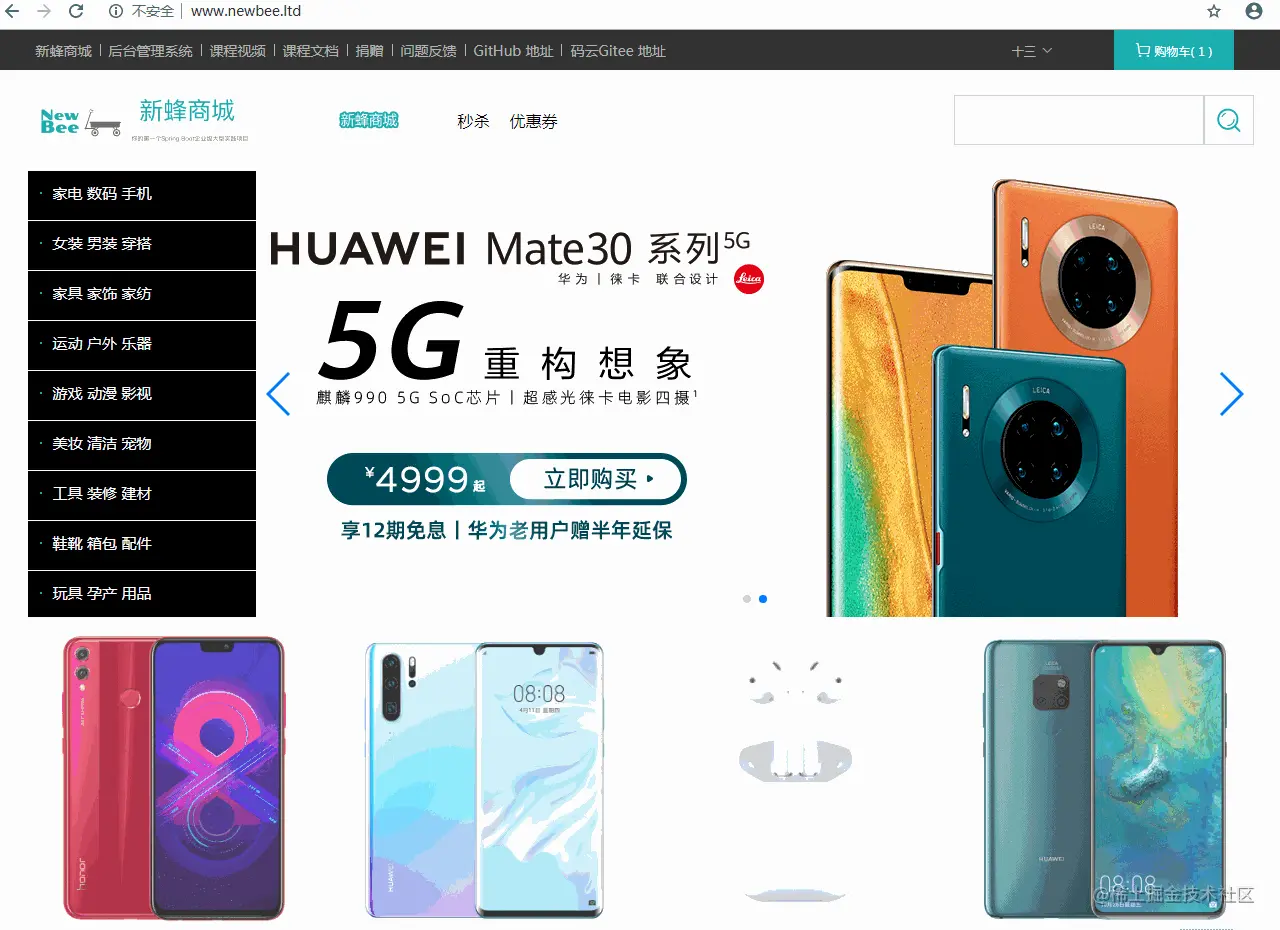
新蜂商城相关页面的设计和制作在会更注重用户体验，虽然涉及到的操作只是数据查询和数据聚合，并不代表开发难度就会降低，这些页面往往更加注重页面观感和元素设计，如果用户觉得第一眼之后就没有了看下去的欲望也是不行的，但是商城系统也并不需要做的过于绚丽多姿，简洁、美观、实用，拥有这三个特点足矣。

