1. 移动端开发-体检预约

学习目标：

1：移动端的开发

2：移动端的页面、详情页面展示

# 第一章. 移动端开发

## 移动端开发方式

随着移动互联网的兴起和手机的普及，目前移动端应用变得愈发重要，成为了各个商家的必争之地。例如，我们可以使用手机购物、支付、打车、玩游戏、订酒店、购票等，以前只能通过PC端完成的事情，现在通过手机都能够实现，而且更加方便，而这些都需要移动端开发进行支持，那如何进行移动端开发呢？

移动端开发主要有三种方式：

1、基于手机API开发（原生APP）

2、基于手机浏览器开发（移动web）

3、混合开发（混合APP）

### 基于手机API开发

手机端使用手机API，例如使用Android、ios 等进行开发，服务端只是一个数据提供者。手机端请求服务端获取数据（json、xml格式）并在界面进行展示。这种方式相当于传统开发中的C/S模式，即需要在手机上安装一个客户端软件。

这种方式需要针对不同的手机系统分别进行开发，目前主要有以下几个平台：

1、苹果ios系统版本，开发语言是Objective-C

2、安卓Android系统版本，开发语言是Java

3、微软Windows phone系统版本，开发语言是C#

4、塞班symbian系统版本，开发语言是C++

此种开发方式举例：手机淘宝、抖音、今日头条、大众点评

### 基于手机浏览器开发

生存在浏览器中的应用，基本上可以说是触屏版的网页应用。这种开发方式相当于传统开发中的B/S模式，也就是手机上不需要额外安装软件，直接基于手机上的浏览器进行访问。这就需要我们编写的html页面需要根据不同手机的尺寸进行自适应调节，目前比较流行的是html5开发。除了直接通过手机浏览器访问，还可以将页面内嵌到一些应用程序中，例如通过微信公众号访问html5页面。

这种开发方式不需要针对不同的手机系统分别进行开发，只需要开发一个版本，就可以在不同的手机上正常访问。

本项目会通过将我们开发的html5页面内嵌到微信公众号这种方式进行开发。

### 混合开发

是半原生半Web的混合类App。需要下载安装，看上去类似原生App，访问的内容是Web网页。其实就是把HTML5页面嵌入到一个原生容器里面。

## 微信公众号开发

要进行微信公众号开发，首先需要访问微信公众平台，官网：<https://mp.weixin.qq.com/>。

### 帐号分类

在微信公众平台可以看到，有四种帐号类型：服务号、订阅号、小程序、企业微信（原企业号）。





本项目会选择订阅号这种方式进行公众号开发。

### 注册帐号

要开发微信公众号，首先需要注册成为会员，然后就可以登录微信公众平台进行自定义菜单的设置。

注册页面：<https://mp.weixin.qq.com/cgi-bin/registermidpage?action=index&lang=zh_CN&token=>



