

# 第5章 移动端开发-套餐列表、套餐详情、页面静态化

---

学习目标:

- 掌握移动端套餐列表页动态展示实现过程
- 掌握移动端套餐详情页
- 掌握Freemarker页面静态化技术
- 能够使用Freemarker生成html静态页面

## 1. 移动端需求分析和环境搭建

---

### 1.1. 需求分析

---

#### 【目标】

- 能够搭建移动端开发环境

#### 【路径】

- 开发需求
- 环境搭建

#### 【讲解】

#### 1.1.1 移动端开发需求分析

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：



#### 【小结】

体检预约-->套餐列表-->套餐详情-->立即预约（发送手机验证码+Redis）

## 1.2. 搭建移动端工程(互联用户)【重点】

### 【目标】

移动端工程搭建

### 【路径】

1. 创建health\_mobile工程, 导入坐标(依赖health\_interface)
2. 导入页面
3. 配置web.xml (springmvc的核心控制器+post请求乱码过滤器)
4. 创建springmvc.xml(配置dubbo, 驱动注解)
5. 导入通用组件

### 【讲解】

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，所以我們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：



### 1.2.1. 导入maven坐标

在health\_parent工程的pom.xml文件中导入阿里短信发送的maven坐标 依赖版本管理

```
1  <!-- 依赖版本管理标签-->
2  <dependencyManagement>
3      <dependencies>
4          <!--阿里云服务器短信平台-->
5          <dependency>
6              <groupId>com.aliyun</groupId>
7              <artifactId>aliyun-java-sdk-core</artifactId>
8              <version>3.3.1</version>
9          </dependency>
10         <dependency>
11             <groupId>com.aliyun</groupId>
12             <artifactId>aliyun-java-sdk-dysmsapi</artifactId>
13             <version>1.0.0</version>
14         </dependency>
15         ...
16     </dependencies>
17 </dependencyManagement>
```

在health\_common工程中添加引入的依赖

```

1 <dependency>
2   <groupId>com.aliyun</groupId>
3   <artifactId>aliyun-java-sdk-core</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>com.aliyun</groupId>
7   <artifactId>aliyun-java-sdk-dysmsapi</artifactId>
8 </dependency>

```

## 1.2.2. health\_mobile

### 【路径】

1: pom.xml

2: 静态资源 (CSS、html、img等)

3: web.xml

4: springmvc.xml

5: spring-jedis.xml

6: redis.properties

7: log4j.properties

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。



### 1.2.2.1. pom.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <parent>
6     <artifactId>health_parent</artifactId>
7     <groupId>com.itheima</groupId>
8     <version>1.0-SNAPSHOT</version>
9   </parent>
10  <modelVersion>4.0.0</modelVersion>
11
12  <artifactId>health_mobile</artifactId>
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    <maven.compiler.source>1.8</maven.compiler.source>
16    <maven.compiler.target>1.8</maven.compiler.target>
17  </properties>
18
19  <packaging>war</packaging>
20  <dependencies>
21    <dependency>
22      <groupId>com.itheima</groupId>
23      <artifactId>health_interface</artifactId>
24      <version>1.0-SNAPSHOT</version>

```

```

25     </dependency>
26 </dependencies>
27 </project>

```

### 1.2.2.2.静态资源 (CSS、html、img等, 详见资料)



### 1.2.2.3. web.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xmlns="http://java.sun.com/xml/ns/javaee"
4         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
5         id="WebApp_ID" version="3.0">
6      <display-name>Archetype Created Web Application</display-name>
7      <!-- 解决post乱码 -->
8      <filter>
9          <filter-name>CharacterEncodingFilter</filter-name>
10         <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
11         <init-param>
12             <param-name>encoding</param-name>
13             <param-value>utf-8</param-value>
14         </init-param>
15     </filter>
16     <filter-mapping>
17         <filter-name>CharacterEncodingFilter</filter-name>
18         <url-pattern>/*</url-pattern>
19     </filter-mapping>
20     <servlet>
21         <servlet-name>springmvc</servlet-name>
22         <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
23         <!-- 指定加载的配置文件，通过参数contextConfigLocation加载 -->
24         <init-param>
25             <param-name>contextConfigLocation</param-name>
26             <param-value>classpath:springmvc.xml</param-value>
27         </init-param>
28         <load-on-startup>1</load-on-startup>
29     </servlet>
30     <servlet-mapping>
31         <servlet-name>springmvc</servlet-name>
32         <url-pattern>*.do</url-pattern>
33     </servlet-mapping>
34 </web-app>

```

### 1.2.2.4. springmvc.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xmlns:mvc="http://www.springframework.org/schema/mvc"

```

```

4      xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
5      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">
6
7      <mvc:annotation-driven>
8          <mvc:message-converters register-defaults="true">
9              <bean
10 class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter">
11          <property name="supportedMediaTypes"
12 value="application/json"/>
13          <property name="features">
14              <list>
15                  <value>writeMapNullValue</value>
16                  <value>writeDateUseDateFormat</value>
17              </list>
18          </property>
19      </bean>
20  </mvc:message-converters>
21  </mvc:annotation-driven>
22  <!-- 指定应用名称 -->
23  <dubbo:application name="health_mobile" />
24  <!--指定服务注册中心地址-->
25  <dubbo:registry address="zookeeper://127.0.0.1:2181"/>
26  <!--批量扫描-->
27  <dubbo:annotation package="com.itheima.health.controller" />
28  <!--
29      超时全局设置 10分钟
30      check=false 不检查服务提供方，开发阶段建议设置为false
31      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
32  -->
33  <dubbo:consumer timeout="600000" check="false"/>
34  <import resource="classpath:spring-redis.xml"></import>
35 </beans>

```

### 1.2.2.5.spring-redis.xml:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
6
7     <context:property-placeholder location="classpath:redis.properties" />
8
9     <!--Jedis连接池的相关配置-->
10    <bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">
11        <property name="maxTotal">
12            <value>${redis.pool.maxActive}</value>
13        </property>
14        <property name="maxIdle">

```

```

15         <value>${redis.pool.maxIdle}</value>
16     </property>
17     <property name="testOnBorrow" value="true"/>
18     <property name="testOnReturn" value="true"/>
19 </bean>
20
21 <bean id="jedisPool" class="redis.clients.jedis.JedisPool">
22     <constructor-arg name="poolConfig" ref="jedisPoolConfig" />
23     <constructor-arg name="host" value="${redis.host}" />
24     <constructor-arg name="port" value="${redis.port}" type="int" />
25     <constructor-arg name="timeout" value="${redis.timeout}" type="int"
26 />
27 </bean>
</beans>

```

### 1.2.2.6. redis.properties

```

1  #最大分配的对象数
2  redis.pool.maxActive=200
3  #最大能够保持idle状态的对象数
4  redis.pool.maxIdle=50
5  redis.pool.minIdle=10
6  redis.pool.maxWaitMillis=20000
7  #当池内没有返回对象时，最大等待时间
8  redis.pool.maxwait=300
9
10 #格式: redis://:[密码]@[服务器地址]:[端口]/[db index]
11 #redis.uri = redis://:12345@127.0.0.1:6379/0
12
13 redis.host = 127.0.0.1
14 redis.port = 6379
15 redis.timeout = 30000

```

### 1.2.2.7. log4j.properties

```

1  ### direct log messages to stdout ###
2  log4j.appender.stdout=org.apache.log4j.ConsoleAppender
3  log4j.appender.stdout.Target=System.err
4  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
5  log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L -
6  %m%n
7
8  ### direct messages to file mylog.log ###
9  log4j.appender.file=org.apache.log4j.FileAppender
10 log4j.appender.file.File=c:\\mylog.log
11 log4j.appender.file.layout=org.apache.log4j.PatternLayout
12 log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L -
13 %m%n
14
15 ### set log levels - for more verbose logging change 'info' to 'debug' ###
16 log4j.rootLogger=debug, stdout

