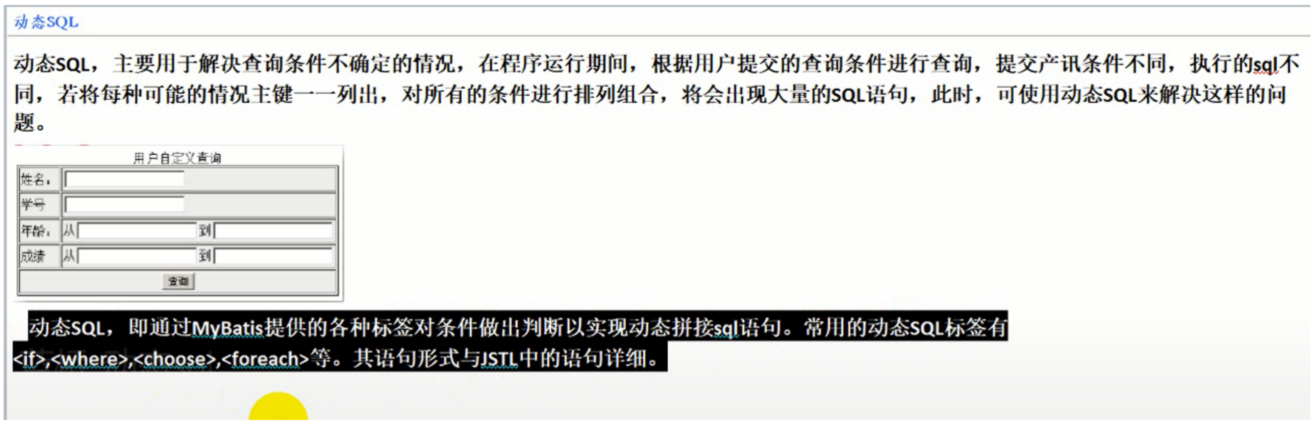
动态SQL

1、概要



动态SQL

动态SQL，主要用于解决查询条件不确定的情况，在程序运行期间，根据用户提交的查询条件进行查询，提交产讯条件不同，执行的sql不同，若将每种可能的情况主键一一列出，对所有的条件进行排列组合，将会出现大量的SQL语句，此时，可使用动态SQL来解决这样的问题。

动态SQL，即通过MyBatis提供的各种标签对条件做出判断以实现动态拼接sql语句。常用的动态SQL标签有<if>,<where>,<choose>,<foreach>等。其语句形式与JSTL中的语句详细。

2、实例

- if标签
  - StudentMapper类

```
//多条件查询:动态代理
    public List<Student> findStuByAll(int id,String sname ,int sage ,String
ssex);
```

  - StudentMapper.xml

```
<select id="findStuByAll" resultType="Student">
        select * from stu where 1 = 1
        <if test="arg0 != 0">
            AND id = #{arg0}
        </if>
        <if test="arg0 != null">
            AND sname like '%' #{arg1} '%'
        </if>
    </select>
```

  - studentTest类

```
 @Test
    //多条件查询: 动态代理
    public void findStuByAllTest(){
        List<Student> studentList = studentMapper.findStuByAll(0,"沛",0,null);
        for (Student str:studentList){
            System.out.println(str);
        }
    }
```

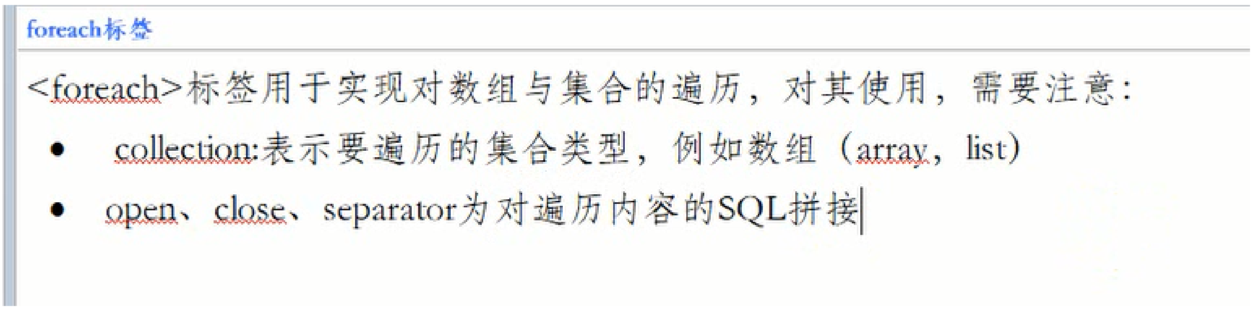
- where标签
  - StudentMapper类
  - StudentMapper.xml

```xml
<!--: 多条件查询:动态代理-->
    <select id="findStuByAll" resultType="Student">
        select * from stu
        <where>
            <if test="arg0 != 0">
                AND id = #{arg0}
            </if>
            <if test="arg0 != null">
                AND sname like '%' #{arg1} '%'
            </if>
        </where>
    </select>
```

