testQueryUserByWhere() {
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    SqlSession sqlSession = this.sqlSessionFactory.openSession();
    // 创建 Mapper 接口的动态代理对象，整合之后，交给 spring 管理
    UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
    // 使用 userMapper 执行根据条件查询用户
    User user = new User();
    user.setSex("1");
    user.setUsername("张");
    List<User> list = userMapper.queryUserByWhere(user);
    for (User u : list) {
        System.out.println(u);
    }
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    sqlSession.close();
}
```

④实验效果

测试效果如下图：

```
DEBUG [main] - ==>  Preparing: SELECT id, username, birthday, sex, address FROM `user`
DEBUG [main] - ==> Parameters: 1(String)
DEBUG [main] - <==      Total: 3
User [id=10, username=张三, sex=1, birthday=Thu Jul 10 00:00:00 CST 2014, address=北京市]
User [id=16, username=张小明, sex=1, birthday=null, address=河南郑州]
User [id=24, username=张三丰, sex=1, birthday=null, address=河南郑州]
```

如果注释掉 `user.setSex("1")`，测试结果如下图：

```
DEBUG [main] - ==>  Preparing: SELECT id, username, birthday, sex, address FROM `user
DEBUG [main] - ==> Parameters: null
DEBUG [main] - <==      Total: 0
```

测试结果二很显然不合理。

按照之前所学的，要解决这个问题，需要编写多个 sql，查询条件越多，需要编写的 sql 就更多了，显然这样是不靠谱的。

解决方案，使用动态 sql 的 if 标签

(二)使用 if 标签改造 UserMapper.xml

①改造 UserMapper.xml，如下：

```
<!-- 根据条件查询用户 -->
<select id="queryUserByWhere" parameterType="user" resultType="user">
    SELECT id, username, birthday, sex, address FROM `user`
    WHERE 1=1
    <if test="sex != null and sex != ''">
        AND sex = #{sex}
    </if>
    <if test="username != null and username != ''">
        AND username LIKE
        '%${username}%'
    </if>
</select>
```

注意字符串类型的数据需要要做不等于空字符串校验。

②实验效果

```
DEBUG [main] - ==>  Preparing: SELECT id, username, birthday, sex, address FROM `user` WHERE 1=1 AND us
DEBUG [main] - ==> Parameters:
DEBUG [main] - <==      Total: 3
User [id=10, username=张三, sex=1, birthday=Thu Jul 10 00:00:00 CST 2014, address=北京市]
User [id=16, username=张小明, sex=1, birthday=null, address=河南郑州]
User [id=24, username=张三丰, sex=1, birthday=null, address=河南郑州]
```

如上图所示，测试 OK

（三）使用 where 标签进行改造 UserMapper.xml

上面的 sql 还有 where 1=1 这样的语句，很麻烦

可以使用 where 标签进行改造

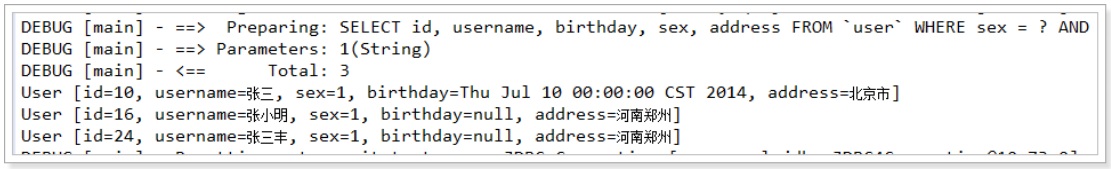
①改造 UserMapper.xml，如下

```
<!-- 根据条件查询用户 -->
```

```
<select id="queryUserByWhere" parameterType="user" resultType="user">
    SELECT id, username, birthday, sex, address FROM `user`
<!-- where 标签可以自动添加 where，同时处理 sql 语句中第一个 and 关键字 -->
    <where>
        <if test="sex != null">
            AND sex = #{sex}
        </if>
        <if test="username != null and username != ''">
            AND username LIKE
            '%${username}%'
        </if>
    </where>
</select>
```

②实验效果

测试效果如下图：



（四）使用 Sql 片段改造 UserMapper.xml

Sql 中可将重复的 sql 提取出来，使用时用 include 引用即可，最终达到 sql 重用的目的。

①把上面例子中的 id, username, birthday, sex, address 提取出来，作为 sql 片段，如下：