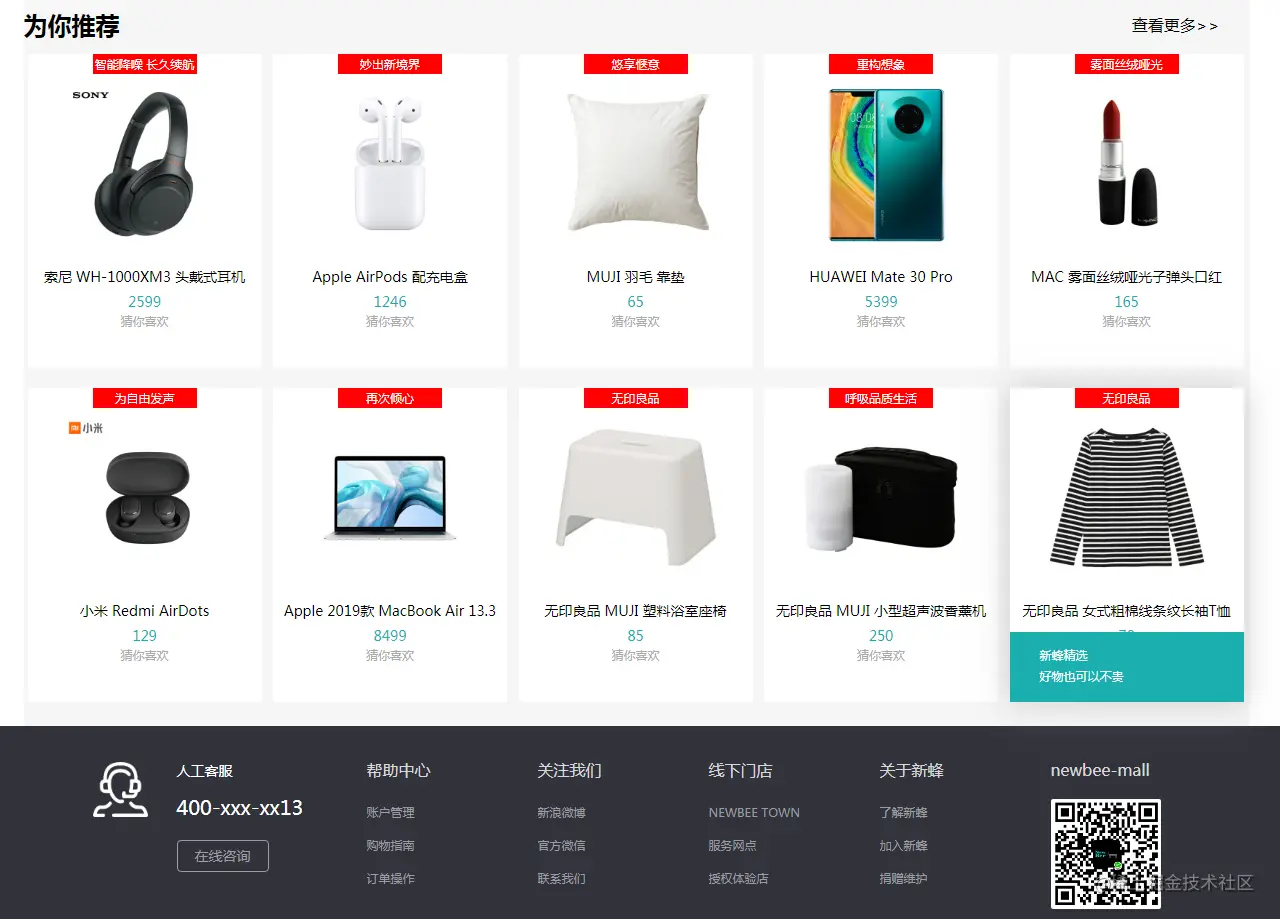
## 商城首页设计要素

在新蜂商城页面中，首页是最先被用户浏览到的页面，这个页面也是非常重要的入口，如果用户在该页面就萌生退意那就证明页面的设计还需要在仔细斟酌，因此新蜂商城的首屏页面也是重中之重，关于商城首页的设计和制作我在新蜂商城第一版 PC 端实现时，总结了以下几个注意事项：