  - studentTest类
- choose标签
  - StudentMapper类
  - StudentMapper.xml

```xml
<!--: 多条件查询:动态代理-->
    <select id="findStuByAll" resultType="Student">
        select * from stu
        <where>
            <choose>
                <when test="arg0 != 0">
                    AND id = #{arg0}
                </when>
                <when test="arg0 != null">
                    AND sname like '%' #{arg1} '%'
                </when>
            </choose>
        </where>
    </select>
```

  - studentTest类
- foreach标签



foreach标签

<foreach>标签用于实现对数组与集合的遍历，对其使用，需要注意：

- collection:表示要遍历的集合类型，例如数组（array，list）
- open、close、separator为对遍历内容的SQL拼接

  - 遍历数组

        - StudentMapper类

```
//多条件查询：动态代理,数组
    public List<Student> findStudentByInCondition(int[] arr);
```

        - StudentMapper.xml

```
 <!--: 多条件查询:动态代理，数组-->
    <select id="findStudentByInCondition" resultType="Student">
        select * from stu
        <where>
            id in
            <foreach collection="array" item="id" open="(" separator=","
close=")">
                #{id}
            </foreach>
        </where>
    </select>
```

        - studentTest类

```
@Test
    //多条件查询：动态代理，数组
    public void findStudentByInConditionTest(){
       List<Student> studentList =
studentMapper.findStudentByInCondition(new int[]{1,2,3,4,5});
        for (Student str:studentList){
            System.out.println(str);
        }
    }
```

    - 遍历基本类型的List
        - StudentMapper类

```
//多条件查询：动态代理,list集合
    public List<Student> findStudentByInCondition(List<Integer> arr);
```

        - StudentMapper.xml

```
 <!--: 多条件查询:动态代理, list-->
    <select id="findStudentByInCondition" resultType="Student">
        select * from stu
        <where>
            id in
            <foreach collection="list" item="id" open="(" separator=","
close=")">
                #{id}
            </foreach>
        </where>
    </select>
```

- studentTest类

```
@Test
    //多条件查询: 动态代理, list
    public void findStudentByInConditionTest(){
        List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(3);
        list.add(5);
        list.add(7);
        list.add(9);
       List<Student> studentList =
studentMapper.findStudentByInCondition(list);
        for (Student str:studentList){
            System.out.println(str);
        }
    }
```

- 遍历自定义类型的List
  - StudentMapper类

```
//多条件查询: 动态代理,自定义类型地list
    public List<Student> findStudentByInCondition(List<Student> arr);
```

  - StudentMapper.xml

```
<!--: 多条件查询:动态代理, 自定义类型-->
    <select id="findStudentByInCondition" resultType="Student">
        select * from stu
        <where>
            id in
            <foreach collection="list" item="student" open="("
separator="," close=")">
                #{student.id}
            </foreach>
        </where>
    </select>
```

        - studentTest类

          ```
          //多条件查询：动态代理，自定义类型Student
              public void findStudentByInConditionTest(){
                  List<Student> list = new ArrayList<>();
                  Student stu1 = new Student();
                  stu1.setId(1);
                  Student stu2 = new Student();
                  stu2.setId(3);
                  Student stu3 = new Student();
                  stu3.setId(5);
                  Student stu4 = new Student();
                  stu4.setId(7);
                  Student stu5 = new Student();
                  stu5.setId(9);
                  list.add(stu1);
                  list.add(stu2);
                  list.add(stu3);
                  list.add(stu4);
                  list.add(stu5);
                 List<Student> studentList =
          studentMapper.findStudentByInCondition(list);
                  for (Student str:studentList){
                      System.out.println(str);
                  }
              }
          ```

- 和

  ```
  <!--sql片段-->
      <sql id="baseSql">
          select * from stu
      </sql>
      <!--：多条件查询:动态代理，自定义类型-->
      <select id="findStudentByInCondition" resultType="Student">
          <!--包含sql片段-->
          <include refid="baseSql"></include>
          <where>
              id in
              <foreach collection="list" item="student" open="(" separator=","
  close=")">
                  #{student.id}
              </foreach>
          </where>
      </select>
  ```

- 当当

3、高级查询(多表查询)

- 关联查询：查询内容涉及具有多个关系的多个表时
    - 一对多

        - 多表链接查询
        - 多表单独查询
    - 多对一
        - 多表链接查询
        - 多表单独查询
    - 多对多
- 延迟加载
- 查询缓存
    - 一级缓存
    - 二级缓存
        - 验证增删改对二级缓存的影响
        - 二级缓存关闭
        - 级缓存的使用原则

4、当当