```

## 1.2.3. 导入通用组件

【路径】

- 1: ValidateCodeUtils工具类：（产生验证码）
- 2: SMSUtils工具类：（短信服务，用于发送短消息服务（SMS））
- 3: RedisMessageConstant常量类：

#### 【讲解】

在health\_common工程中导入如下通用组件

- 1: ValidateCodeUtils工具类：（产生验证码）

```
1  package com.itheima.health.utils;
2
3  import java.util.Random;
4
5  /**
6   * 随机生成验证码工具类
7   */
8  public class ValidateCodeUtils {
9      /**
10       * 随机生成验证码
11       * @param length 长度为4位或者6位
12       * @return
13       */
14     public static Integer generateValidateCode(int length){
15         Integer code =null;
16         if(length == 4){
17             code = new Random().nextInt(9999);//生成随机数，最大为9999
18             if(code < 1000){
19                 code = code + 1000;//保证随机数为4位数字
20             }
21         }else if(length == 6){
22             code = new Random().nextInt(999999);//生成随机数，最大为999999
23             if(code < 100000){
24                 code = code + 100000;//保证随机数为6位数字
25             }
26         }else{
27             throw new RuntimeException("只能生成4位或6位数字验证码");
28         }
29         return code;
30     }
31
32     /**
33      * 随机生成指定长度字符串验证码
34      * @param length 长度
35      * @return
36      */
37     public static String generateValidateCode4String(int length){
38         Random rdm = new Random();
39         String hash1 = Integer.toHexString(rdm.nextInt());
40         String capstr = hash1.substring(0, length);
41         return capstr;
42     }
43 }
```

- 2: SMSUtils工具类：（短信服务，用于发送短消息服务（SMS））

```

1 package com.itheima.health.utils;
2
3 import com.aliyuncs.DefaultAcsClient;
4 import com.aliyuncs.IAcsClient;
5 import com.aliyuncs.dysmsapi.model.v20170525.SendSmsRequest;
6 import com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;
7 import com.aliyuncs.exceptions.ClientException;
8 import com.aliyuncs.http.MethodType;
9 import com.aliyuncs.profile.DefaultProfile;
10 import com.aliyuncs.profile.IClientProfile;
11
12 /**
13  * 短信发送工具类
14  */
15 public class SMSUtils {
16     public static final String VALIDATE_CODE = "SMS_189616640";//发送短信验证码
17     public static final String ORDER_NOTICE = "SMS_159771588";//体检预约成功通知
18     private static final String SIGN_NAME = "黑马程序员";// 短信的签名
19     private static final String PARAMETER_NAME="code";
20     private static final String ACCESS_KEY="LTAI4GERJj7v71F3FKjw3z2A"; //你的AccessKey ID
21     private static final String SECRET_KEY="dIVZnHGdUTYbqOKMlxZ7R7jXVcnPoz";
22     //你的AccessKey Secret
23
24     public static void main(String[] args) throws ClientException {
25         SMSUtils.sendShortMessage(VALIDATE_CODE,"13652431027","666666");
26     }
27
28     /**
29      * 发送短信
30      * @param phoneNumbers
31      * @param param
32      * @throws ClientException
33      */
34     public static void sendShortMessage(String templateCode,String
35     phoneNumbers,String param) throws ClientException{
36         // 设置超时时间-可自行调整
37         System.setProperty("sun.net.client.defaultConnectTimeout", "10000");
38         System.setProperty("sun.net.client.defaultReadTimeout", "10000");
39         // 初始化ascClient需要的几个参数
40         final String product = "Dysmsapi";// 短信API产品名称（短信产品名固定，无
41         需修改）
42         final String domain = "dysmsapi.aliyuncs.com";// 短信API产品域名（接口
43         地址固定，无需修改）
44         // 替换成你的AK
45         final String accessKeyId = "LTAIak3CfAehK7CE";// 你的accessKeyId,参考
46         本文档步骤2
47         final String accessKeySecret = "zsykwhTIFa48f8fFdu06GOKjHWHe14";//
48         你的accessKeySecret, 参考本文档步骤2
49         // 初始化ascClient,暂时不支持多region（请勿修改）
50         IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou",
51         ACCESS_KEY, SECRET_KEY);
52         DefaultProfile.addEndpoint("cn-hangzhou", "cn-hangzhou", product,
53         domain);
54         IAcsClient acsClient = new DefaultAcsClient(profile);
55         // 组装请求对象

```



```

48     SendSmsRequest request = new SendSmsRequest();
49     // 使用post提交
50     request.setMethod(MethodType.POST);
51     // 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
52     request.setPhoneNumbers(phoneNumbers);
53     // 必填:短信签名-可在短信控制台中找到
54     request.setSignName(SIGN_NAME);
55     // 必填:短信模板-可在短信控制台中找到
56     request.setTemplateCode(templateCode);
57     // 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
58     // 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
59     //request.setTemplateParam("{\"code\":\""+param+"\"}");
60     request.setTemplateParam(String.format(
    "{\"%s\":\"%s\"}", PARAMETER_NAME, param));
61     // 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)
62     // request.setSmsUpExtendCode("90997");
63     // 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
64     // request.setOutId("yourOutId");
65     // 请求失败这里会抛ClientException异常
66     SendSmsResponse sendSmsResponse = acsClient.getAcsResponse(request);
67     if (sendSmsResponse.getCode() != null &&
    sendSmsResponse.getCode().equals("OK")) {
68         // 请求成功
69         System.out.println("请求成功");
70     } else {
71         System.out.println(sendSmsResponse.getMessage());
72     }
73 }
74 }
75

```

3: RedisMessageConstant常量类:

```

1 package com.itheima.health.constant;
2
3 public interface RedisMessageConstant {
4     static final String SENDTYPE_ORDER = "001";//用于缓存体检预约时发送的验证码
5     static final String SENDTYPE_LOGIN = "002";//用于缓存手机号快速登录时发送的验
    证码
6     static final String SENDTYPE_GETPWD = "003";//用于缓存找回密码时发送的验证码
7 }

```

## 【小结】

步骤:

-->创建工程-->导入坐标-->页面-->配置文件 (web.xml、springmvc.xml、spring-jedis.xml)

-->导入通用组件 (生成验证码、发送短信工具类、Redis存储常量类)

## 2.套餐列表页面动态展示【重点】

### 【目标】

实现套餐列表功能

访问：首页<http://localhost:80>

点击“体检预约”，跳转到“套餐列表”页面



## 【路径】

1：前台代码编写

1. 在/pages/setmeal.html，在mounted()钩子函数里面

```
1 完成需求：
2 1. 使用axios请求服务器，获得数据 进行模型绑定
3 2. 数据遍历(v-for)
```

2：后台代码编写

1.创建类SetmealMobileController.java，添加查询所有的方法

拼接完整的图片路径

2. SetmealService与实现类添加查询所有的方法  
3. SetmealDao与映射文件 添加查询所有的方法

```
1 完成需求：
2 1: 查询所有的套餐
```

## 【讲解-需求】

移动端首页为/pages/index.html，效果如下：

(1) 在web.xml中配置：



(2) 访问：首页<http://localhost:80>，在index.html页面中点击体检预约直接跳转到体检套餐列表页面 (/pages/setmeal.html)



## 2.1. 前台代码

### 2.1.1. 展示套餐信息

(1) setmeal.html，遍历v-for="setmeal in setmealList"



### 2.1.2. 获取套餐列表数据

```
1 <script>
2   var vue = new Vue({
3     el: '#app',
```

```

4      data:{
5          setmealList:[] // 套餐列表数据
6      },
7      // 挂载后
8      mounted (){
9          axios.get("/setmeal/getSetmeal.do").then((res)=>{
10              if(res.data.flag){
11                  this.setmealList = res.data.data;
12              }else{
13                  // 失败的提示
14                  alert(res.data.message);
15              }
16          });
17      }
18  });
19 </script>