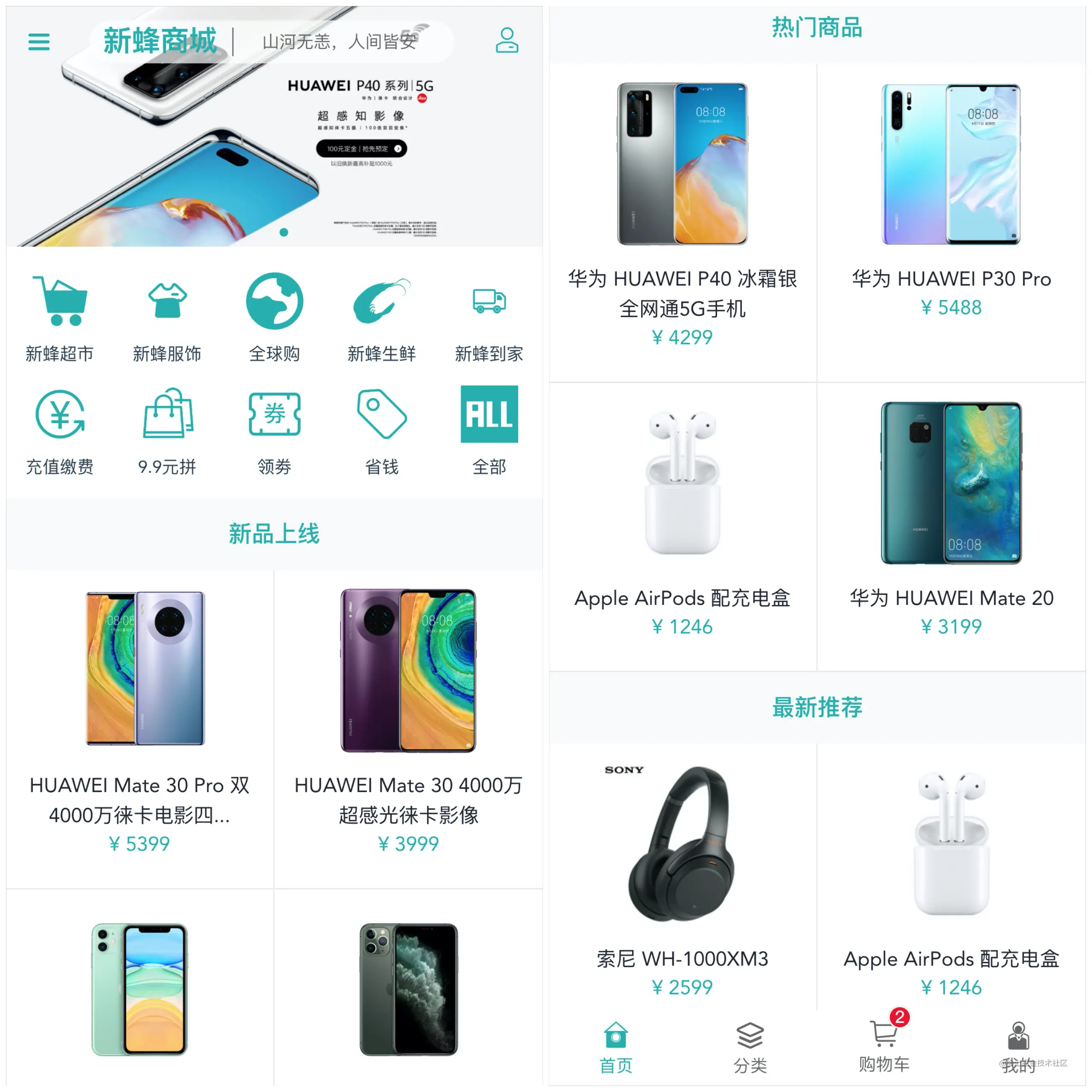
* 图文相结合这种方式最大的优势就是直观、易懂、生动，图文并茂可以更加吸引眼球
* 寸土寸金因此需要精心设计
* 不要过度设计，减少用户的猜测，简单、符合用户习惯即可。
* 清晰的网站 Logo 图片能让用户直观地记住商城的名称和形象，也能够给用户展现品牌的特色，一个优秀的 Logo 图片也会给网站带来积极的影响

