第8章 移动端开发-体检预约

1. 体检预约流程

用户可以通过如下操作流程进行体检预约：

1、在移动端首页点击体检预约，页面跳转到套餐列表页面

2、在套餐列表页面点击要预约的套餐，页面跳转到套餐详情页面

3、在套餐详情页面点击立即预约，页面跳转到预约页面

4、在预约页面录入体检人信息，包括手机号，点击发送验证码

5、在预约页面录入收到的手机短信验证码，点击提交预约，完成体检预约

2. 体检预约

2.1 页面调整

在预约页面（/pages/orderInfo.html）进行调整

2.1.1 展示预约的套餐信息

第一步：从请求路径中获取当前套餐的id

```
<script>
  var id = getUrlParam("id");//套餐id
</script>
```

第二步：定义模型数据setmeal，用于套餐数据展示

```
var vue = new Vue({
  el:'#app',
  data:{
    setmeal:{},//套餐信息
    orderInfo:{
      setmealId:id,
      sex:'1'
    }//预约信息
  }
});
```

```
<div class="card">
  <div class="">
    <img :src="'http://pqjroc654.bkt.clouddn.com/'+setmeal.img" width="100%"
height="100%" />
  </div>
  <div class="project-text">
    <h4 class="tit">{{setmeal.name}}</h4>
    <p class="subtit">{{setmeal.remark}}</p>
    <p class="keywords">
```

```
        <span>{{setmeal.age}}</span>
    </p>
  </div>
  <div class="project-know">
    <a href="orderNotice.html" class="link-page">
      <i class="icon-ask-circle"><span class="path1"></span><span class="path2">
</span></i>
      <span class="word">预约须知</span>
      <span class="arrow"><i class="icon-rit-arrow"></i></span>
    </a>
  </div>
</div>
```

第三步：在VUE的钩子函数中发送ajax请求，根据id查询套餐信息

```
mounted(){
  axios.post("/setmeal/findById.do?id=" + id).then((response) => {
    this.setmeal = response.data.data;
  });
}
```

2.1.2 手机号校验

第一步：在页面导入的healthmobile.js文件中已经定义了校验手机号的方法

```
/**
 * 手机号校验
 1--以1为开头；
 2--第二位可为3,4,5,7,8,中的任意一位；
 3--最后以0-9的9个整数结尾。
 */
function checkTelephone(telephone) {
    var reg=/^[1][3,4,5,7,8][0-9]{9}$/;
    if (!reg.test(telephone)) {
        return false;
    } else {
        return true;
    }
}
```

第二步：为发送验证码按钮绑定事件sendValidateCode

```
<div class="input-row">
  <label>手机号</label>
  <input v-model="orderInfo.telephone"
         type="text" class="input-clear" placeholder="请输入手机号">
  <input style="font-size: x-small;"
         id="validateCodeButton" @click="sendValidateCode()" type="button"
value="发送验证码">
</div>
```

```
  //获取用户输入的手机号
  var telephone = this.orderInfo.telephone;
  //校验手机号输入是否正确
  if (!checkTelephone(telephone)) {
    this.$message.error('请输入正确的手机号');
    return false;
  }
}
```

2.1.3 30秒倒计时效果

前面在sendValidateCode方法中进行了手机号校验，如果校验通过，需要显示30秒倒计时效果

```
//发送验证码
sendValidateCode(){
  //获取用户输入的手机号
  var telephone = this.orderInfo.telephone;
  //校验手机号输入是否正确
  if (!checkTelephone(telephone)) {
    this.$message.error('请输入正确的手机号');
    return false;
  }
  validateCodeButton = $("#validateCodeButton")[0];
  clock = window.setInterval(doLoop, 1000); //一秒执行一次
}
```

其中，validateCodeButton和clock是在healthmobile.js文件中定义的变量，doLoop是在healthmobile.js文件中定义的方法

```
var clock = '';//定时器对象，用于页面30秒倒计时效果
var nums = 30;
var validateCodeButton;
//基于定时器实现30秒倒计时效果
function doLoop() {
    validateCodeButton.disabled = true;//将按钮置为不可点击
    nums--;
    if (nums > 0) {
        validateCodeButton.value = nums + '秒后重新获取';
    } else {
        clearInterval(clock); //清除js定时器
        validateCodeButton.disabled = false;
        validateCodeButton.value = '重新获取验证码';
        nums = 30; //重置时间
    }
}
```