```

mounted钩子和created钩子

mounted钩子：页面被初始化后执行

created钩子：vue模型被初始化后执行

mounted钩子是在created钩子后面执行的。

## 2.2. 后台代码

### 2.2.1. Controller

在healthmobile\_web工程中创建SetmealMobileController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

```

1  package com.itheima.health.controller;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.itheima.health.constant.MessageConstant;
5  import com.itheima.health.entity.Result;
6  import com.itheima.health.pojo.Setmeal;
7  import com.itheima.health.service.SetmealService;
8  import com.itheima.health.utils.QiniuUtils;
9  import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RestController;
12
13 import java.util.List;
14
15 /**
16  * Description: No Description
17  * User: Eric
18  */
19 @RestController
20 @RequestMapping("/setmeal")
21 public class SetmealMobileController {
22
23     @Reference
24     private SetmealService setmealService;
25

```

```

26     /**
27      * 查询所有
28      */
29     @GetMapping("/getSetmeal")
30     public Result getSetmeal(){
31         // 查询所有的套餐
32         List<Setmeal> list = setmealService.findAll();
33         // 套餐里有图片有全路径吗? 拼接全路径
34         list.forEach(s->{
35             s.setImg(QiniuUtils.DOMAIN + s.getImg());
36         });
37         return new Result(true,
38             MessageConstant.GET_SETMEAL_LIST_SUCCESS,list);
39     }
40 }

```

### 2.2.2. 服务接口

在SetmealService服务接口中扩展findAll方法

```

1  /**
2   * 查询所有
3   * @return
4   */
5  List<Setmeal> findAll();

```

### 2.2.3. 服务实现类

在SetmealServiceImpl服务实现类中实现findAll方法

```

1  /**
2   * 查询所有的套餐
3   * @return
4   */
5  @Override
6  public List<Setmeal> findAll() {
7      return setmealDao.findAll();
8  }

```

### 2.2.4. Dao接口

在SetmealDao接口中扩展findAll方法

```

1  /**
2   * 查询所有的套餐
3   * @return
4   */
5  List<Setmeal> findAll();

```

### 2.2.5. Mapper映射文件

在SetmealDao.xml映射文件中扩展SQL语句

```
1 <!--查询所有-->
2 <select id="findAll" resultType="setmeal">
3     select * from t_setmeal
4 </select>
```

查看效果



## 【小结】

套餐列表功能需要展示发布的所有套餐，用于体检人选择指定套餐。

从数据查询出来的套餐信息的图片只有图片名称，没有全路径，在Controller返回之前，设置完整的路径

## 3. 套餐详情页面动态展示【重点】

### 【目标】

套餐详情页面动态展示



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。

### 【路径】

前台代码编写

1. 在/pages/setmeal\_detail.html

```
1 完成需求：
2  1. 获取请求参数中套餐id的值
3  2. 获取套餐详细信息
4  3. 展示套餐信息（套餐信息、检查组集合、检查项集合）
```

后台代码编写

1.类SetmealMobileController.java

2.类SetmealService.java

3.类SetmealServiceImpl.java

4.类SetmealDao.java

类CheckGroupDao

类CheckItemDao

5.配置文件SetmealDao.xml

配置文件CheckGroupDao.xml

配置文件CheckItemDao.xml

- 1 完成需求:
- 2 1: 根据套餐id查询对应的套餐信息
- 3 2: 查询每个套餐对应的检查组集合
- 4 3: 查询每个检查组对应的检查项集合

## 【讲解-需求】

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](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。



## 3.1. 前台代码

### 3.1.1. 获取请求参数中套餐id

(1) 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

```
1 //获取指定的URL参数值 http://localhost/pages/setmeal_detail.html?id=15
2 // 从url中获取参数的值
3 function getUrlParam(paraName) {
4     var url = document.location.toString();
5     //http://localhost/pages/setmeal_detail.html?id=15
6     //alert(url);
7     var arrObj = url.split("?");
8     // arrObj[0]=http://localhost/pages/setmeal_detail.html
9     // arrObj[1]=id=15
10    if (arrObj.length > 1) {
11        //id=15
12        var arrPara = arrObj[1].split("&");
13        var arr;
14        //arrPara[0] id=15
15        for (var i = 0; i < arrPara.length; i++) {
16            // id=15
17            arr = arrPara[i].split("=");
18            //arr[0]=id arr[1]=15
19            if (arr != null && arr[0] == paraName) {
20                return arr[1];
21            }
22        }
23        return "";
24    }
25    else {
26        return "";
27    }
28 }
```

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

### 3.1.2. 获取套餐详细信息

```
1 <script>
2   var vue = new Vue({
3     el: '#app',
4     data: {
5       setmeal: {}
6     },
7     methods: {
8       toOrderInfo() {
9         window.location.href = "orderInfo.html?id=" + id;
10      }
11    },
12    mounted() {
13      // 获取套餐详情信息
14      axios.get("/setmeal/findDetailById.do?id=" + id).then((res) => {
15        if(res.data.flag){
16          this.setmeal = res.data.data;
17        }else{
18          alert(res.data.message);
19        }
20      });
21    }
22  });
23 </script>
```

其中：imgUrl要指定七牛云的空间地址

### 3.1.3. 展示套餐信息

1: {{setmeal.name}}: 套餐信息

2: v-for="checkgroup in setmeal.checkGroups: 检查组信息

3: v-for="checkitem in checkgroup.checkItems": 检查项信息

套餐图片的展示

```
1 <!-- 页面内容 -->
2 <div class="contentBox">
3   <div class="card">
4     <div class="project-img">
5       
6     </div>
7     <div class="project-text">
8       <h4 class="tit">{{setmeal.name}}</h4>
9       <p class="subtit">{{setmeal.remark}}</p>
10      <p class="keywords">
11        <span>{{setmeal.sex == '0' ? '性别不限' : setmeal.sex == '1'
? '男': '女'}}</span>
```

```

12         <span>{{setmeal.age}}</span>
13     </p>
14 </div>
15 </div>
16 <div class="table-listbox">
17     <div class="box-title">
18         <i class="icon-zhen"><span class="path1"></span><span
class="path2"></span></i>
19         <span>套餐详情</span>
20     </div>
21     <div class="box-table">
22         <div class="table-title">
23             <div class="tit-item flex2">项目名称</div>
24             <div class="tit-item flex3">项目内容</div>
25             <div class="tit-item flex3">项目解读</div>
26         </div>
27         <div class="table-content">
28             <ul class="table-list">
29                 <li class="table-item" v-for="checkgroup in
setmeal.checkGroups">
30                     <div class="item flex2">{{checkgroup.name}}</div>
31                     <div class="item flex3">
32                         <label v-for="checkitem in
checkgroup.checkItems">
33                             {{checkitem.name}}
34                         </label>
35                     </div>
36                     <div class="item flex3">{{checkgroup.remark}}</div>
37                 </li>
38             </ul>
39         </div>
40         <div class="box-button">
41             <a @click="toOrderInfo()" class="order-btn">立即预约</a>
42         </div>
43     </div>
44 </div>
45 </div>

```

## 3.2. 后台代码

### 3.2.1. Controller

在SetmealMobileController中提供findById方法

```

1  /**
2   * 查询套餐详情
3   */
4  @GetMapping("/findDetailById")
5  public Result findDetailById(int id){
6      // 调用服务查询详情
7      Setmeal setmeal = setmealService.findDetailById(id);
8      // 设置图片的完整路径
9      setmeal.setImg(QiniuUtils.DOMAIN + setmeal.getImg());
10     return new Result(true, MessageConstant.QUERY_SETMEAL_SUCCESS, setmeal);
11 }