视觉效果如下：



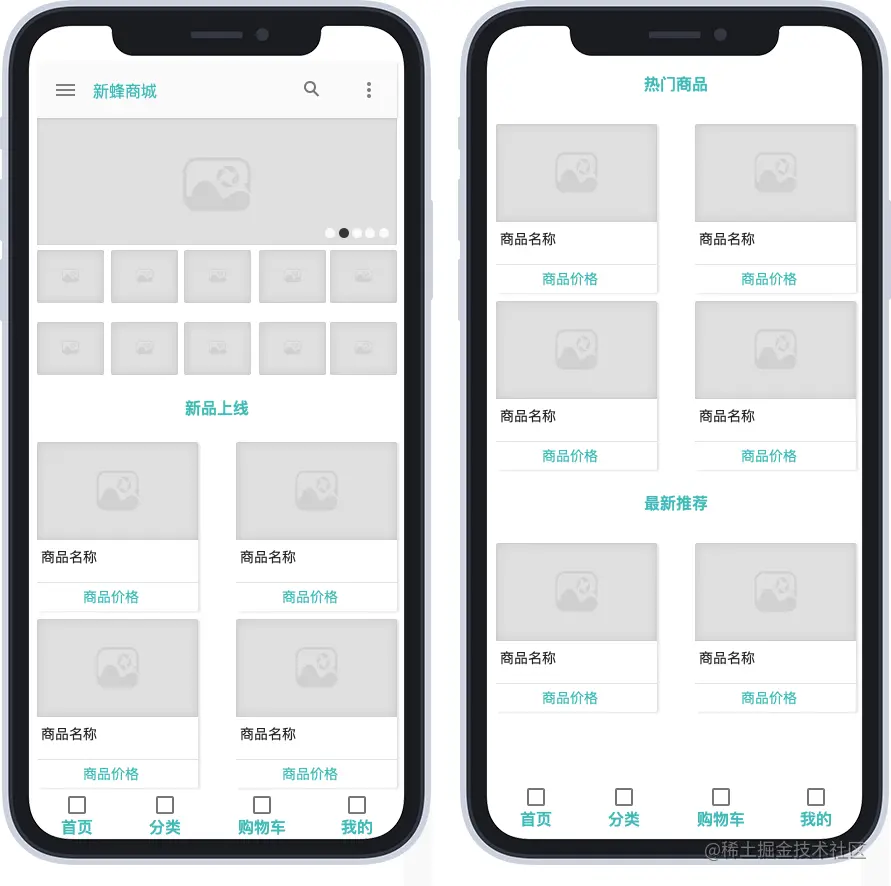


移动端与 PC 端肯定不同，首先从版面来说就有较大的区别，从大屏幕到小屏幕，屏幕的大小直接决定了页面所需要承载的内容量，PC 端一个页面能够展现给用户的内容量，就可能需要移动端好几个页面来完成，因此移动端的设计理念就是要比 PC 端更简洁、更要突出重点。



## 页面设计

通过前文中首页的页面展现效果，我们大致可以得出一个新蜂商城首页布局和排版，总结如下：



由上图可以看出，新蜂商城首屏页面的整个设计版面被切分成六个部分：

1. 商城 Logo 及搜索框：可以放置 Logo 图片、商品搜索输入框；
2. 轮播图：以轮播的形式展示后台配置的轮播图；
3. 热销商品：展示后台配置的热销商品数据；
4. 新品推荐：展示后台配置的新品数据；
5. 推荐商品：展示后台配置的推荐商品数据；
6. 导航栏：固定导航栏，放置新蜂商城几个重要的功能模块页面。

当然，以上版面设计的分析只是对于新蜂商城这个项目，开源社区或者企业开发中还有许多其他的商城系统项目，由于前端的设计和实现非常灵活且多变，这些商城系统可能又是另外一些页面样式和页面布局。

与 PC 端相比，新蜂商城移动端的首屏页面所展示的信息内容减少了，比如分类信息就单独划分了一个页面，数据量减少了，本来展示 10 条现在展示 4 条，其他介绍信息和链接信息也全部删除，放到一个单独的介绍页面来展示，只展示最重要的数据。

## 数据定义

页面布局和交互实现，由前端代码来实现，后端只需要将所需的数据，通过接口响应回去即可，页面中有一部分是静态数据，需要通过接口获取的数据有轮播图和推荐商品。

#### 轮播图和商品推荐

横跨屏幕的轮播图是时下比较流行的网页设计手法，在新蜂商城 PC 端中我添加了首屏轮播的效果，丰富版面的同时也提升了页面的感官，因此在移动端也嵌入了这个设计，关于配置轮播图的相关内容我已经在[《新蜂商城轮播图管理模块开发》](https://juejin.cn/book/5da2f9d4f265da5b81794d48/section/5dcf6f9be51d453ff236595d" \o "https://juejin.cn/book/5da2f9d4f265da5b81794d48/section/5dcf6f9be51d453ff236595d" \t "/home/erfenjiao/文档\\x/_blank)这一章节详细的讲解，这是后台管理系统中做的功能，商城端只需要读取和展示即可。