选择订阅号进行注册：

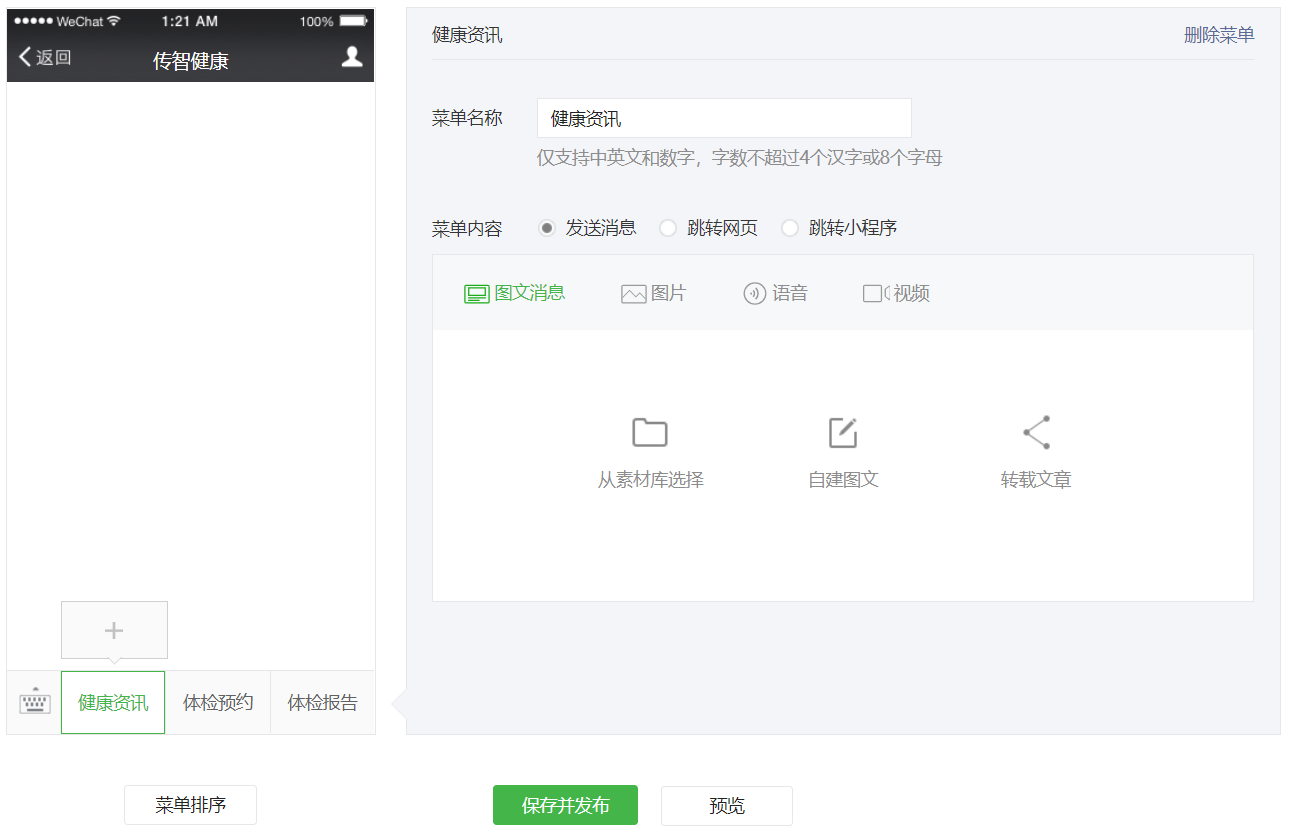
使用邮箱注册。



输入邮箱、邮箱验证码、密码、确认密码等按照页面流程进行注册

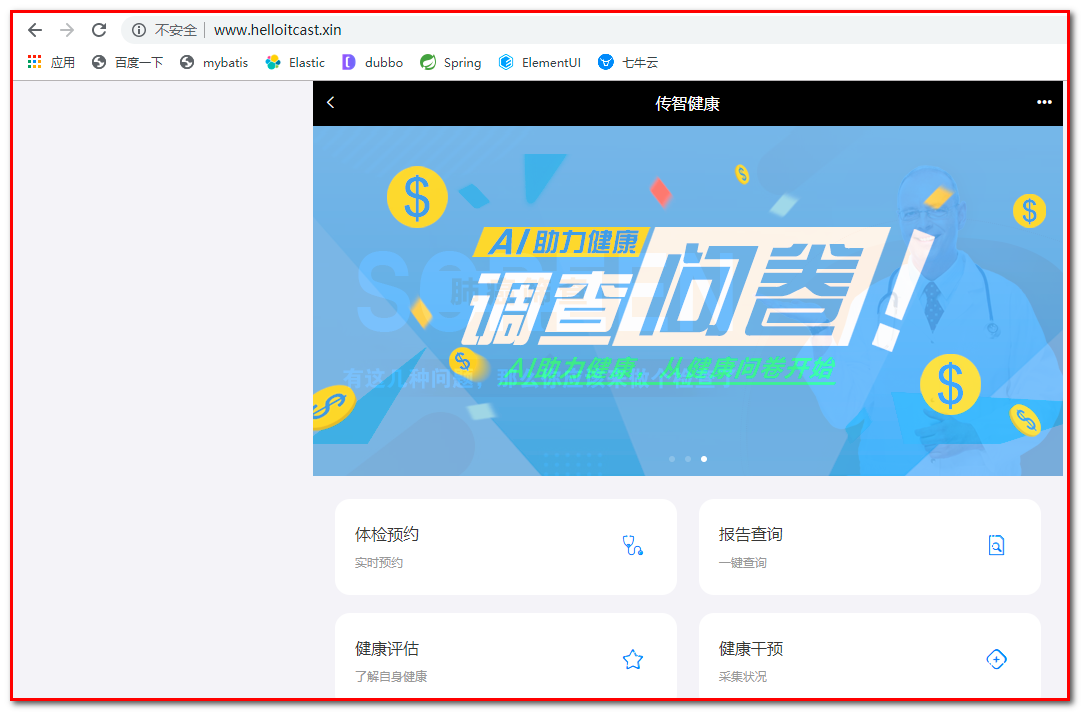
### 自定义菜单

注册成功后就可以使用注册的邮箱和设置的密码进行登录，登录成功后点击左侧“自定义菜单”进入自定义菜单页面



在自定义菜单页面可以根据需求创建一级菜单和二级菜单，其中一级菜单最多可以创建3个，每个一级菜单下面最多可以创建5个二级菜单。每个菜单由菜单名称和菜单内容组成，其中菜单内容有3种形式：发送消息、跳转网页、跳转小程序。

跳转页面：<http://www.helloitcast.xin/>



关注微信公众号

1：搜索传智健康



2：扫描：



### 上线要求

如果是个人用户身份注册的订阅号，则自定义菜单的菜单内容不能进行跳转网页，因为个人用户目前不支持微信认证，而跳转网页需要微信认证之后才有权限。

如果是企业用户，首先需要进行微信认证，通过后就可以进行跳转网页了，跳转网页的地址要求必须有域名并且域名需要备案通过。

# 第二章. 需求分析和环境搭建

## 需求分析

用户在体检之前需要进行预约，可以通过电话方式进行预约，此时会由体检中心客服人员通过后台系统录入预约信息。用户也可以通过手机端自助预约。本章节开发的功能为用户通过手机自助预约。