```



### 3.2.2. 服务接口

在SetmealService服务接口中提供findById方法

```
1  /**
2   * 查询套餐详情
3   * @param id
4   * @return
5   */
6  Setmeal findDetailById(int id);
```

### 3.2.3. 服务实现类

在SetmealServiceImpl服务实现类中实现findById方法

```
1  /**
2   * 查询套餐详情
3   * @param id
4   * @return
5   */
6  @Override
7  public Setmeal findDetailById(int id) {
8      return setmealDao.findDetailById(id);
9  }
```

### 3.2.4. Dao接口

1: 在SetmealDao接口中提供findById方法

```
1  /**
2   * 查询套餐详情
3   * @param id
4   * @return
5   */
6  Setmeal findDetailById(int id);
```

### 3.2.5. 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。

1: SetmealDao.xml文件:

```
1  <!-- 【注意】这里用的是resultMap，不要写错了resultType -->
2  <select id="findDetailById" parameterType="int"
3  resultMap="setmealDetailResultMap">
4      select s.id,s.name,s.age,s.sex,s.remark,s.img,
5             sc.checkgroup_id,g.name checkgroup_name,g.remark checkgroup_remark,
6             cc.checkitem_id,ci.name checkitem_name
7  From t_setmeal s
8  left join t_setmeal_checkgroup sc on s.id=sc.setmeal_id
9  left join t_checkgroup g on sc.checkgroup_id=g.id
10 left join t_checkgroup_checkitem cc on g.id=cc.checkgroup_id
```

```

10     left join t_checkitem ci on cc.checkitem_id=ci.id
11     where s.id=#{id}
12 </select>
13 <resultMap id="setmealDetailResultMap" type="Setmeal">
14     <id property="id" column="id"/>
15     <result property="name" column="name"/>
16     <result property="sex" column="sex"/>
17     <result property="age" column="age"/>
18     <result property="remark" column="remark"/>
19     <result property="img" column="img"/>
20     <!-- ofType 指定多方的类型，必须的 -->
21     <collection property="checkGroups" ofType="CheckGroup">
22         <id property="id" column="checkgroup_id"/>
23         <result property="name" column="checkgroup_name"/>
24         <result property="remark" column="checkgroup_remark"/>
25         <collection property="checkItems" ofType="CheckItem">
26             <id property="id" column="checkitem_id"/>
27             <result property="name" column="checkitem_name"/>
28         </collection>
29     </collection>
30 </resultMap>
31

```

测试效果：

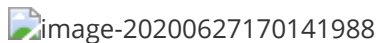


## 【小结】

- 1: 掌握sql语句的联合查询（sql语句的嵌套查询），应用在多对多场景
- 2: 掌握使用<resultMap>完成映射
- 3: 掌握使用<collection> 1对多 完成集合的封装（<association> 1对1 完成对象的封装）

## 套餐详情实现-方式二

1 页面修改



2 SetmealMobileController 添加方式二的方法

```

1  /**
2   * 查询套餐详情
3   */
4  @GetMapping("/findDetailById2")
5  public Result findDetailById2(int id){
6      // 调用服务查询详情
7      Setmeal setmeal = setmealService.findDetailById2(id);
8      // 设置图片的完整路径
9      setmeal.setImg(QiniuUtils.DOMAIN + setmeal.getImg());
10     return new Result(true, MessageConstant.QUERY_SETMEAL_SUCCESS, setmeal);
11 }

```

### 3 SetmealService接口与实现类

#### SetmealService添加方法

```
1  /**
2   * 查询套餐详情
3   * @param id
4   * @return
5   */
6  Setmeal findDetailById2(int id);
```

#### SetmealServiceImpl 添加实现

```
1  /**
2   * 查询套餐详情
3   * @param id
4   * @return
5   */
6  @Override
7  public Setmeal findDetailById2(int id) {
8      return setmealDao.findDetailById2(id);
9  }
```

### 3 SetmealDao与映射文件

#### SetmealDao添加方法

```
1  /**
2   * 查询套餐详情 方式二
3   * @param id
4   * @return
5   */
6  Setmeal findDetailById2(int id);
```

#### 映射文件

```
1  <!-- SetmealDao.xml -->
2  <select id="findDetailById2" resultMap="setmealResultMap">
3      select * from t_setmeal where id=#{id}
4  </select>
5
6  <resultMap type="Setmeal" id="setmealResultMap">
7      <id column="id" property="id"/>
8      <result column="name" property="name"/>
9      <result column="code" property="code"/>
10     <result column="helpCode" property="helpCode"/>
11     <result column="sex" property="sex"/>
12     <result column="age" property="age"/>
13     <result column="price" property="price"/>
14     <result column="remark" property="remark"/>
15     <result column="attention" property="attention"/>
16     <result column="img" property="img"/>
```


```

17     <collection property="checkGroups" column="id"
18
19     select="com.itheima.health.dao.CheckGroupDao.findCheckGroupListById">
20     </collection>
21 </resultMap>
22 <!-- CheckGroupDao.xml -->
23 <resultMap type="com.itheima.health.pojo.CheckGroup" id="findByIdResultMap">
24     <id column="id" property="id"/>
25     <result column="name" property="name"/>
26     <result column="code" property="code"/>
27     <result column="helpCode" property="helpCode"/>
28     <result column="sex" property="sex"/>
29     <result column="remark" property="remark"/>
30     <result column="attention" property="attention"/>
31     <collection property="checkItems" column="id"
32
33     select="com.itheima.health.dao.CheckItemDao.findCheckItemListById">
34     </collection>
35 </resultMap>
36 <!--根据套餐id查询检查项信息-->
37 <select id="findCheckGroupListById" resultMap="findByIdResultMap">
38     select * from t_checkgroup where id
39     in (select checkgroup_id from t_setmeal_checkgroup where setmeal_id=#
40 {id})
41 </select>
42 <!-- CheckItemDao.xml -->
43 <!--根据检查组id查询检查项信息-->
44 <select id="findCheckItemListById"
45 resultMap="com.itheima.health.pojo.CheckItem">
46     select * from t_checkitem where id
47     in (select checkitem_id from t_checkgroup_checkitem where
48 checkgroup_id=#{id})
49 </select>

```

## 套餐详情实现-方式三

### 1 页面修改

 image-20200627170800718

### 2 SetmealMobileController

```

1  /**
2   * 查询套餐详情
3   */
4  @GetMapping("/findDetailById3")
5  public Result findDetailById3(int id){
6      // 调用服务查询详情
7      Setmeal setmeal = setmealService.findDetailById3(id);
8      // 设置图片的完整路径
9      setmeal.setImg(QiniuUtils.DOMAIN + setmeal.getImg());
10     return new Result(true, MessageConstant.QUERY_SETMEAL_SUCCESS, setmeal);
11 }

```

### 3 SetmealService接口与实现类

#### SetmealService接口

```
1  /**
2   * 查询套餐详情 方式三
3   * @param id
4   * @return
5   */
6  Setmeal findDetailById3(int id);
```

#### SetmealServiceImpl实现类

```
1  /**
2   * 查询套餐详情 方式三
3   */
4  @Override
5  public Setmeal findDetailById3(int id) {
6      // 查询套餐信息
7      Setmeal setmeal = setmealDao.findById(id);
8      // 查询套餐下的检查组
9      List<CheckGroup> checkGroups =
10         setmealDao.findCheckGroupListBySetmealId(id);
11         if(null != checkGroups){
12             for (CheckGroup checkGroup : checkGroups) {
13                 // 通过检查组id检查检查项列表
14                 List<CheckItem> checkItems =
15                     setmealDao.findCheckItemByCheckGroupId(checkGroup.getId());
16                 // 设置这个检查组下所拥有的检查项
17                 checkGroup.setCheckItems(checkItems);
18             }
19             //设置套餐下的所拥有的检查组
20             setmeal.setCheckGroups(checkGroups);
21         }
22         return setmeal;
23     }
```