2.1.4 发送ajax请求

在按钮上显示30秒倒计时效果的同时，需要发送ajax请求，在后台给用户发送手机验证码

```
//发送验证码
sendValidateCode(){
  //获取用户输入的手机号
```

```
  if (!checkTelephone(telephone)) {
    this.$message.error('请输入正确的手机号');
    return false;
  }
  validateCodeButton = $("#validateCodeButton")[0];
  clock = window.setInterval(doLoop, 1000); //一秒执行一次
  axios.post("/validateCode/send4Order.do?telephone=" +
telephone).then((response) => {
    if(!response.data.flag){
      //验证码发送失败
      this.$message.error('验证码发送失败，请检查手机号输入是否正确');
    }
  });
}
```

创建ValidateCodeController，提供方法发送短信验证码，并将验证码保存到redis

```
package com.itheima.controller;

import com.aliyuncs.exceptions.ClientException;
import com.itheima.constant.MessageConstant;
import com.itheima.constant.RedisConstant;
import com.itheima.constant.RedisMessageConstant;
import com.itheima.entity.Result;
import com.itheima.utils.JedisUtils;
import com.itheima.utils.SMSUtils;
import com.itheima.utils.ValidateCodeUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import java.util.Random;

/**
 * 短信验证码
 */
@RestController
@RequestMapping("/validateCode")
public class ValidateCodeController {
    @Autowired
    private JedisPool jedisPool;

    //体检预约时发送手机验证码
    @RequestMapping("/send4Order")
    public Result send4Order(String telephone){
        Integer code = ValidateCodeUtils.generateValidateCode(4);//生成4位数字验证
码
        try {
            //发送短信

 SMSUtils.sendShortMessage(SMSUtils.VALIDATE_CODE,telephone,code.toString());
        } catch (ClientException e) {
            e.printStackTrace();
            //验证码发送失败
```

```
            System.out.println("发送的手机验证码为：" + code);
            //将生成的验证码缓存到redis
            jedisPool.getResource().setex(
              telephone + RedisMessageConstant.SENDTYPE_ORDER,5 *
60,code.toString());
            //验证码发送成功
            return new Result(true,MessageConstant.SEND_VALIDATECODE_SUCCESS);
        }
}
```

2.1.5 日历展示

页面中使用DatePicker控件来展示日历。根据需求，最多可以提前一个月进行体检预约，所以日历控件只展示未来一个月的日期

```
<div class="date">
  <label>体检日期</label>
  <i class="icon-date" class="picktime"></i>
  <input v-model="orderInfo.orderDate" type="text" class="picktime" readonly>
</div>
```

```
<script>
  //日期控件
  var calendar = new datePicker();
  calendar.init({
    'trigger': '.picktime',/*按钮选择器，用于触发弹出插件*/
    'type': 'date',/*模式：date日期；datetime日期时间；time时间；ym年月；*/
    'minDate': getSpecifiedDate(new Date(),1),/*最小日期*/
    'maxDate': getSpecifiedDate(new Date(),30),/*最大日期*/
    'onSubmit': function() { /*确认时触发事件*/},
    'onClose': function() { /*取消时触发事件*/ }
  });
</script>
```

其中getSpecifiedDate方法定义在healthmobile.js文件中

```
//获得指定日期后指定天数的日期
function getSpecifiedDate(date,days) {
    date.setDate(date.getDate() + days);//获取指定天之后的日期
    var year = date.getFullYear();
    var month = date.getMonth() + 1;
    var day = date.getDate();
    return (year + "-" + month + "-" + day);
}
```