预约流程如下：

1、访问移动端首页

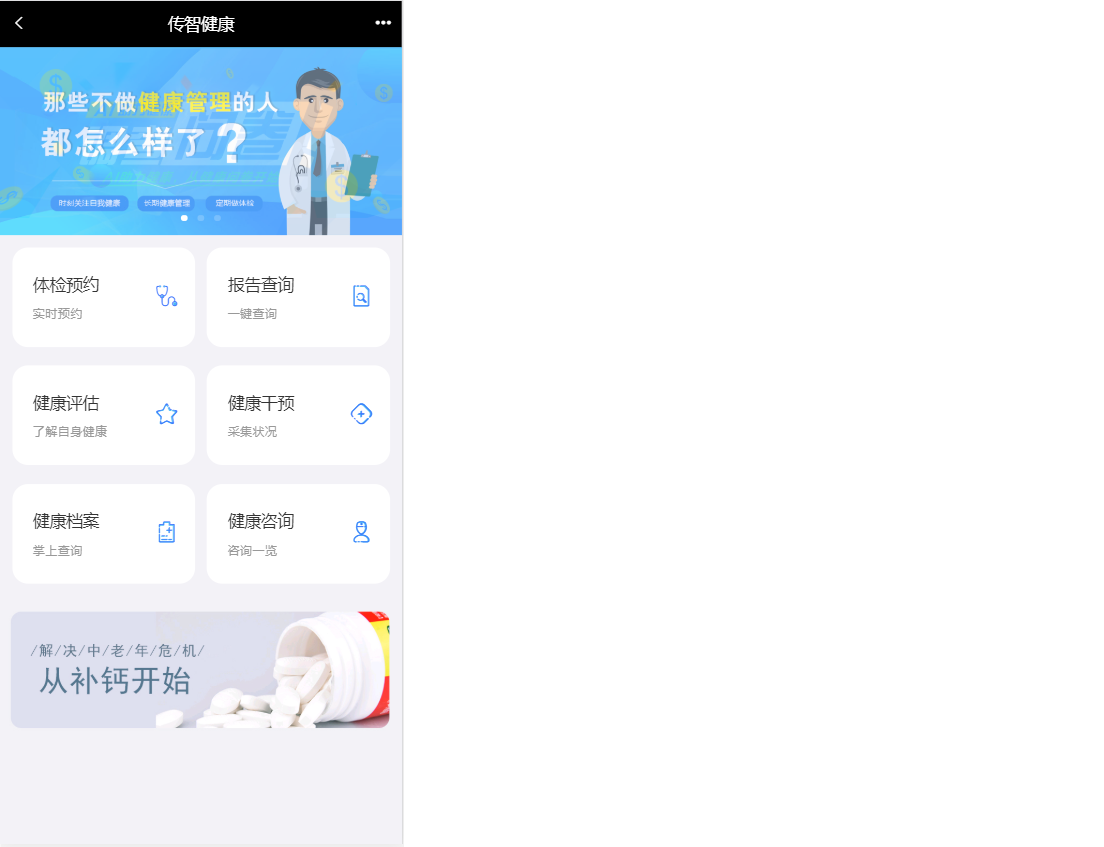
2、点击体检预约进入体检套餐列表页面

3、在体检套餐列表页面点击具体套餐进入套餐详情页面

4、在套餐详情页面点击立即预约进入预约页面

5、在预约页面录入体检人相关信息点击提交预约

效果如下图：



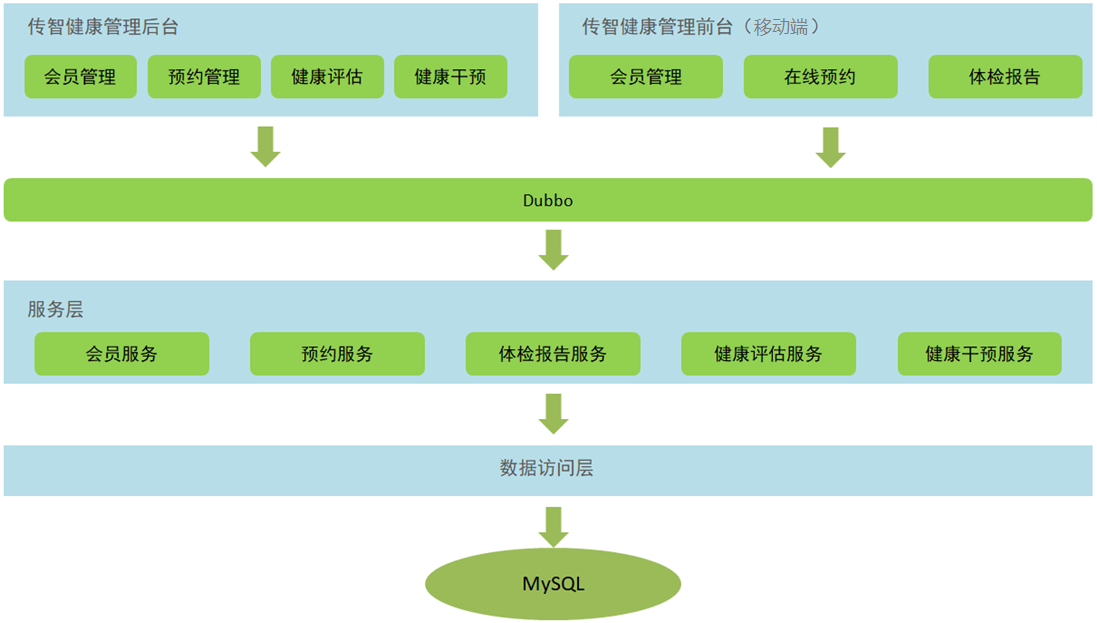






## 搭建移动端工程

本项目是基于SOA架构进行开发，前面我们已经完成了后台系统的部分功能开发，在后台系统中都是通过Dubbo调用服务层发布的服务进行相关的操作。本章节我们开发移动端工程也是同样的模式，所以我们也需要在移动端工程中通过Dubbo调用服务层发布的服务，如下图：



### 导入maven坐标

在health\_common工程的pom.xml文件中导入阿里短信发送的maven坐标

*<!--阿里云服务器短信平台-->*<**dependency**>  
 <**groupId**>com.aliyun</**groupId**>  
 <**artifactId**>aliyun-java-sdk-core</**artifactId**>  
 <**version**>3.3.1</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.aliyun</**groupId**>  
 <**artifactId**>aliyun-java-sdk-dysmsapi</**artifactId**>  
 <**version**>1.0.0</**version**>  
</**dependency**>