### 4 SetmealDao与映射文件

#### SetmealDao接口

```
1  /**
2   * 通过套餐id查询检查组列表
3   * @param id
4   * @return
5   */
6  List<CheckGroup> findCheckGroupListBySetmealId(Integer id);
7
8  /**
9   * 通过检查组id检查检查项列表
10   * @param id
11   * @return
12   */
13  List<CheckItem> findCheckItemByCheckGroupId(Integer id);
```

#### SetmealDao.xml映射文件

```
1 <select id="findCheckGroupListBySetmealId" resultType="Checkgroup"  
  parameterType="int">  
2     select * from t_checkgroup where id in (  
3         select checkgroup_id from t_setmeal_checkgroup where setmeal_id=#  
        {id}  
4     )  
5 </select>  
6 <select id="findCheckItemByCheckGroupId" resultType="checkitem"  
  parameterType="int">  
7     select * from t_checkitem where id in (  
8         select checkitem_id from t_checkgroup_checkitem where  
        checkgroup_id=#{id}  
9     )  
10 </select>
```

套餐详情方式实现结果：

推荐使用方式一，获取1个连接，查询一次就返回结果

方式二：获取1个连接，期间需要执行多次查询，查询次数取决于检查组的个数

方式三：每次查询都获取新的连接，查询的次数与方式二一样。

## 4. Freemarker 页面静态化技术

### 【目标】

- 1：什么是页面静态化技术（作用：用于减少查询数据库的频率, 利于SEO(搜索引擎优化) 搜索排名靠前)
- 2：什么是Freemarker
- 3：Freemarker的操作入门

### 【路径】

- 1：什么是页面静态化技术（作用：用于减少查询数据库的频率)
- 2：什么是Freemarker（作用：可生成html静态资源文件，从而达到减少查询数据库的频率)
- 3：Freemarker入门案例
  - (1) 搭建环境
  - (2) 创建模板文件
  - (3) 使用模板文件，生成静态文件
- 4：Freemarker相关指令
  - (1) assign指令
  - (2) include指令
  - (3) if指令
  - (4) list指令

### 【讲解】

## 4.1. 页面静态化介绍

本章课程中我们已经实现了移动端套餐列表页面和套餐详情页面的动态展示。但是我们需要思考一个问题，就是对于这两个页面来说，每次用户访问这两个页面都需要查询数据库获取动态数据进行展示，而且这两个页面的访问量是比较大的，这就对数据库造成了很大的访问压力，并且数据库中的数据变化频率并不高。那我们需要通过什么方法为数据库减压并提高系统运行性能呢？答案就是页面静态化。

页面静态化其实就是将原来的动态网页(例如通过ajax请求动态获取数据库中的数据并展示的网页)改为通过静态化技术生成的静态网页，这样用户在访问网页时，服务器直接给用户响应静态html页面，没有了动态查询数据库的过程。

那么这些静态HTML页面还需要我们自己去编写吗？其实并不需要，我们可以通过专门的页面静态化技术帮我们生成所需的静态HTML页面，例如：Freemarker、thymeleaf, velocity等。

页面静态化技术：把页面需要所有数据都写死文件里，下次使用时，不需要再去查询数据库，直接用即可

小结：

什么页面静态化：返回的页面就已经包含所有的内容，不需要请求后台查询数据

为什么使用页面静态化：减少与服务器交互次数，提升Seo，搜索排名靠前

## 4.2. Freemarker介绍

FreeMarker 是一个用 Java 语言编写的模板引擎，它基于模板来生成文本输出。FreeMarker与 Web 容器无关，即在 Web 运行时，它并不知道 Servlet 或 HTTP。它不仅可以用于表现层的实现技术，而且还可以用于生成 XML，JSP 或 Java 等。



学习网站：<http://freemarker.foofun.cn/>

Freemarker: 是一款文本模板引擎，运行后台代码。Velocity, Thymeleaf

作用：实现页面静态化，通过模板，来生成文件，往模板填写数据即可

## 4.3. Freemarker入门案例

### 4.3.1. 环境搭建

创建freemarkdemo工程并导入Freemarker的maven坐标

```
1 <dependency>
2   <groupId>org.freemarker</groupId>
3   <artifactId>freemarker</artifactId>
4   <version>2.3.23</version>
5 </dependency>
```

### 4.3.2. 创建模板文件

模板文件中有四种元素：

1、文本，直接输出的部分 2、注释，即<#--...-->格式不会输出 3、插值（Interpolation）：即\${..}部分，将使用数据模型中的部分替代输出 4、FTL指令：FreeMarker指令，和HTML标记类似，名字前加#予以区分，不会输出

Freemarker的模板文件后缀可以任意，一般建议为ftl。

在D盘创建ftl目录，在ftl目录中创建名称为test.ftl的模板文件，内容如下：

```
1 <html>
2 <head>
3     <meta charset="utf-8">
4     <title>Freemarker入门</title>
5 </head>
6 <body>
7     <#--我只是一个注释，我不会有任何输出 -->
8     ${name}你好，${message}
9 </body>
10 </html>
```

### 4.3.3. 生成文件

使用步骤：

第一步：创建一个 Configuration 对象，直接 new 一个对象。构造方法的参数就是 freemarker的版本号。

第二步：设置模板文件所在的路径。

第三步：设置模板文件使用的字符集。一般就是 utf-8。

第四步：加载一个模板，创建一个模板对象。

第五步：创建一个模板使用的数据集，可以是 pojo 也可以是 map。一般是 Map。

第六步：创建一个 Writer 对象，一般创建 FileWriter 对象，指定生成的文件名。

第七步：调用模板对象的 process 方法输出文件。

第八步：关闭流。

```
1 public static void main(String[] args) throws Exception{
2     //1.创建配置类
3     Configuration configuration=new
Configuration(Configuration.getVersion());
4     //2.设置模板所在的目录
5     configuration.setDirectoryForTemplateLoading(new File("D:\\ftl"));
6     //3.设置字符集
7     configuration.setDefaultEncoding("utf-8");
8     //4.加载模板
9     Template template = configuration.getTemplate("test.ftl");
10    //5.创建数据模型
11    Map map=new HashMap();
12    map.put("name", "张三");
13    map.put("message", "欢迎来到传智播客!");
14    //6.创建Writer对象
15    Writer out =new FileWriter(new File("d:\\test.html"));
16    //7.输出
17    template.process(map, out);
18    //8.关闭Writer对象
```



```
19 |     out.close();
20 | }
```

上面的入门案例中Configuration配置对象是自己创建的，字符集和模板文件所在目录也是在Java代码中指定的。在项目中应用时可以将Configuration对象的创建交由Spring框架来完成，并通过依赖注入方式将字符集和模板所在目录注入进去。

## 4.4. Freemarker指令

### 4.4.1. assign指令

assign指令用于在页面上定义一个变量

(1) 定义简单类型

```
1 | <#assign linkman="周先生">
2 | 联系人: ${linkman}
```

(2) 定义对象类型

```
1 | <#assign info={"mobile":"13812345678",'address':'北京市昌平区'} >
2 | 电话: ${info.mobile} 地址: ${info.address}
```

### 4.4.2. include指令

include指令用于模板文件的嵌套，用于页面布局, 页面公共内容的抽取

(1) 创建模板文件head.ftl

```
1 | <h1>黑马程序员</h1>
```

(2) 修改入门案例中的ftl.ftl，在ftl.ftl模板文件中使用include指令引入上面的模板文件

```
1 | <#include "head.ftl"/>
```

### 4.4.3. if指令

if指令用于判断

(1) 在模板文件中使用if指令进行判断

```
1 | <#if success=true>
2 |     你已通过实名认证
3 | <#else>
4 |     你未通过实名认证
5 | </#if>
```