2.1.6 提交预约请求

为提交预约按钮绑定事件

```
<div class="box-button">
  <button @click="submitOrder()" type="button" class="btn order-btn">提交预约
</button>
</div>
```

```
  //校验身份证号格式
  if(!checkIdCard(this.orderInfo.idCard)){
    this.$message.error('身份证号码输入错误，请重新输入');
    return ;
  }
  axios.post("/order/submit.do",this.orderInfo).then((response) => {
    if(response.data.flag){
      //预约成功，跳转到预约成功页面
      window.location.href="orderSuccess.html?orderId=" + response.data.data;
    }else{
      //预约失败，提示预约失败信息
      this.$message.error(response.data.message);
    }
  });
}
```

其中checkIdCard方法是在healthmobile.js文件中定义的

```
/**
 * 身份证号码校验
 * 身份证号码为15位或者18位，15位时全为数字，18位前17位为数字，最后一位是校验位，可能为数字或字符X
 */
function checkIdCard(idCard){
    var reg = /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/;
    if(reg.test(idCard)){
        return true;
    }else{
        return false;
    }
}
```

2.2 后台代码

2.2.1 Controller

在health_mobile工程中创建OrderController并提供submitOrder方法

```
package com.itheima.controller;

import com.alibaba.dubbo.config.annotation.Reference;
import com.aliyuncs.exceptions.ClientException;
import com.itheima.constant.MessageConstant;
import com.itheima.constant.RedisConstant;
import com.itheima.constant.RedisMessageConstant;
import com.itheima.entity.Result;
import com.itheima.pojo.Member;
import com.itheima.pojo.Order;
import com.itheima.pojo.Setmeal;
import com.itheima.service.OrderService;
import com.itheima.utils.JedisUtils;
import com.itheima.utils.SMSUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
```

```
import redis.clients.jedis.JedisPool;
import java.util.HashMap;
import java.util.Map;

/**
 * 体检预约
 */
@RestController
@RequestMapping("/order")
public class OrderController {
    @Reference
    private OrderService orderService;
    @Autowired
    private JedisPool jedisPool;

    /**
     * 体检预约
     * @param map
     * @return
     */
    @RequestMapping("/submit")
    public Result submitOrder(@RequestBody Map map){
        String telephone = (String) map.get("telephone");
        //从Redis中获取缓存的验证码，key为手机号+RedisConstant.SENDTYPE_ORDER
        String codeInRedis = jedisPool.getResource().get(
          telephone + RedisMessageConstant.SENDTYPE_ORDER);
        String validateCode = (String) map.get("validateCode");
        //校验手机验证码
        if(codeInRedis == null || !codeInRedis.equals(validateCode)){
            return new Result(false, MessageConstant.VALIDATECODE_ERROR);
        }
        Result result =null;
        //调用体检预约服务
        try{
            map.put("orderType", Order.ORDERTYPE_WEIXIN);
            result = orderService.order(map);
        }catch (Exception e){
            e.printStackTrace();
            //预约失败
            return result;
        }
        if(result.isFlag()){
            //预约成功，发送短信通知
            String orderDate = (String) map.get("orderDate");
            try {

 SMSUtils.sendShortMessage(SMSUtils.ORDER_NOTICE,telephone,orderDate);
            } catch (ClientException e) {
                e.printStackTrace();
            }
        }
        return result;
    }
}
```

在health_interface工程中创建体检预约服务接口OrderService并提供预约方法

```
package com.itheima.service;

import com.itheima.entity.Result;
import java.util.Map;
/**
 * 体检预约服务接口
 */
public interface OrderService {
    //体检预约
    public Result order(Map map) throws Exception;
}
```