### 导入通用组件

在health\_common工程中导入如下通用组件

1：ValidateCodeUtils工具类：（产生验证码）

**package** com.itheima.health.utils;  
  
**import** java.util.Random;  
  
*/\*\*  
 \* 随机生成验证码工具类  
 \*/***public class** ValidateCodeUtils {  
 */\*\*  
 \* 随机生成验证码  
 \** ***@param length*** *长度为4位或者6位  
 \** ***@return*** *\*/* **public static** Integer generateValidateCode(**int** length){  
 Integer code =**null**;  
 **if**(length == 4){  
 code = **new** Random().nextInt(9999);*//生成随机数，最大为9999* **if**(code < 1000){  
 code = code + 1000;*//保证随机数为4位数字* }  
 }**else if**(length == 6){  
 code = **new** Random().nextInt(999999);*//生成随机数，最大为999999* **if**(code < 100000){  
 code = code + 100000;*//保证随机数为6位数字* }  
 }**else**{  
 **throw new** RuntimeException(**"只能生成4位或6位数字验证码"**);  
 }  
 **return** code;  
 }  
  
 */\*\*  
 \* 随机生成指定长度字符串验证码  
 \** ***@param length*** *长度  
 \** ***@return*** *\*/* **public static** String generateValidateCode4String(**int** length){  
 Random rdm = **new** Random();  
 String hash1 = Integer.*toHexString*(rdm.nextInt());  
 String capstr = hash1.substring(0, length);  
 **return** capstr;  
 }  
}

2：SMSUtils工具类：（短信服务，用于发送短消息服务（SMS））

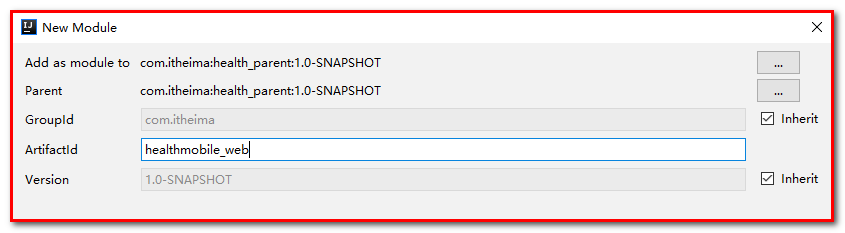
**package** com.itheima.health.utils;  
  
  
**import** com.aliyuncs.DefaultAcsClient;  
**import** com.aliyuncs.IAcsClient;  
**import** com.aliyuncs.dysmsapi.model.v20170525.SendSmsRequest;  
**import** com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;  
**import** com.aliyuncs.exceptions.ClientException;  
**import** com.aliyuncs.http.MethodType;  
**import** com.aliyuncs.profile.DefaultProfile;  
**import** com.aliyuncs.profile.IClientProfile;  
  
*/\*\*  
 \* 短信发送工具类  
 \*/***public class** SMSUtils {  
 **public static final** String ***VALIDATE\_CODE*** = **"SMS\_159620392"**;*//发送短信验证码* **public static final** String ***ORDER\_NOTICE*** = **"SMS\_159771588"**;*//体检预约成功通知  
  
 /\*\*  
 \* 发送短信  
 \** ***@param phoneNumbers*** *\** ***@param param*** *\** ***@throws*** *ClientException  
 \*/* **public static void** sendShortMessage(String templateCode,String phoneNumbers,String param) **throws** ClientException{  
 *// 设置超时时间-可自行调整* System.*setProperty*(**"sun.net.client.defaultConnectTimeout"**, **"10000"**);  
 System.*setProperty*(**"sun.net.client.defaultReadTimeout"**, **"10000"**);  
 *// 初始化ascClient需要的几个参数* **final** String product = **"Dysmsapi"**;*// 短信API产品名称（短信产品名固定，无需修改）* **final** String domain = **"dysmsapi.aliyuncs.com"**;*// 短信API产品域名（接口地址固定，无需修改）  
 // 替换成你的AK* **final** String accessKeyId = **"accessKeyId"**;*// 你的accessKeyId,参考本文档步骤2* **final** String accessKeySecret = **"accessKeySecret"**;*// 你的accessKeySecret，参考本文档步骤2  
 // 初始化ascClient,暂时不支持多region（请勿修改）* IClientProfile profile = DefaultProfile.*getProfile*(**"cn-hangzhou"**, accessKeyId, accessKeySecret);  
 DefaultProfile.*addEndpoint*(**"cn-hangzhou"**, **"cn-hangzhou"**, product, domain);  
 IAcsClient acsClient = **new** DefaultAcsClient(profile);  
 *// 组装请求对象* SendSmsRequest request = **new** SendSmsRequest();  
 *// 使用post提交* request.setMethod(MethodType.***POST***);  
 *// 必填:待发送手机号。支持以逗号分隔的形式进行批量调用，批量上限为1000个手机号码,批量调用相对于单条调用及时性稍有延迟,验证码类型的短信推荐使用单条调用的方式* request.setPhoneNumbers(phoneNumbers);  
 *// 必填:短信签名-可在短信控制台中找到* request.setSignName(**"传智健康"**);  
 *// 必填:短信模板-可在短信控制台中找到* request.setTemplateCode(templateCode);  
 *// 可选:模板中的变量替换JSON串,如模板内容为"亲爱的${name},您的验证码为${code}"时,此处的值为  
 // 友情提示:如果JSON中需要带换行符,请参照标准的JSON协议对换行符的要求,比如短信内容中包含\r\n的情况在JSON中需要表示成\\r\\n,否则会导致JSON在服务端解析失败* request.setTemplateParam(**"{\"code\":\""**+param+**"\"}"**);  
 *// 可选-上行短信扩展码(扩展码字段控制在7位或以下，无特殊需求用户请忽略此字段)  
 // request.setSmsUpExtendCode("90997");  
 // 可选:outId为提供给业务方扩展字段,最终在短信回执消息中将此值带回给调用者  
 // request.setOutId("yourOutId");  
 // 请求失败这里会抛ClientException异常* SendSmsResponse sendSmsResponse = acsClient.getAcsResponse(request);  
 **if** (sendSmsResponse.getCode() != **null** && sendSmsResponse.getCode().equals(**"OK"**)) {  
 *// 请求成功* System.***out***.println(**"请求成功"**);  
 }  
 }  
}