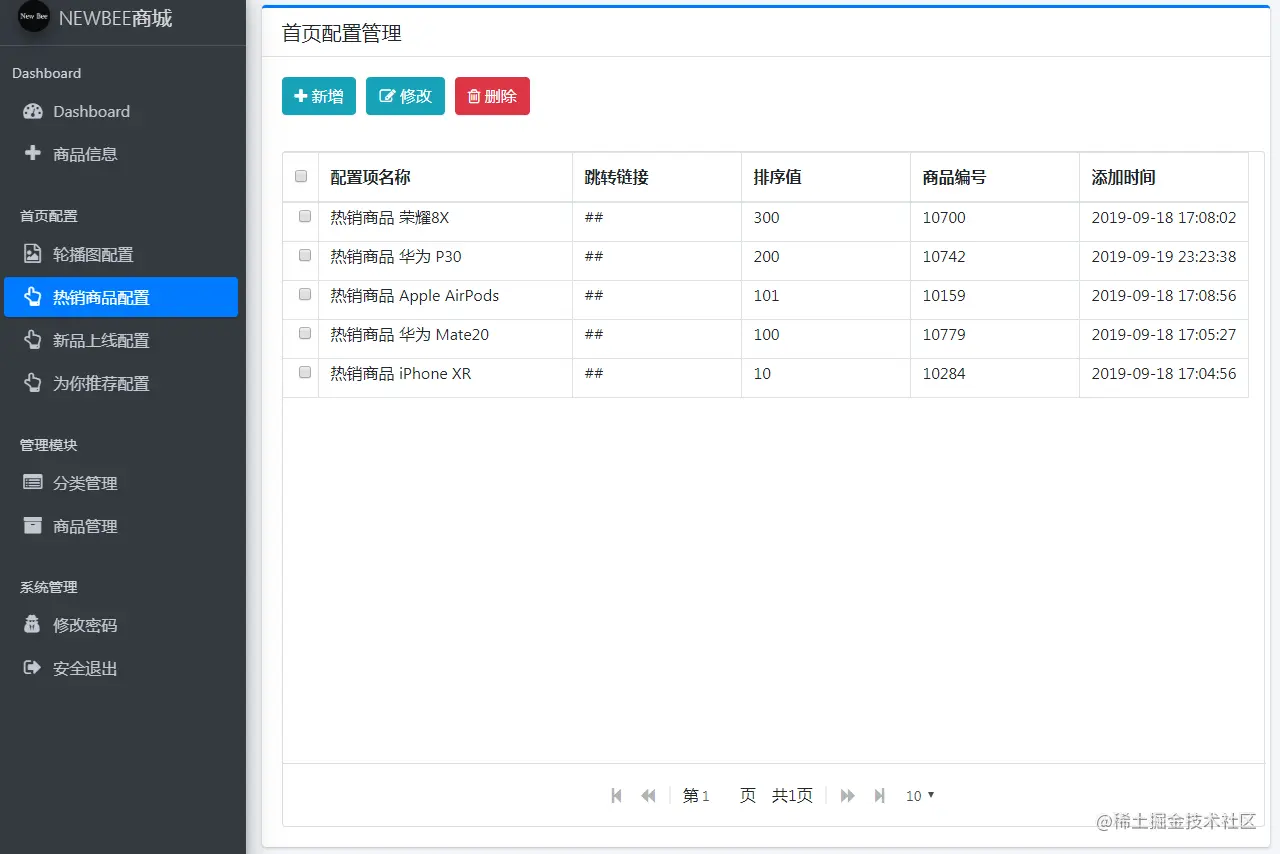
除了轮播图是后台配置，首页中还有三个版块的数据需要动态读取，分别是：

* ****热销商品****
* ****新品上线****
* ****推荐商品****

在首页中设计这三个版块儿，****主要是为了丰富版面，使得页面不是那么单调****；****当然，这部分的设计也是参考了其他商城的商品推荐设计，不过那些线上商城都有大量的正式数据做支撑，做的肯定要比新蜂商城复杂的多，比如热销商品，一定是在大量实际订单的统计下做出来的数据渲染，又比如商品推荐，也一定是在用户的浏览痕迹和下单习惯上计算出来的****。目前来说，新蜂商城的开发人员只有我一个人，下单量也很少很少，如果要做到淘宝、京东那种效果是万万不可能的，我会慢慢完善这个商城，也期待更多的开发者可以参与进来。