(2) 在java代码中为success变量赋值

```
1 | map.put("success", true);
```

在freemarker的判断中，可以使用= 也可以使用

if指令可以配合assign success=true使用

```
1 <#assign success1=true>
2 <#if success1=true>
3     你已通过实名认证1
4 <#else>
5     你未通过实名认证1
6 </#if>
```

#### 4.4.4. list指令

list指令用于遍历

(1) 在模板文件中使用list指令进行遍历

```
1 <#list goodsList as goods>
2     商品名称: ${goods.name} 价格: ${goods.price}<br>
3 </#list>
```

(2) 在java代码中为goodsList赋值

```
1 List goodsList=new ArrayList();
2
3 Map goods1=new HashMap();
4 goods1.put("name", "苹果");
5 goods1.put("price", 5.8);
6
7 Map goods2=new HashMap();
8 goods2.put("name", "香蕉");
9 goods2.put("price", 2.5);
10
11 Map goods3=new HashMap();
12 goods3.put("name", "橘子");
13 goods3.put("price", 3.2);
14
15 goodsList.add(goods1);
16 goodsList.add(goods2);
17 goodsList.add(goods3);
18
19 map.put("goodsList", goodsList);
20
```

### 【小结】

<http://freemarker.foofun.cn/> 官方

1: 什么是页面静态化技术, 返回给浏览器时, 所有的数据都已经写好了, 不需要再访问服务去获取数据, 页面中的内容已经写死了。减少数据库访问、提高seo

2: 什么是Freemarker (作用: 可生成html静态资源文件, 从而达到减少查询数据库的频率)

是一种模板引擎, 通过往模板填充数据即生成文件。

3: Freemarker入门案例

- (1) 搭建环境 (maven, 引入依赖)
- (2) 创建模板文件(【utf-8】 【utf-8】 【utf-8】 另存为选择utf-8, ANSI->latin-1 -> ISO-8859-1)
- (3) 使用模板文件(加载文件, 配置模板文件目录, 设置默认的编码utf-8, 获取模板), 生成静态文件(数据模型map)

1. 通过版本来创建配置类
2. 设置模板路径
3. 设置编码
4. 通过文件名获取模板
5. 创建数据模型Map<String,Object> 实体类
6. process方法, 填充数据到输出文件里
7. 关闭流

#### 4: Freemarker相关指令

- (1) assign指令 声明变量且给它赋值, 如果数据模型中有相同的变量, 则以assign为准
- (2) include指令 导入其它模板文件, 页面布局, 公共内容的抽取
- (3) if指令 (常用) 判断输出
- (4) list指令 (常用) 遍历集合 list as item
5. 如果数据模型dataModel与assign使用同一变量时, 则以assign为准
6. 如果模板中使用的变量必须声明且不为null, 否则运行报错

## 5. 生成移动端静态页面, 整合项目

前面我们已经学习了Freemarker的基本使用方法, 下面我们就可以将Freemarker应用到项目中, 帮我们生成移动端套餐列表静态页面和套餐详情静态页面。接下来我们需要思考几个问题:

1. 什么时候生成静态页面比较合适呢? 套餐的增删改(列表, 修改的套餐详情)
2. 将静态页面生成到什么位置呢?

如果是在开发阶段可以将文件生成到项目工程中, 如果上线后可以将文件生成到移动端系统运行的nginx的html目录下。

3. 应该生成几个静态页面呢?

套餐列表.html, setmeal\_{id}.html

#### 4. 技术选型

- 使用redis的set集合记录每次变更了的套餐id, key为setmeal:static:html, value为id | 操作类型(0,1) | 时间戳
  - value值说明, id为要处理的套餐的id值
  - 操作类型0:代表着删除, 1: 代表着需要重新生成页面(添加/修改)
  - 时间戳: 保证每次对套餐的操作都能被处理到。防止用户在生成静态页面还未完成时的多次操作同一个套餐, 导致生成的静态页面与修改后的不致问题。
- 使用Quartz后台任务来生成静态页面

### 【目标】

- 1: 在传智健康前端系统中使用页面静态化Freemarker技术
- 2: 使用Freemarker生成 套餐列表静态化页面 列表页
- 3: 使用Freemarker生成 套餐详情静态化页面 多个, 以id区分

## 【路径】

### 1. 环境搭建

- 引入依赖

### 2. 创建模板文件

- mobile\_setmeal.ftl
- mobile\_setmeal\_detail.ftl
- 去掉所有的js代码，插值表达式改为freemaker, v-for #list

### 3. health\_job整合Freemarker

- freemarker.properties定义生成文件存入目录
- spring-freemarker.xml spring整合freemarker
  - 配置类
    - 模板目录
    - 编码
- spring-redis.xml 使用redis

### 4. 创建任务类，实现生成静态页面

- 获取redis中需要处理的套餐id
- 如果key(setmeal:static:html)不存在或数量<=0则不处理
- 遍历循环每个值，按|分割，分别获取套餐的id（下标为0），操作类型（下标为1）
- 操作类型=1，则生成详情页面
  - 通过id查询套餐详情, 设置完整的图片路径
  - 构建数据模型map, 把套餐详情放入map
  - 获取模板
  - 创建writer
  - 填充数据到模板
  - 关闭writer
- 操作类型=0，则删除已经生成的页面
- 删除redis中对应的值(srem)
- 循环结束后，再重新生成列表页面
  - 查询所有的套餐列表
  - 设置完整的图片路径
  - 生成静态页面

### 5. 配置任务

- spring-jobs.xml中导入spring-freemarker.xml 并配置触发任务
- 注解支持
- 调度线程池

### 6. health\_web套餐功能修改

- 引入redis
- 套餐controller的添加、更新、删除

### 7. health\_mobile页面连接修改

- /pages/index.html

## 【讲解】

## 5.1. 环境搭建

在health\_parent工程的pom文件中导入Freemarker的maven坐标

```
1 <properties>
2     ....
3     <freemarker.version>2.3.23</freemarker.version>
4 </properties>
```

```
1 版本管理
2 <dependencyManagement>
3     <dependencies>
4         <dependency>
5             <groupId>org.freemarker</groupId>
6             <artifactId>freemarker</artifactId>
7             <version>${freemarker.version}</version>
8         </dependency>
9         ....
10    </dependencies>
11 </dependencyManagement>
```

在health\_common工程的pom文件中导入Freemarker的maven坐标

```
1 <dependency>
2     <groupId>org.freemarker</groupId>
3     <artifactId>freemarker</artifactId>
4 </dependency>
```

## 5.2. 创建模板文件

在health\_jobs工程的resources下创建ftl目录，在ftl目录中创建模板文件mobile\_setmeal.ftl和mobile\_setmeal\_detail.ftl文件，前者是用于生成套餐列表页面的模板文件，后者是生成套餐详情页面的模板文件

(1) mobile\_setmeal.ftl

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后! -->
7     <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=0,user-scalable=no,minimal-ui">
8     <meta name="description" content="">
9     <meta name="author" content="">
10    <link rel="icon" href="../img/asset-favico.ico">
11    <title>预约</title>
12    <link rel="stylesheet" href="../css/page-health-order.css" />
13 </head>
14 <body data-spy="scroll" data-target="#myNavbar" data-offset="150">
15 <div class="app" id="app">
```