3：RedisMessageConstant常量类：

**package** com.itheima.health.constant;  
  
**public class** RedisMessageConstant {  
 **public static final** String ***SENDTYPE\_ORDER*** = **"001"**;*//用于缓存体检预约时发送的验证码* **public static final** String ***SENDTYPE\_LOGIN*** = **"002"**;*//用于缓存手机号快速登录时发送的验证码* **public static final** String ***SENDTYPE\_GETPWD*** = **"003"**;*//用于缓存找回密码时发送的验证码*}

### healthmobile\_web

移动端工程，打包方式为war，用于存放Controller，在Controller中通过Dubbo可以远程访问服务层相关服务，所以需要依赖health\_interface接口工程。

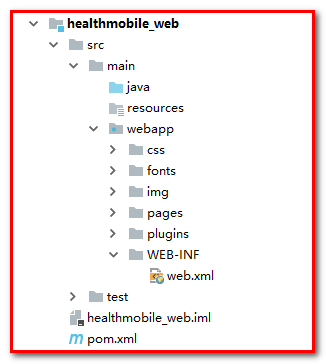


pom.xml：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**parent**>  
 <**artifactId**>health\_parent</**artifactId**>  
 <**groupId**>com.itheima</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**parent**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**artifactId**>healthmobile\_web</**artifactId**>  
 <**packaging**>war</**packaging**>  
  
 <**name**>healthmobile\_web Maven Webapp</**name**>  
 <**url**>http://www.example.com</**url**>  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**maven.compiler.source**>1.8</**maven.compiler.source**>  
 <**maven.compiler.target**>1.8</**maven.compiler.target**>  
 </**properties**>  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>com.itheima</**groupId**>  
 <**artifactId**>health\_interface</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
 </**dependencies**>  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.apache.tomcat.maven</**groupId**>  
 <**artifactId**>tomcat7-maven-plugin</**artifactId**>  
 <**configuration**>  
 *<!-- 指定端口，给客户端使用的端口，尽量用80端口，因为80端口是默认端口 -->* <**port**>80</**port**>  
 *<!-- 请求路径 -->* <**path**>/</**path**>  
 </**configuration**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
</**project**>

静态资源（CSS、html、img等，详见资料）：





web.xml：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns="http://java.sun.com/xml/ns/javaee"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 id="WebApp\_ID" version="3.0"**>  
 <**display-name**>Archetype Created Web Application</**display-name**>  
 *<!-- 解决post乱码 -->* <**filter**>  
 <**filter-name**>CharacterEncodingFilter</**filter-name**>  
 <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>  
 <**init-param**>  
 <**param-name**>encoding</**param-name**>  
 <**param-value**>utf-8</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>forceEncoding</**param-name**>  
 <**param-value**>true</**param-value**>  
 </**init-param**>  
 </**filter**>  
 <**filter-mapping**>  
 <**filter-name**>CharacterEncodingFilter</**filter-name**>  
 <**url-pattern**>/\*</**url-pattern**>  
 </**filter-mapping**>  
 <**servlet**>  
 <**servlet-name**>springmvc</**servlet-name**>  
 <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  
 *<!-- 指定加载的配置文件 ，通过参数contextConfigLocation加载 -->* <**init-param**>  
 <**param-name**>contextConfigLocation</**param-name**>  
 <**param-value**>classpath:springmvc.xml</**param-value**>  
 </**init-param**>  
 <**load-on-startup**>1</**load-on-startup**>  
 </**servlet**>  
 <**servlet-mapping**>  
 <**servlet-name**>springmvc</**servlet-name**>  
 <**url-pattern**>\*.do</**url-pattern**>  
 </**servlet-mapping**>  
 <**welcome-file-list**>  
 <**welcome-file**>/pages/index.html</**welcome-file**>  
 </**welcome-file-list**>  
</**web-app**>

springmvc.xml：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc.xsd  
 http://code.alibabatech.com/schema/dubbo  
 http://code.alibabatech.com/schema/dubbo/dubbo.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd"**>  
  
 <**mvc:annotation-driven**>  
 <**mvc:message-converters register-defaults="true"**>  
 <**bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"**>  
 <**property name="supportedMediaTypes" value="application/json"**/>  
 <**property name="features"**>  
 <**list**>  
 <**value**>WriteMapNullValue</**value**>  
 <**value**>WriteDateUseDateFormat</**value**>  
 </**list**>  
 </**property**>  
 </**bean**>  
 </**mvc:message-converters**>  
 </**mvc:annotation-driven**>  
 *<!-- 指定应用名称 -->* <**dubbo:application name="healthmobile\_web"** />  
 *<!--指定服务注册中心地址-->* <**dubbo:registry address="zookeeper://127.0.0.1:2181"**/>  
 *<!--批量扫描-->* <**dubbo:annotation package="com.itheima.health.controller"** />  
 *<!--  
 超时全局设置 10分钟  
 check=false 不检查服务提供方，开发阶段建议设置为false  
 check=true 启动时检查服务提供方，如果服务提供方没有启动则报错  
 -->* <**dubbo:consumer timeout="600000" check="false"**/>  
 <**import resource="spring-jedis.xml"**></**import**>  
</**beans**>

spring-jedis.xml：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc.xsd  
 http://code.alibabatech.com/schema/dubbo  
 http://code.alibabatech.com/schema/dubbo/dubbo.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd"**>  
  
