这个动态SQL呢是mybatis的一个强大的特性，使用了动态SQL后sql语句的拼接变得更加灵活

（目前的自我体会是动态SQL只是将sql变灵活，貌似并不是必须的，只能说我目前接触的层面比较低吧，等日后应该就会有体会了）

常用的动态SQL元素

if

choose

where

set

foreach

bind

if：我也能判断

```
<select id="findUserById" resultType="user">
            select * from user where
            <if test="id != null">
                id=#{id}
            </if>
             and deleteFlag=0;
</select>
```

可以看出这里的sql语句加了if，如表面理解就是id不为null的时候才会拼接id=#｛id｝，

但是明眼人就能看出这里的问题了，要是id为null，sql语句不就成where and不是明显有问题的吗，不急，接下来我就要讲这个处理了，动态sql的魅力还没完全展现呢

where：这就是解决上述问题的方法

```
<select id="findUserById" resultType="user">
            select * from user
            <where>
                <if test="id != null">
                    id=#{id}
```

```
            </if>
            and deleteFlag=0;
        </where>
 </select>
```

有人看了可能要懵逼了，你把where用where标签替代了下，能产生什么奇妙的效果？还真是，mybatis对where做过处理，如果where后紧跟了and或or都会直接无视掉，就是这么奇妙，就是这么神奇

set：处理另外一种形式的良策

```
<update id="updateUser" parameterType="com.dy.entity.User">
        update user set
        <if test="name != null">
            name = #{name},
        </if>
        <if test="password != null">
            password = #{password},
        </if>
        <if test="age != null">
            age = #{age}
        </if>
        <where>
            <if test="id != null">
                id = #{id}
            </if>
            and deleteFlag = 0;
        </where>
</update>
```

看完这个例子很多朋友应该清除我要说什么了，没错，如果最后一个if不拼接，不就多出一个“，”了吗，这时我们这么处理

```
<update id="updateUser" parameterType="com.dy.entity.User">
        update user
      <set>
        <if test="name != null">name = #{name},</if>
          <if test="password != null">password = #{password},</if>
          <if test="age != null">age = #{age},</if>
      </set>
        <where>
            <if test="id != null">
                id = #{id}
```

```
            </if>
            and deleteFlag = 0;
        </where>
</update>
```

和where一个意思，这个set也是mybatis做过处理的，最后多出的“，”会被忽略掉

foreach：一句话，强大的可以，它允许了查询的参数可以是list类型，相当于会遍历list中的所有值都作为一个条件查询一条结果

```
<select id="selectPostIn" resultType="domain.blog.Post">
  SELECT *
  FROM POST P
  WHERE ID in
  <foreach item="item" index="index" collection="list"
      open="(" separator="," close=")">
        #{item}
  </foreach>
</select>
```

如这个例子，遍历了list中所有结果，封装在“（）”中，之间以“，”隔开，整个作为sql语句中in的条件集合

choose：让条件有了可选择性

```
<select id="findActiveBlogLike"
     resultType="Blog">
  SELECT * FROM BLOG WHERE state = ‘ACTIVE’
  <choose>
    <when test="title != null">
      AND title like #{title}
    </when>
    <when test="author != null and author.name != null">
      AND author_name like #{author.name}
    </when>
    <otherwise>
      AND featured = 1
    </otherwise>
  </choose>
</select>
```

如这个例子， 当title和author都不为null的时候， 那么选择二选一（前者优先）， 如果都为null，那么就选择 otherwise中的， 如果tilte和author只有一个不为null，那么就选择不为null的那个