```
<!-- 根据条件查询用户 -->
<select id="queryUserByWhere" parameterType="user" resultType="user">
    <!-- SELECT id, username, birthday, sex, address FROM `user` -->
    <!-- 使用 include 标签加载 sql 片段；refid 是 sql 片段 id -->
    SELECT <include refid="userFields" /> FROM `user`
    <!-- where 标签可以自动添加 where 关键字，同时处理 sql 语句中第一个 and
关键字 -->
    <where>
        <if test="sex != null">
```

```
            AND sex = #{sex}
        </if>
        <if test="username != null and username != ''">
            AND username LIKE
            '%${username}%'
        </if>
    </where>
</select>
<!-- 声明 sql 片段 -->
<sql id="userFields">
    id, username, birthday, sex, address
</sql>
```

如果要使用别的 Mapper.xml 配置的 sql 片段，可以在 refid 前面加上对应的 Mapper.xml 的 namespace。

(六) 利用 foreach 标签实现根据多个 id 查询用户信息

向 sql 传递数组或 List，mybatis 使用 foreach 解析，如下：

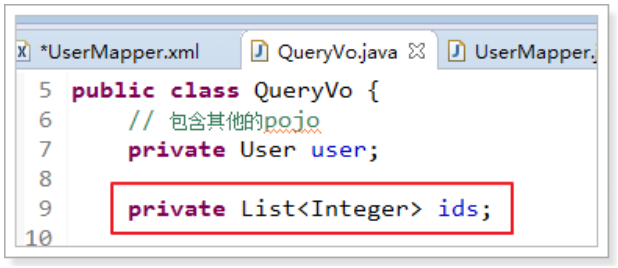
根据多个 id 查询用户信息

查询 sql：

SELECT * FROM user WHERE id IN (1,10,24)

①改造 QueryVo

如下图在 pojo 中定义 list 属性 ids 存储多个用户 id，并添加 getter/setter 方法



②Mapper.xml 文件

UserMapper.xml 添加 sql，如下：

```
<!-- 根据 ids 查询用户 -->
```

```
<select id="queryUserByIds" parameterType="queryVo" resultType="user">
    SELECT * FROM `user`
    <where>
        <!-- foreach 标签，进行遍历 -->
        <!-- collection：遍历的集合，这里是 QueryVo 的 ids 属性 -->
        <!-- item：遍历的项目，可以随便写,，但是和后面的#{}里面要一致 -->
        <!-- open：在前面添加的 sql 片段 -->
        <!-- close：在结尾处添加的 sql 片段 -->
        <!-- separator：指定遍历的元素之间使用的分隔符 -->
        <foreach collection="ids" item="item" open="id IN (" close=")"
            separator=",">
            #{item}
        </foreach>
    </where>
</select>
```

③测试方法如下图：

```
@Test
public void testQueryUserByIds() {
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    SqlSession sqlSession = this.sqlSessionFactory.openSession();
    // 创建 Mapper 接口的动态代理对象，整合之后，交给 spring 管理
    UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
    // 使用 userMapper 执行根据条件查询用户
    QueryVo queryVo = new QueryVo();
    List<Integer> ids = new ArrayList<>();
    ids.add(1);
    ids.add(10);
    ids.add(24);
    queryVo.setIds(ids);
    List<User> list = userMapper.queryUserByIds(queryVo);
```

```
    for (User u : list) {
        System.out.println(u);
    }
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    sqlSession.close();
}
```

④实验效果

测试效果如下图：

```
DEBUG [main] - ==>  Preparing: SELECT * FROM `user` WHERE id IN ( ? , ? , ? )
DEBUG [main] - ==> Parameters: 1(Integer), 10(Integer), 24(Integer)
DEBUG [main] - <==      Total: 3
User [id=1, username=李四, sex=2, birthday=null, address=null]
User [id=10, username=张三, sex=1, birthday=Thu Jul 10 00:00:00 CST 2014, address=北京市]
User [id=24, username=张三丰, sex=1, birthday=null, address=河南郑州]
```

五 、思考题：

动态 SQL 的优点和不足?

六 、实验作业要求：

⑴ 验目的：

⑵ 验内容：

⑶ 验结果：可以是运行结果截图或其他形式的结果展示

⑷ 问题及解决：实验中遇到的问题及解决方法。

⑸ 回答思考题提出的问题。