 <**context:property-placeholder location="classpath:redis.properties"** />  
  
 *<!--Jedis连接池的相关配置-->* <**bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig"**>  
 <**property name="maxTotal"**>  
 <**value**>${redis.pool.maxActive}</**value**>  
 </**property**>  
 <**property name="maxIdle"**>  
 <**value**>${redis.pool.maxIdle}</**value**>  
 </**property**>  
 <**property name="testOnBorrow" value="true"**/>  
 <**property name="testOnReturn" value="true"**/>  
 </**bean**>  
  
 <**bean id="jedisPool" class="redis.clients.jedis.JedisPool"**>  
 <**constructor-arg name="poolConfig" ref="jedisPoolConfig"** />  
 <**constructor-arg name="host" value="${redis.host}"** />  
 <**constructor-arg name="port" value="${redis.port}" type="int"** />  
 <**constructor-arg name="timeout" value="${redis.timeout}" type="int"** />  
 </**bean**>  
</**beans**>

redis.properties：

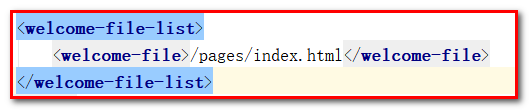
*#最大分配的对象数***redis.pool.maxActive**=**200***#最大能够保持idel状态的对象数***redis.pool.maxIdle**=**50  
redis.pool.minIdle**=**10  
redis.pool.maxWaitMillis**=**20000***#当池内没有返回对象时，最大等待时间***redis.pool.maxWait**=**300***#格式：redis://:[密码]@[服务器地址]:[端口]/[db index]  
#redis.uri = redis://:12345@127.0.0.1:6379/0***redis.host** = **127.0.0.1  
redis.port** = **6379  
redis.timeout** = **30000**

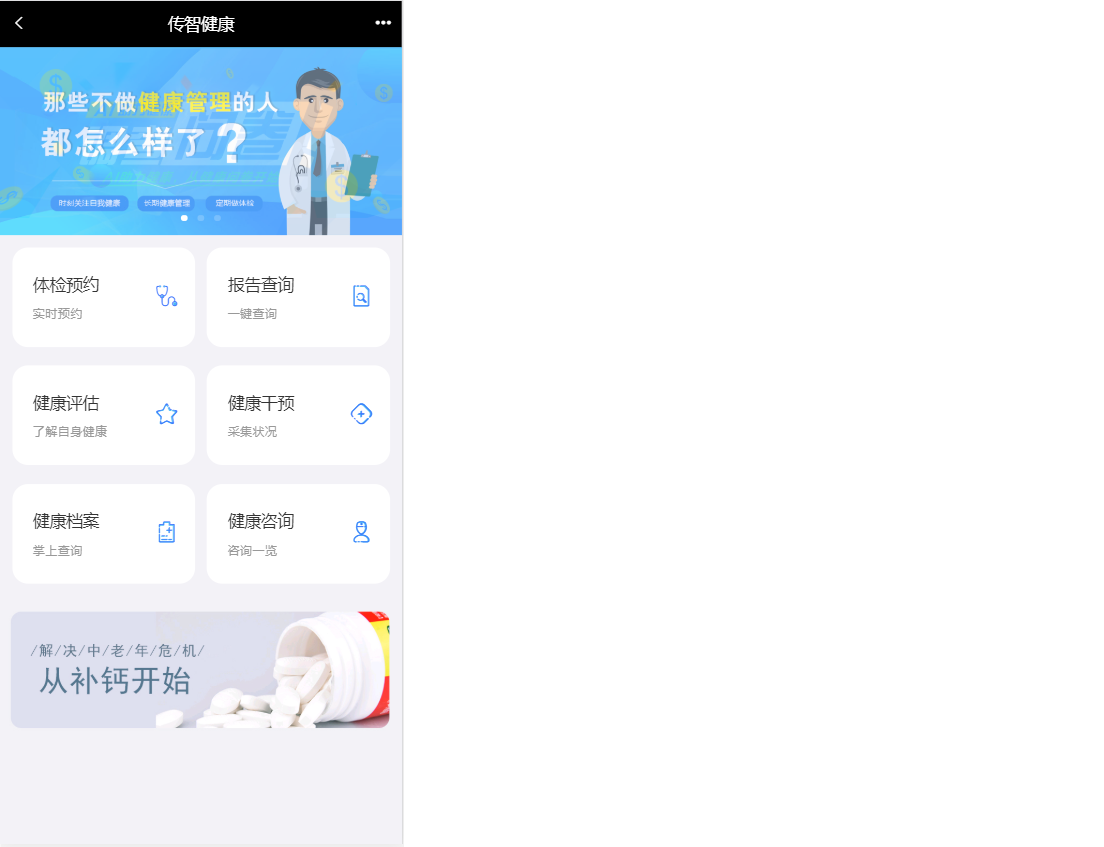
log4j.properties：

*### direct log messages to stdout ###***log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target**=**System.err  
log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern**=**%d{ABSOLUTE} %5p %c{1}:%L - %m%n***### direct messages to file mylog.log ###***log4j.appender.file**=**org.apache.log4j.FileAppender  
log4j.appender.file.File**=**c:\\mylog.log  
log4j.appender.file.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern**=**%d{ABSOLUTE} %5p %c{1}:%L - %m%n***### set log levels - for more verbose logging change 'info' to 'debug' ###***log4j.rootLogger**=**info, stdout**

# 第三章. 套餐列表页面动态展示

移动端首页为/pages/index.html，效果如下：





点击体检预约直接跳转到体检套餐列表页面（/pages/setmeal.html）

<**a href="/pages/setmeal.html" class="link-page"**>  
 <**div class="type-title"**>  
 <**h3**>体检预约</**h3**>  
 <**p**>实时预约</**p**>  
 </**div**>  
 <**div class="type-icon"**>  
 <**i class="icon-zhen"**>  
 <**span class="path1"**></**span**><**span class="path2"**></**span**>  
 </**i**>  
 </**div**>  
</**a**>