新蜂商城中的热销商品、新品上线、推荐商品这三个版块儿中的数据是在后台中配置的，首页直接读取就可以了，并没有进行实时的数据统计显示在页面上，因为数据量不大，那种做法意义不大。当然，这三个模块也都可以使用真实数据来实现，比如新品上线中的数据可以读取最新添加的几条商品数据，热销商品中的数据可以读取下单量最高的几条商品，这也是一种实现思路。

新蜂商城中这三个版块儿中的内容是在后台进行配置的，页面如下：



#### 数据格式

介绍完模块内容后，我们来定义一下首页接口中的返回数据。

首先是轮播图数据结构，我们需要将轮播图图片地址和点击轮播图后的跳转路径返回给前端，因此定义了 NewBeeMallIndexCarouselVO，代码如下：

*/\*\**

*\* 首页轮播图VO*

*\*/***@Datapublic** **class** **NewBeeMallIndexCarouselVO** **implements** **Serializable** {

**@ApiModelProperty("轮播图图片地址")**

**private** String carouselUrl;

**@ApiModelProperty("轮播图点击后的跳转路径")**

**private** String redirectUrl;

}

商品推荐模块中，我们可以看到每个商品信息都只有 3 个字段：商品名称、商品图片、商品价格，因为还需要进行跳转，所以我们需要加上商品 id，因此定义了 NewBeeMallIndexConfigGoodsVO，代码如下：

*/\*\**

*\* 首页配置商品VO*

*\*/***@Datapublic** **class** **NewBeeMallIndexConfigGoodsVO** **implements** **Serializable** {

**@ApiModelProperty("商品id")**

**private** Long goodsId;

**@ApiModelProperty("商品名称")**

**private** String goodsName;

**@ApiModelProperty("商品图片地址")**

**private** String goodsCoverImg;

**@ApiModelProperty("商品价格")**

**private** Integer sellingPrice;

}

当然，这些都是单项定义，轮播图可以返回多张，需要返回一个 List 数据，推荐商品有三种类型，每种类型也是多条商品数据，也是 List 类型的数据，最终首页返回的数据类型为 IndexInfoVO，代码为：

**@Datapublic** **class** **IndexInfoVO** **implements** **Serializable** {

**@ApiModelProperty("轮播图(列表)")**

**private** List<NewBeeMallIndexCarouselVO> carousels;

**@ApiModelProperty("首页热销商品(列表)")**

**private** List<NewBeeMallIndexConfigGoodsVO> hotGoodses;

**@ApiModelProperty("首页新品推荐(列表)")**

**private** List<NewBeeMallIndexConfigGoodsVO> newGoodses;

**@ApiModelProperty("首页推荐商品(列表)")**

**private** List<NewBeeMallIndexConfigGoodsVO> recommendGoodses;

}

## 首页接口实现

#### 表结构介绍

首页的轮播图数据和推荐商品数据，我们是通过读取 tb\_newbee\_mall\_carousel 表、tb\_newbee\_mall\_index\_config 表和 tb\_newbee\_mall\_goods\_info 表获得的，三张表的表结构如下：

**CREATE** **TABLE** `tb\_newbee\_mall\_carousel` (

`carousel\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT COMMENT '首页轮播图主键id',

`carousel\_url` **varchar**(100) **NOT** **NULL** **DEFAULT** '' COMMENT '轮播图',

`redirect\_url` **varchar**(100) **NOT** **NULL** **DEFAULT** '''##''' COMMENT '点击后的跳转地址(默认不跳转)',

`carousel\_rank` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '排序值(字段越大越靠前)',

`is\_deleted` tinyint(4) **NOT** **NULL** **DEFAULT** '0' COMMENT '删除标识字段(0-未删除 1-已删除)',

`create\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '创建时间',

`create\_user` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '创建者id',

`update\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '修改时间',

`update\_user` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '修改者id',

**PRIMARY** KEY (`carousel\_id`) **USING** BTREE

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 ROW\_FORMAT=**DYNAMIC**;

**CREATE** **TABLE** `tb\_newbee\_mall\_goods\_info` (

`goods\_id` **bigint**(20) unsigned **NOT** **NULL** AUTO\_INCREMENT COMMENT '商品表主键id',

`goods\_name` **varchar**(200) **NOT** **NULL** **DEFAULT** '' COMMENT '商品名',

`goods\_intro` **varchar**(200) **NOT** **NULL** **DEFAULT** '' COMMENT '商品简介',

`goods\_category\_id` **bigint**(20) **NOT** **NULL** **DEFAULT** '0' COMMENT '关联分类id',