```

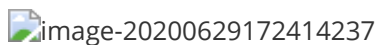
16      <!-- 页面头部 -->
17      <div class="top-header">
18          <span class="f-left"><i class="icon-back" onclick="history.go(-1)">
19      </i></span>
20          <span class="center">传智健康</span>
21          <span class="f-right"><i class="icon-more"></i></span>
22      </div>
23      <!-- 页面内容 -->
24      <div class="contentBox">
25          <div class="list-column1">
26              <ul class="list">
27                  <#list setmealList as setmeal>
28                      <li class="list-item" >
29                          <a class="link-page" href="setmeal_${setmeal.id}.html">
30                              
32                              <div class="item-body">
33                                  <h4 class="ellipsis item-title">${setmeal.name}</h4>
34                                  <p class="ellipsis-more item-desc">${setmeal.remark}
35                                  <p class="item-keywords">
36                                      <span>
37                                          <#if setmeal.sex=='0'>
38                                              性别不限
39                                          <#else>
40                                              <#if setmeal.sex=='1'>
41                                                  男
42                                              <#else>
43                                                  女
44                                              </#if>
45                                          </#if>
46                                      </span>
47                                      <span>${setmeal.age}</span>
48                                  </p>
49                              </div>
50                          </a>
51                      </li>
52                  </#list>
53              </ul>
54          </div>
55      </div>
56  </body>

```

注意1：上面的数据需保证数据库中不能为null。

注意2：套餐列表 的值里要设置图片的完整路径。

注意3：上面模板文件中每个套餐对应的超链接如下：



可以看到，链接的地址是动态构成的，

如果套餐的id为1，则对应的超链接地址为setmeal\_1.html；

如果套餐的id为5，则对应的超链接地址为setmeal\_5.html。

所以我们需要为每个套餐生成一个套餐详情静态页面。

(2) mobile\_setmeal\_detail.ftl

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <!-- 上述3个meta标签*必须*放在最前面,任何其他内容都*必须*跟随其后! -->
7     <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=0,user-scalable=no,minimal-ui">
8     <meta name="description" content="">
9     <meta name="author" content="">
10    <link rel="icon" href="../img/asset-favico.ico">
11    <title>预约详情</title>
12    <link rel="stylesheet" href="../css/page-health-orderDetail.css" />
13 </head>
14 <body data-spy="scroll" data-target="#myNavbar" data-offset="150">
15 <div id="app" class="app">
16     <!-- 页面头部 -->
17     <div class="top-header">
18         <span class="f-left"><i class="icon-back" onclick="history.go(-1)">
</i></span>
19         <span class="center">传智健康</span>
20         <span class="f-right"><i class="icon-more"></i></span>
21     </div>
22     <!-- 页面内容 -->
23     <div class="contentBox">
24         <div class="card">
25             <div class="project-img">
26                 
27             </div>
28             <div class="project-text">
29                 <h4 class="tit">${setmeal.name}</h4>
30                 <p class="subtit">${setmeal.remark}</p>
31                 <p class="keywords">
32                     <span>
33                         <#if setmeal.sex=='0'>
34                             性别不限
35                         <#else>
36                             <#if setmeal.sex=='1'>
37                                 男
38                             <#else>
39                                 女
40                             </#if>
41                         </#if>
42                     </span>
43                     <span>${setmeal.age}</span>
44                 </p>
45             </div>
46         </div>
47         <div class="table-listbox">
48             <div class="box-title">
49                 <i class="icon-zhen"><span class="path1"></span><span
class="path2"></span></i>
50                 <span>套餐详情</span>
51             </div>
52             <div class="box-table">
```

```

53         <div class="table-title">
54             <div class="tit-item flex2">项目名称</div>
55             <div class="tit-item flex3">项目内容</div>
56             <div class="tit-item flex3">项目解读</div>
57         </div>
58         <div class="table-content">
59             <ul class="table-list">
60                 <#list setmeal.checkGroups as checkgroup>
61                     <li class="table-item">
62                         <div class="item flex2">${checkgroup.name}</div>
63                         <div class="item flex3">
64                             <#list checkgroup.checkItems as checkitem>
65                                 <label>
66                                     ${checkitem.name}
67                                 </label>
68                             </#list>
69                         </div>
70                         <div class="item flex3">${checkgroup.remark}
71                     </li>
72                 </#list>
73             </ul>
74         </div>
75         <div class="box-button">
76             <a href="orderInfo.html?id=${setmeal.id}" class="order-
77 btn">立即预约</a>
78         </div>
79     </div>
80 </div>
81 </div>
82 </body>

```

## 5.3. 整合freemarker配置

(1) 在health\_jobs工程的resources中创建属性文件freemarker.properties。这个路径要改你的health\_mobile中的webapp/pages目录，全路径

```

1  out_put_path=D:/ideaProjects/health_parent/health_mobile/src/main/webapp/page
   s

```

通过上面的配置可以指定将静态HTML页面生成的目录位置

(2) 在health\_jobs工程的resources目录下，创建spring-freemarker.xml



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
6
7     <context:property-placeholder location="classpath:freemarker.properties"
ignore-unresolvable="true"/>
8
9     <bean id="freemarkerConfiguration"
class="freemarker.template.Configuration">
10         <constructor-arg index="0" ref="freemarkerVersion"/>
11     </bean>
12     <bean id="freemarkerVersion" class="freemarker.template.Configuration"
factory-method="getVersion"/>
13 </beans>

```

### (3) 创建spring-redis.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7     <!-- Jedis连接池的相关配置-->
8     <bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">
9         <!--    maxTotal 连接池中 最大的连接个数    -->
10         <property name="maxTotal">
11             <value>50</value>
12         </property>
13         <property name="maxIdle">
14             <value>10</value>
15         </property>
16         <!--    获取连接对象时，要测试一下是否能连上redis。能连接上则返回这个对象，不能连接上就获
17             取下个连接
18             保证从连接池中获取到的连接是可用的
19         -->
20         <property name="testOnBorrow" value="true"/>
21         <property name="testOnReturn" value="true"/>
22     </bean>
23
24     <bean id="jedisPool" class="redis.clients.jedis.JedisPool">
25         <constructor-arg name="poolConfig" ref="jedisPoolConfig" />
26         <constructor-arg name="host" value="127.0.0.1" />
27         <constructor-arg name="port" value="6379" type="int" />
28         <constructor-arg name="timeout" value="3000" type="int" />
29     </bean>
30 </beans>

```

## 5.4. 编写任务类

在health\_jobs工程中创建任务处理类，实现生成静态页面的逻辑。

```
1  package com.itheima.health.job;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.itheima.health.pojo.Setmeal;
5  import com.itheima.health.service.SetmealService;
6  import com.itheima.health.utils.QiNiuUtils;
7  import freemarker.cache.ClassTemplateLoader;
8  import freemarker.template.Configuration;
9  import freemarker.template.Template;
10 import org.slf4j.Logger;
11 import org.slf4j.LoggerFactory;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.beans.factory.annotation.Value;
14 import org.springframework.scheduling.annotation.Scheduled;
15 import org.springframework.stereotype.Component;
16 import redis.clients.jedis.Jedis;
17 import redis.clients.jedis.JedisPool;
18
19 import javax.annotation.PostConstruct;
20 import java.io.BufferedWriter;
21 import java.io.File;
22 import java.io.FileOutputStream;
23 import java.io.OutputStreamWriter;
24 import java.util.HashMap;
25 import java.util.List;
26 import java.util.Map;
27 import java.util.Set;
28
29 /**
30  * Description: No Description
31  * User: Eric
32  */
33 //@Component("generateHtmlJob")
34 public class GenerateHtmlJob {
35
36     /** 日志 */
37     private static final Logger log =
38         LoggerFactory.getLogger(GenerateHtmlJob.class);
39
40     /**
41      * spring创建对象后，调用的初始化方法
42      */
43     @PostConstruct
44     private void init(){
45         // 设置模板所在
46         configuration.setTemplateLoader(new
47             ClassTemplateLoader(this.getClass(), "/ftl"));
48         // 指定默认编码
49         configuration.setDefaultEncoding("utf-8");
50     }
51
52     /** jedis连接池 */
53     @Autowired
```

```

52     private JedisPool jedisPool;
53
54     /** 订阅套餐服务 */
55     @Reference
56     private SetmealService setmealService;
57
58     /** 注入freemarker主配置类 */
59     @Autowired
60     private Configuration configuration;
61
62     /** 生成静态页面存放的目录 */
63     @Value("${out_put_path}")
64     private String out_put_path;
65
66     /**
67      * 任务执行的方法
68      */
69     @Scheduled(initialDelay = 3000, fixedDelay = 1800000)
70     public void doGenerateHtml() {
71         // 获取redis连接对象
72         Jedis jedis = jedisPool.getResource();
73         // redis中set集合的key
74         String key = "static:setmeal:html";
75         // 获取集合中所有的套餐id数据
76         Set<String> setmealIds = jedis.smembers(key);
77         if (null != setmealIds && setmealIds.size() > 0) {
78             // 有数据则需要处理
79             for (String setmealId : setmealIds) {
80                 String[] ss = setmealId.split("\\|");
81                 // 获取套餐的id
82                 int id = Integer.valueOf(ss[0]);
83                 // 获取操作类型
84                 String operation = ss[1];
85                 // 需要生成套餐详情页面的操作
86                 if ("1".equals(operation)) {
87                     // 查询套餐详情
88                     Setmeal setmealDetail =
setmealService.findDetailById(id);
89                     // 设置图片的全路径
90                     setmealDetail.setImg(QiniuUtils.DOMAIN +
setmealDetail.getImg());
91                     // 生成套餐详情静态页面
92                     generateSetmealDetailHtml(setmealDetail);
93                 } else {
94                     // 删除套餐静态页面
95                     removeSetmealDetailFile(id);
96                 }
97                 // 每处理完一个，删除set集合中对应的数据
98                 jedis.srem(key, setmealId);
99             }
100             // 套餐列表的数据也发生了变化，要重新生成静态页面
101             generateSetmealList();
102         }
103     }
104
105     /**
106      * 删除（被删除的套餐）静态页面
107      * @param id 套餐id