## 完善页面

### 展示套餐信息

<**ul class="list"**>  
 <**li class="list-item" v-for="setmeal in setmealList"**>  
 <**a class="link-page" :href="'setmeal\_detail.html?id='+setmeal.id"**>  
 <**img class="img-object f-left" :src="'http://pqjroc654.bkt.clouddn.com/'+setmeal.img" alt=""**>  
 <**div class="item-body"**>  
 <**h4 class="ellipsis item-title"**>{{setmeal.name}}</**h4**>  
 <**p class="ellipsis-more item-desc"**>{{setmeal.remark}}</**p**>  
 <**p class="item-keywords"**>  
 <**span**>{{setmeal.sex == '0' ? '性别不限' : setmeal.sex == '1' ? '男':'女'}}</**span**>  
 <**span**>{{setmeal.age}}</**span**>  
 </**p**>  
 </**div**>  
 </**a**>  
 </**li**>  
</**ul**>

其中：img标签中的src需要指定七牛云的空间地址。

### 获取套餐列表数据

<**script**>  
 **var *vue*** = **new** *Vue*({  
 **el**:**'#app'**,  
 **data**:{  
 **setmealList**:[]  
 },  
 mounted (){  
 **axios**.post(**"/setmeal/getSetmeal.do"**).then((response)=>{  
 **if**(response.**data**.**flag**){  
 **this**.**setmealList** = response.**data**.**data**;  
 }  
 });  
 }  
 });  
</**script**>

## 后台代码

### Controller

在healthmobile\_web工程中创建SetmealController并提供getSetmeal方法，在此方法中通过Dubbo远程调用套餐服务获取套餐列表数据

**package** com.itheima.health.controller;  
  
**import** com.alibaba.dubbo.config.annotation.Reference;  
**import** com.itheima.health.constant.MessageConstant;  
**import** com.itheima.health.entity.Result;  
**import** com.itheima.health.pojo.Setmeal;  
**import** com.itheima.health.service.SetmealService;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
  
**import** java.util.List;  
  
@RestController  
@RequestMapping(**"/setmeal"**)  
**public class** SetmealController {  
  
 @Reference*//(check = false)* **private** SetmealService **setmealService**;  
  
 *//获取所有套餐信息* @RequestMapping(**"/getSetmeal"**)  
 **public** Result getSetmeal(){  
 **try**{  
 List<Setmeal> list = **setmealService**.findAll();  
 **return new** Result(**true**, MessageConstant.***GET\_SETMEAL\_LIST\_SUCCESS***,list);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 **return new** Result(**true**,MessageConstant.***GET\_SETMEAL\_LIST\_FAIL***);  
 }  
 }  
}

### 服务接口

在SetmealService服务接口中扩展findAll方法

List<Setmeal> findAll();

### 服务实现类

在SetmealServiceImpl服务实现类中实现findAll方法

**public** List<Setmeal> findAll() {  
 **return setmealDao**.findAll();  
}

### Dao接口

在SetmealDao接口中扩展findAll方法

List<Setmeal> findAll();

### Mapper映射文件

在SetmealDao.xml映射文件中扩展SQL语句

*<!--查询所有-->*<**select id="findAll" resultType="com.itheima.health.pojo.Setmeal"**>  
 select *\** from t\_setmeal  
</**select**>

查看效果



# 第四章. 套餐详情页面动态展示

前面我们已经完成了体检套餐列表页面动态展示，点击其中任意一个套餐则跳转到对应的套餐详情页面（/pages/setmeal\_detail.html），并且会携带此套餐的id作为参数提交。



请求路径格式：<http://localhost/pages/setmeal_detail.html?id=10>

在套餐详情页面需要展示当前套餐的信息（包括图片、套餐名称、套餐介绍、适用性别、适用年龄）、此套餐包含的检查组信息、检查组包含的检查项信息等。

## 完善页面

### 获取请求参数中套餐id

在页面中已经引入了healthmobile.js文件，此文件中已经封装了getUrlParam方法可以根据URL请求路径中的参数名获取对应的值

*//获取指定的URL参数值 http://localhost/pages/setmeal\_detail.html?id=3&name=jack***function** *getUrlParam*(paraName) {  
 **var** url = **document**.**location**.toString();  
 *//alert(url);* **var** arrObj = url.split(**"?"**);  
 **if** (arrObj.**length** > 1) {  
 **var** arrPara = arrObj[1].split(**"&"**);  
 **var** arr;  
 **for** (**var** i = 0; i < arrPara.**length**; i++) {  
 arr = arrPara[i].split(**"="**);  
 **if** (arr != **null** && arr[0] == paraName) {  
 **return** arr[1];  
 }  
 }  
 **return ""**;  
 }  
 **else** {  
 **return ""**;  
 }  
}

在setmeal\_detail.html中调用上面定义的方法获取套餐id的值

<**script src="../plugins/healthmobile.js"**></**script**>  
<**script**>  
 **var *id*** = *getUrlParam*(**"id"**);

*//alert(id);*   
</**script**>

### 获取套餐详细信息

<**script**>  
 **var *vue*** = **new** *Vue*({  
 **el**:**'#app'**,  
 **data**:{  
 **imgUrl**:**null**,*//套餐对应的图片链接* **setmeal**:{}  
 },  
 **methods**:{  
 toOrderInfo(){  
 **window**.**location**.**href** = **"orderInfo.html?id="** + ***id***;  
 }  
 },  
 mounted(){