`goods\_cover\_img` **varchar**(200) **NOT** **NULL** **DEFAULT** '/admin/dist/img/no-img.png' COMMENT '商品主图',

`goods\_carousel` **varchar**(500) **NOT** **NULL** **DEFAULT** '/admin/dist/img/no-img.png' COMMENT '商品轮播图',

`goods\_detail\_content` text **NOT** **NULL** COMMENT '商品详情',

`original\_price` **int**(11) **NOT** **NULL** **DEFAULT** '1' COMMENT '商品价格',

`selling\_price` **int**(11) **NOT** **NULL** **DEFAULT** '1' COMMENT '商品实际售价',

`stock\_num` **int**(11) unsigned **NOT** **NULL** **DEFAULT** '0' COMMENT '商品库存数量',

`tag` **varchar**(20) **NOT** **NULL** **DEFAULT** '' COMMENT '商品标签',

`goods\_sell\_status` tinyint(4) **NOT** **NULL** **DEFAULT** '0' COMMENT '商品上架状态 1-下架 0-上架',

`create\_user` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '添加者主键id',

`create\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '商品添加时间',

`update\_user` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '修改者主键id',

`update\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '商品修改时间',

**PRIMARY** KEY (`goods\_id`) **USING** BTREE

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 ROW\_FORMAT=**DYNAMIC**;

**CREATE** **TABLE** `tb\_newbee\_mall\_index\_config` (

`config\_id` **bigint**(20) **NOT** **NULL** AUTO\_INCREMENT COMMENT '首页配置项主键id',

`config\_name` **varchar**(50) **NOT** **NULL** **DEFAULT** '' COMMENT '显示字符(配置搜索时不可为空，其他可为空)',

`config\_type` tinyint(4) **NOT** **NULL** **DEFAULT** '0' COMMENT '1-搜索框热搜 2-搜索下拉框热搜 3-(首页)热销商品 4-(首页)新品上线 5-(首页)为你推荐',

`goods\_id` **bigint**(20) **NOT** **NULL** **DEFAULT** '0' COMMENT '商品id 默认为0',

`redirect\_url` **varchar**(100) **NOT** **NULL** **DEFAULT** '##' COMMENT '点击后的跳转地址(默认不跳转)',

`config\_rank` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '排序值(字段越大越靠前)',

`is\_deleted` tinyint(4) **NOT** **NULL** **DEFAULT** '0' COMMENT '删除标识字段(0-未删除 1-已删除)',

`create\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '创建时间',

`create\_user` **int**(11) **NOT** **NULL** **DEFAULT** '0' COMMENT '创建者id',

`update\_time` datetime **NOT** **NULL** **DEFAULT** CURRENT\_TIMESTAMP COMMENT '最新修改时间',

`update\_user` **int**(11) **DEFAULT** '0' COMMENT '修改者id',

**PRIMARY** KEY (`config\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

由于相关的添加操作和配置操作都是在后台管理系统中进行的，在首页接口中我们只需要查询相关数据即可。

#### 轮播图数据读取

NewBeeMallCarouselService 业务类的 getCarouselsForIndex() 方法实现如下：

*/\*\**

*\* 返回固定数量的轮播图对象(首页调用)*

*\**

*\* @param number*

*\* @return*

*\*/*

**public** List<NewBeeMallIndexCarouselVO> **getCarouselsForIndex**(**int** number) {

List<NewBeeMallIndexCarouselVO> newBeeMallIndexCarouselVOS = **new** **ArrayList**<>(number);

List<Carousel> carousels = carouselMapper.findCarouselsByNum(number);

**if** (!CollectionUtils.isEmpty(carousels)) {

newBeeMallIndexCarouselVOS = BeanUtil.copyList(carousels, NewBeeMallIndexCarouselVO.class);