2.2.3 服务实现类

在health_service_provider工程中创建体检预约服务实现类OrderServiceImpl并实现体检预约方法。

体检预约方法处理逻辑比较复杂，需要进行如下业务处理：

1、检查用户所选择的预约日期是否已经提前进行了预约设置，如果没有设置则无法进行预约

2、检查用户所选择的预约日期是否已经约满，如果已经约满则无法预约

3、检查用户是否重复预约（同一个用户在同一天预约了同一个套餐），如果是重复预约则无法完成再次预约

4、检查当前用户是否为会员，如果是会员则直接完成预约，如果不是会员则自动完成注册并进行预约

5、预约成功，更新当日的已预约人数

实现代码如下：

```
package com.itheima.service;

import com.alibaba.dubbo.config.annotation.Service;
import com.itheima.constant.MessageConstant;
import com.itheima.dao.MemberDao;
import com.itheima.dao.OrderDao;
import com.itheima.dao.OrderSettingDao;
import com.itheima.dao.SetmealDao;
import com.itheima.entity.Result;
import com.itheima.pojo.Member;
import com.itheima.pojo.Order;
import com.itheima.pojo.OrderSetting;
import com.itheima.pojo.Setmeal;
import com.itheima.utils.DateUtils;
import org.apache.poi.ss.usermodel.DateUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
 */
@Service(interfaceClass = OrderService.class)
@Transactional
public class OrderServiceImpl implements OrderService{
    @Autowired
    private OrderSettingDao orderSettingDao;
    @Autowired
    private MemberDao memberDao;
    @Autowired
    private OrderDao orderDao;

    //体检预约
    public Result order(Map map) throws Exception {
        //检查当前日期是否进行了预约设置
        String orderDate = (String) map.get("orderDate");
        Date date = DateUtils.parseString2Date(orderDate);
        OrderSetting orderSetting = orderSettingDao.findByOrderDate(date);
        if(orderSetting == null){
            return new Result(false,
MessageConstant.SELECTED_DATE_CANNOT_ORDER);
        }

        //检查预约日期是否预约已满
        int number = orderSetting.getNumber();//可预约人数
        int reservations = orderSetting.getReservations();//已预约人数
        if(reservations >= number){
            //预约已满，不能预约
            return new Result(false,MessageConstant.ORDER_FULL);
        }

        //检查当前用户是否为会员，根据手机号判断
        String telephone = (String) map.get("telephone");
        Member member = memberDao.findByTelephone(telephone);
        //防止重复预约
        if(member != null){
            Integer memberId = member.getId();
            int setmealId = Integer.parseInt((String) map.get("setmealId"));
            Order order = new Order(memberId,date,null,null,setmealId);
            List<Order> list = orderDao.findByCondition(order);
            if(list != null && list.size() > 0){
                //已经完成了预约，不能重复预约
                return new Result(false,MessageConstant.HAS_ORDERED);
            }
        }

        //可以预约，设置预约人数加一
        orderSetting.setReservations(orderSetting.getReservations()+1);
        orderSettingDao.editReservationsByOrderDate(orderSetting);

        if(member == null){
            //当前用户不是会员，需要添加到会员表
            member = new Member();
            member.setName((String) map.get("name"));
            member.setPhoneNumber(telephone);
            member.setIdCard((String) map.get("idCard"));
            member.setSex((String) map.get("sex"));
```

```
        }

        //保存预约信息到预约表
        Order order = new Order(member.getId(),
                                date,
                                (String)map.get("orderType"),
                                Order.ORDERSTATUS_NO,
                                Integer.parseInt((String)
map.get("setmealId")));
        orderDao.add(order);

        return new Result(true,MessageConstant.ORDER_SUCCESS,order.getId());
    }
}
```

2.2.4 Dao接口

```
package com.itheima.dao;

import com.itheima.pojo.OrderSetting;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public interface OrderSettingDao {
    public void add(OrderSetting orderSetting);
    //更新可预约人数
    public void editNumberByOrderDate(OrderSetting orderSetting);
    //更新已预约人数
    public void editReservationsByOrderDate(OrderSetting orderSetting);
    public long findCountByOrderDate(Date orderDate);
    //根据日期范围查询预约设置信息
    public List<OrderSetting> getOrderSettingByMonth(Map date);
    //根据预约日期查询预约设置信息
    public OrderSetting findByOrderDate(Date orderDate);
}
```

```
package com.itheima.dao;

import com.github.pagehelper.Page;
import com.itheima.pojo.Member;
import java.util.List;

public interface MemberDao {
    public List<Member> findAll();
    public Page<Member> selectByCondition(String queryString);
    public void add(Member member);
    public void deleteById(Integer id);
    public Member findById(Integer id);
    public Member findByTelephone(String telephone);
    public void edit(Member member);
    public Integer findMemberCountBeforeDate(String date);
```

```
    public Integer findMemberTotalCount();
}
```

```
package com.itheima.dao;

import com.itheima.pojo.Order;
import java.util.List;
import java.util.Map;

public interface OrderDao {
    public void add(Order order);
    public List<Order> findByCondition(Order order);
}
```