*// 发送ajax请求，根据套餐id查询套餐详情（套餐基本信息、套餐关联的多个检查组信息、检查组关联的多个检查项信息）*  
 **axios**.post(**"/setmeal/findById.do?id="** + ***id***).then((response) => {  
 **if**(response.**data**.**flag**){  
 **this**.**setmeal** = response.**data**.**data**;  
 **this**.**imgUrl** = **'http://pqjroc654.bkt.clouddn.com/'** + **this**.**setmeal**.**img**;  
 }  
 });  
 }  
 });  
</**script**>

其中：imgUrl要指定犀牛云的空间地址

### 展示套餐信息

*<!-- 页面内容 -->*<**div class="contentBox"**>  
 <**div class="card"**>  
 <**div class="project-img"**>  
 <**img :src="imgUrl" 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.sex == '0' ? '性别不限' : setmeal.sex == '1' ? '男':'女'}}</**span**>  
 <**span**>{{setmeal.age}}</**span**>  
 </**p**>  
 </**div**></**div**>  
 <**div class="table-listbox"**>  
 <**div class="box-title"**>  
 <**i class="icon-zhen"**><**span class="path1"**></**span**><**span class="path2"**></**span**></**i**>  
 <**span**>套餐详情</**span**>  
 </**div**>  
 <**div class="box-table"**>  
 <**div class="table-title"**>  
 <**div class="tit-item flex2"**>项目名称</**div**>  
 <**div class="tit-item flex3"**>项目内容</**div**>  
 <**div class="tit-item flex3"**>项目解读</**div**>  
 </**div**>  
 <**div class="table-content"**>  
 <**ul class="table-list"**>  
 <**li class="table-item" v-for="checkgroup in setmeal.checkGroups"**>  
 <**div class="item flex2"**>{{checkgroup.name}}</**div**>  
 <**div class="item flex3"**>  
 <**label v-for="checkitem in checkgroup.checkItems"**>  
 {{checkitem.name}}  
 </**label**>  
 </**div**>  
 <**div class="item flex3"**>{{checkgroup.remark}}</**div**>  
 </**li**>  
 </**ul**>  
 </**div**>  
 <**div class="box-button"**>  
 <**a @click="toOrderInfo()" class="order-btn"**>立即预约</**a**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

## 后台代码

### Controller

在SetmealController中提供findById方法

*//根据id查询套餐信息*@RequestMapping(**"/findById"**)  
**public** Result findById(**int** id){  
 **try**{  
 Setmeal setmeal = **setmealService**.findById(id);  
 **return new** Result(**true**,MessageConstant.***QUERY\_SETMEAL\_SUCCESS***,setmeal);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 **return new** Result(**false**,MessageConstant.***QUERY\_SETMEAL\_FAIL***);  
 }  
}

### 服务接口

在SetmealService服务接口中提供findById方法

Setmeal findById(**int** id);

### 服务实现类

在SetmealServiceImpl服务实现类中实现findById方法

**public** Setmeal findById(**int** id) {  
 **return setmealDao**.findById(id);  
}

### Dao接口

在SetmealDao接口中提供findById方法

Setmeal findById(**int** id);

在CheckGroupDao接口中提供findCheckGroupListById方法

List<CheckGroup> findCheckGroupListById(Integer id);

在CheckItemDao接口中提供findCheckItemListById方法

List<CheckItem> findCheckItemListById(Integer id);

### Mapper映射文件

此处会使用mybatis提供的关联查询，在根据id查询套餐时，同时将此套餐包含的检查组都查询出来，并且将检查组包含的检查项都查询出来。

SetmealDao.xml文件：

<**resultMap type="com.itheima.health.pojo.Setmeal" id="findByIdResultMap"**>  
 <**id column="id" property="id"**/>  
 <**result column="name" property="name"**/>  
 <**result column="code" property="code"**/>  
 <**result column="helpCode" property="helpCode"**/>  
 <**result column="sex" property="sex"**/>  
 <**result column="age" property="age"**/>  
 <**result column="price" property="price"**/>  
 <**result column="remark" property="remark"**/>  
 <**result column="attention" property="attention"**/>  
 <**result column="img" property="img"**/>  
 <**collection property="checkGroups" column="id"  
 select="com.itheima.health.dao.CheckGroupDao.findCheckGroupListById"**>  
 </**collection**>  
</**resultMap**>  
<**select id="findById" resultMap="findByIdResultMap"**>  
 select *\** from t\_setmeal where id=#{id}  
</**select**>

CheckGroupDao.xml文件：

<**resultMap type="com.itheima.health.pojo.CheckGroup" id="findByIdResultMap"**>  
 <**id column="id" property="id"**/>  
 <**result column="name" property="name"**/>  
 <**result column="code" property="code"**/>  
 <**result column="helpCode" property="helpCode"**/>  
 <**result column="sex" property="sex"**/>  
 <**result column="remark" property="remark"**/>  
 <**result column="attention" property="attention"**/>  
 <**collection property="checkItems" column="id"  
 select="com.itheima.health.dao.CheckItemDao.findCheckItemListById"**>  
 </**collection**>  
</**resultMap**>  
*<!--根据套餐id查询检查项信息-->*<**select id="findCheckGroupListById" resultMap="findByIdResultMap"**>  
 select *\** from t\_checkgroup where id  
 in (select checkgroup\_id from t\_setmeal\_checkgroup where setmeal\_id=#{id})  
</**select**>

或者使用sql：联合查询

SELECT c.\* FROM t\_checkgroup c,t\_setmeal\_checkgroup sc WHERE c.id = sc.checkgroup\_id AND sc.setmeal\_id = #{id}

CheckItemDao.xml文件：

*<!--根据检查组id查询检查项信息-->*<**select id="findCheckItemListById" resultType="com.itheima.health.pojo.CheckItem"**>  
 select *\** from t\_checkitem where id  
 in (select checkitem\_id from t\_checkgroup\_checkitem where checkgroup\_id=#{id})  
</**select**>

测试效果：