```

```

108     */
109     private void removeSetmealDetailFile(int id){
110         File file = new
File(out_put_path,String.format("setmeal_%d.html",id));
111         if(file.exists()){
112             // 如果文件存在，则删除
113             file.delete();
114         }
115     }
116
117     /**
118     * 生成套餐详情页面
119     * @param setmeal
120     */
121     private void generateSetmealDetailHtml(Setmeal setmeal){
122         // 构建数据模型
123         Map<String,Object> dataMap = new HashMap<String,Object>();
124         dataMap.put("setmeal", setmeal);
125         // 模板名
126         String templateName = "mobile_setmeal_detail.ftl";
127         // 生成文件的全路径
128         String filename =
String.format("%s/setmeal_%d.html",out_put_path,setmeal.getId());
129         // 生成文件
130         generateFile(templateName,dataMap,filename);
131     }
132
133     /**
134     * 生成套餐列表页面
135     */
136     private void generateSetmealList(){
137         // 获取所有套餐信息
138         List<Setmeal> setmealList = setmealService.findAll();
139         setmealList.forEach(setmeal -> {
140             // 设置每个套餐图片的全路径
141             setmeal.setImg(QiniuUtils.DOMAIN + setmeal.getImg());
142         });
143         // 构建数据模型
144         Map<String,Object> dataMap = new HashMap<String,Object>();
145         // key setmealList 与模板中的变量要一致
146         dataMap.put("setmealList",setmealList);
147         // 生成的文件全路径
148         String setmealListFile = out_put_path + "/mobile_setmeal.html";
149         // 生成文件
150         generateFile("mobile_setmeal.ftl", dataMap, setmealListFile);
151     }
152
153     /**
154     * 生成文件
155     * @param templateName 模板名
156     * @param dataMap 要填充的数据
157     * @param filename 生成的文件名 全路径
158     */
159     private void generateFile(String templateName, Map<String,Object>
dataMap, String filename){
160         try {
161             // 获取模板
162             Template template = configuration.getTemplate(templateName);

```

```

163         // utf-8 不能少了。少了就中文乱码
164         BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(new FileOutputStream(filename),"utf-8"));
165         // 填充数据到模板
166         template.process(dataMap,writer);
167         // 关闭流
168         writer.flush();
169         writer.close();
170     } catch (Exception e) {
171         log.error("生成静态页面失败",e);
172     }
173 }
174 }
175

```

## 5.5. 配置任务

修改spring-jobs.xml配置文件

```

1 <!--导入freemarker配置-->
2 <import resource="classpath:spring-freemarker.xml"/>
3 <!--导入redis配置-->
4 <import resource="classpath:spring-redis.xml"/>

```

## 5.6. health\_web套餐功能修改

1. 添加redis配置文件，复制health\_jobs中的redis配置文件到health\_web的resources目录下

 image-20200911155445574

2. 修改springmvc.xml，导入redis的配置文件

```

1 <!--导入redis配置-->
2 <import resource="classpath:spring-redis.xml"/>

```

3. 修改health\_web中的SetmealController，分别修改添加、修改、删除的方法，添加相应的redis操作

- 在SetmealController中注入JedisPool

```

1 @Autowired
2 private JedisPool jedisPool;

```

- 修改SetmealService中的add方法，添加返回类型

```

1  /**
2   * 添加套餐
3   * @param setmeal
4   * @param checkgroupIds
5   */
6  Integer add(Setmeal setmeal, Integer[] checkgroupIds);

```

- 修改SetmealService中的add方法，添加返回新增套餐的id

```

1  /**
2   * 添加套餐
3   * @param setmeal
4   * @param checkgroupIds
5   */
6  @Override
7   @Transactional
8  public Integer add(Setmeal setmeal, Integer[] checkgroupIds) {
9      // 先添加套餐
10     setmealDao.add(setmeal);
11     // 获取新增的套餐的id
12     Integer setmealId = setmeal.getId();
13     // 遍历选中的检查组id,
14     if(null != checkgroupIds){
15         for (Integer checkgroupId : checkgroupIds) {
16             //添加套餐与检查组的关系
17
18             setmealDao.addSetmealCheckgroup(setmealId,checkgroupId);
19         }
20     }
21     return setmealId;
22 }

```

- 修改添加套餐的方法

```

1  /**
2   * 添加套餐
3   */
4  @PostMapping("/add")
5  public Result add(@RequestBody Setmeal setmeal, Integer[]
6  checkgroupIds){
7      // 调用服务添加
8      setmealService.add(setmeal,checkgroupIds);
9      // 获取redis连接对象
10     Jedis jedis = jedisPool.getResource();
11     // redis中set集合中保存的元素格式为: 套餐id|操作类型|时间戳
12     jedis.sadd("setmeal:static:html",setmeal.getId() + "|1|" +
13     System.currentTimeMillis());
14     // 还回连接池
15     jedis.close();
16     return new Result(true, MessageConstant.ADD_SETMEAL_SUCCESS);
17 }

```

- 修改更新套餐的方法

```

1  /**
2   * 更新套餐
3   */
4  @PostMapping("/update")
5  public Result update(@RequestBody Setmeal setmeal, Integer[]
checkgroupIds){
6      // 调用服务修改
7      setmealService.update(setmeal,checkgroupIds);
8      Jedis jedis = jedisPool.getResource();
9      jedis.sadd("setmeal:static:html",setmeal.getId() + "|1|" +
System.currentTimeMillis());
10     jedis.close();
11     return new Result(true, MessageConstant.EDIT_SETMEAL_SUCCESS);
12 }

```

- 修改删除套餐的方法


```

1  /**
2   * 通过id删除
3   */
4  @PostMapping("/deleteById")
5  public Result deleteById(int id){
6      setmealService.deleteById(id);
7      Jedis jedis = jedisPool.getResource();
8      // 操作符0代表删除
9      jedis.sadd("setmeal:static:html",id + "|0|" +
System.currentTimeMillis());
10     jedis.close();
11     return new Result(true, MessageConstant.DELETE_SETMEAL_SUCCESS);
12 }

```

## 5.7. health\_mobile页面连接修改

/pages/index.html页面的超链接地址进行修改：

image-20200911142007500

之前为：

修改为：

### 【小结】