2.2.5 Mapper映射文件

OrderSettingDao.xml

```
<!--根据日期查询预约设置信息-->
<select id="findByOrderDate" parameterType="date"
resultType="com.itheima.pojo.OrderSetting">
  select * from t_ordersetting where orderDate = #{orderDate}
</select>
<!--更新已预约人数-->
<update id="editReservationsByOrderDate"
parameterType="com.itheima.pojo.OrderSetting">
  update t_ordersetting set reservations = #{reservations} where orderDate = #
{orderDate}
</update>
```

MemberDao.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="com.itheima.dao.MemberDao" >
    <select id="findAll" resultType="com.itheima.pojo.Member">
        select * from t_member
    </select>

    <!--根据条件查询-->
    <select id="selectByCondition"
            parameterType="string" resultType="com.itheima.pojo.Member">
        select * from t_member
        <if test="value != null and value.length > 0">
            where fileNumber = #{value} or phoneNumber = #{value} or name = #
{value}
        </if>
    </select>

    <!--新增会员-->
    <insert id="add" parameterType="com.itheima.pojo.Member">
```

```
            SELECT LAST_INSERT_ID()
        </selectKey>
        insert into
            t_member
              (fileNumber,name,sex,idCard,phoneNumber,
               regTime,password,email,birthday,remark)
             values
              (#{fileNumber},#{name},#{sex},#{idCard},#{phoneNumber},
               #{regTime},#{password},#{email},#{birthday},#{remark})
    </insert>

    <!--删除会员-->
    <delete id="deleteById" parameterType="int">
        delete from t_member where id = #{id}
    </delete>

    <!--根据id查询会员-->
    <select id="findById" parameterType="int"
resultType="com.itheima.pojo.Member">
        select * from t_member where id = #{id}
    </select>

    <!--根据id查询会员-->
    <select id="findByTelephone"
            parameterType="string" resultType="com.itheima.pojo.Member">
        select * from t_member where phoneNumber = #{phoneNumber}
    </select>

    <!--编辑会员-->
    <update id="edit" parameterType="com.itheima.pojo.Member">
        update t_member
        <set>
            <if test="fileNumber != null">
                fileNumber = #{fileNumber},
            </if>
            <if test="name != null">
                name = #{name},
            </if>
            <if test="sex != null">
                sex = #{sex},
            </if>
            <if test="idCard != null">
                idCard = #{idCard},
            </if>
            <if test="phoneNumber != null">
                phoneNumber = #{phoneNumber},
            </if>
            <if test="regTime != null">
                regTime = #{regTime},
            </if>
            <if test="password != null">
                password = #{password},
            </if>
            <if test="email != null">
                email = #{email},
            </if>
```

```
            </if>
            <if test="remark != null">
                remark = #{remark},
            </if>
        </set>
        where id = #{id}
    </update>

    <!--根据日期统计会员数，统计指定日期之前的会员数-->
    <select id="findMemberCountBeforeDate" parameterType="string"
resultType="int">
        select count(id) from t_member where regTime &lt;= #{value}
    </select>

    <!--根据日期统计会员数-->
    <select id="findMemberCountByDate" parameterType="string" resultType="int">
        select count(id) from t_member where regTime = #{value}
    </select>

    <!--根据日期统计会员数，统计指定日期之后的会员数-->
    <select id="findMemberCountAfterDate" parameterType="string"
resultType="int">
        select count(id) from t_member where regTime &gt;= #{value}
    </select>

    <!--总会员数-->
    <select id="findMemberTotalCount" resultType="int">
        select count(id) from t_member
    </select>
</mapper>
```

OrderDao.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="com.itheima.dao.OrderDao" >
    <resultMap id="baseResultMap" type="com.itheima.pojo.Order">
        <id column="id" property="id"/>
        <result column="member_id" property="memberId"/>
        <result column="orderDate" property="orderDate"/>
        <result column="orderType" property="orderType"/>
        <result column="orderStatus" property="orderStatus"/>
        <result column="setmeal_id" property="setmealId"/>
    </resultMap>
    <!--新增-->
    <insert id="add" parameterType="com.itheima.pojo.Order">
        <selectKey resultType="java.lang.Integer" order="AFTER"
keyProperty="id">
            SELECT LAST_INSERT_ID()
        </selectKey>
        insert into
            t_order
        (member_id,orderDate,orderType,orderStatus,setmeal_id)
            values
```

```
    <!--动态条件查询-->
    <select id="findByCondition"
            parameterType="com.itheima.pojo.Order"
            resultMap="baseResultMap">
        select * from t_order
        <where>
            <if test="id != null">
                and id = #{id}
            </if>
            <if test="memberId != null">
                and member_id = #{memberId}
            </if>
            <if test="orderDate != null">
                and orderDate = #{orderDate}
            </if>
            <if test="orderType != null">
                and orderType = #{orderType}
            </if>
            <if test="orderStatus != null">
                and orderStatus = #{orderStatus}
            </if>
            <if test="setmealId != null">
                and setmeal_id = #{setmealId}
            </if>
        </where>
    </select>
</mapper>
```

3. 预约成功页面展示

前面已经完成了体检预约，预约成功后页面会跳转到成功提示页面（orderSuccess.html）并展示预约的相关信息（体检人、体检套餐、体检时间等）。

3.1 页面调整

提供orderSuccess.html页面，展示预约成功后相关信息

```
<div class="info-title">
  <span class="name">体检预约成功</span>
</div>
<div class="notice-item">
  <div class="item-title">预约信息</div>
  <div class="item-content">
    <p>体检人：{{orderInfo.member}}</p>
    <p>体检套餐：{{orderInfo.setmeal}}</p>
    <p>体检日期：{{orderInfo.orderDate}}</p>
    <p>预约类型：{{orderInfo.orderType}}</p>
  </div>
</div>
```

```
<script>
  //从请求URL根据参数名获取对应值，orderId为预约id
```

```
<script>
  var vue = new Vue({
    el:'#app',
    data:{
      orderInfo:{}
    },
    mounted(){
      axios.post("/order/findById.do?id=" + id).then((response) => {
        this.orderInfo = response.data.data;
      });
    }
  });
</script>
```

3.2 后台代码

3.2.1 Controller

在OrderController中提供findById方法，根据预约id查询预约相关信息

```
/**
 * 根据id查询预约信息，包括套餐信息和会员信息
 * @param id
 * @return
*/
@RequestMapping("/findById")
public Result findById(Integer id){
  try{
    Map map = orderService.findById(id);
    //查询预约信息成功
    return new Result(true,MessageConstant.QUERY_ORDER_SUCCESS,map);
  }catch (Exception e){
    e.printStackTrace();
    //查询预约信息失败
    return new Result(false,MessageConstant.QUERY_ORDER_FAIL);
  }
}
```

3.2.2 服务接口

在OrderService服务接口中扩展findById方法

```
//根据id查询预约信息，包括体检人信息、套餐信息
public Map findById(Integer id) throws Exception;
```

3.2.3 服务实现类

在OrderServiceImpl服务实现类中实现findById方法

```
public Map findById(Integer id) throws Exception {
  Map map = orderDao.findById4Detail(id);
  if(map != null){
    //处理日期格式
    Date orderDate = (Date) map.get("orderDate");
    map.put("orderDate",DateUtils.parseDate2String(orderDate));
  }
  return map;
}
```

3.2.4 Dao接口

在OrderDao接口中扩展findById4Detail方法

```
public Map findById4Detail(Integer id);
```

3.2.5 Mapper映射文件

在OrderDao.xml映射文件中提供SQL语句

```
<!--根据预约id查询预约信息，包括体检人信息、套餐信息-->
<select id="findById4Detail" parameterType="int" resultType="map">
  select m.name member ,s.name setmeal,o.orderDate orderDate,o.orderType
orderType
  from
  t_order o,
  t_member m,
  t_setmeal s
  where o.member_id=m.id and o.setmeal_id=s.id and o.id=#{id}
</select>
```