}

**return** newBeeMallIndexCarouselVOS;

}

该方法的作用是返回固定数量的轮播图对象供首页数据渲染，渲染部分在后续章节中会介绍到，这里我们只介绍后端的接口实现逻辑，查询 SQL 语句为：

<select id="findCarouselsByNum" parameterType="int" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List"/>

from tb\_newbee\_mall\_carousel

where is\_deleted = 0

order by carousel\_rank desc

limit #{number}

</select>

该方法名称为 findCarouselsByNum()，参数为 number，类型为 SELECT 语句， 根据传入的 number 数值，查询固定数量的记录。

#### 首页推荐商品数据读取

分别读取一定数量的首页配置信息，并分别将其设置到 request 对象中。

NewBeeMallIndexConfigService 业务类的 getConfigGoodsesForIndex() 方法实现如下：

*/\*\**

*\* 返回固定数量的首页配置商品对象(首页调用)*

*\*/*

**public** List<NewBeeMallIndexConfigGoodsVO> **getConfigGoodsesForIndex**(**int** configType, **int** number) {

List<NewBeeMallIndexConfigGoodsVO> newBeeMallIndexConfigGoodsVOS = **new** **ArrayList**<>(number);

List<IndexConfig> indexConfigs = indexConfigMapper.findIndexConfigsByTypeAndNum(configType, number);

**if** (!CollectionUtils.isEmpty(indexConfigs)) {

*//取出所有的goodsId*

List<Long> goodsIds = indexConfigs.stream().map(IndexConfig::getGoodsId).collect(Collectors.toList());

List<NewBeeMallGoods> newBeeMallGoods = goodsMapper.selectByPrimaryKeys(goodsIds);

newBeeMallIndexConfigGoodsVOS = BeanUtil.copyList(newBeeMallGoods, NewBeeMallIndexConfigGoodsVO.class);

**for** (NewBeeMallIndexConfigGoodsVO newBeeMallIndexConfigGoodsVO : newBeeMallIndexConfigGoodsVOS) {

**String** goodsName = newBeeMallIndexConfigGoodsVO.getGoodsName();

*// 字符串过长导致文字超出的问题*

**if** (goodsName.length() > 30) {

goodsName = goodsName.substring(0, 30) + "...";

newBeeMallIndexConfigGoodsVO.setGoodsName(goodsName);

}

}

}

**return** newBeeMallIndexConfigGoodsVOS;

}

该方法的作用是返回数量的配置项对象供首页数据渲染，实现思路如下：

首先根据 configType 参数读取固定数量的首页配置数据，之后获取配置项中关联的商品记录属性，然后对字符串进行处理并封装到 VO 对象里，之后设置到 request 域中，这里主要是将需要的数据读取出来并封装成一个对象返回给视图层。

VO 对象就是视图层使用的对象，一般与 entity 对象有一点点区别，entity 对象中的字段与数据库表字段逐一对应，VO 对象里的字段则是视图层需要哪些字段就设置哪些字段，也可以不新增 VO 对象直接返回 entity 对象，这个取决于各个开发者的写法习惯。

查询 SQL 语句分别为：

<select id="findIndexConfigsByTypeAndNum" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List"/>

from tb\_newbee\_mall\_index\_config

where config\_type = #{configType} and is\_deleted = 0

order by config\_rank desc

limit #{number}

</select>

该方法名称为 findIndexConfigsByTypeAndNum()，参数为 configType 和 number，类型为 SELECT 语句， 根据传入的推荐商品类型和 number 数值，查询固定数量的记录。

<select id="selectByPrimaryKeys" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List"/>

from tb\_newbee\_mall\_goods\_info

where goods\_id in

<foreach item="id" collection="list" open="(" separator="," close=")">

#{id}

</foreach>

order by field(goods\_id,

<foreach item="id" collection="list" separator=",">

#{id}

</foreach>

);

</select>

该方法名称为 selectByPrimaryKeys()，参数为商品 id 的列表，类型为 SELECT 语句， 根据传入的商品主键列表，查询出所有的商品记录。

#### 首页接口实现

最终，我们的首页接口实现代码如下：

**@RestController@Api(value = "v1", tags = "新蜂商城首页接口")@RequestMapping("/api/v1")public** **class** **NewBeeMallIndexAPI** {

**@Resource**

**private** NewBeeMallCarouselService newBeeMallCarouselService;

**@Resource**

**private** NewBeeMallIndexConfigService newBeeMallIndexConfigService;

**@GetMapping("/index-infos")**

**@ApiOperation(value = "获取首页数据", notes = "轮播图、新品、推荐等")**

**public** Result<IndexInfoVO> **indexInfo**() {

**IndexInfoVO** indexInfoVO = **new** **IndexInfoVO**();

List<NewBeeMallIndexCarouselVO> carousels = newBeeMallCarouselService.getCarouselsForIndex(Constants.INDEX\_CAROUSEL\_NUMBER);

List<NewBeeMallIndexConfigGoodsVO> hotGoodses = newBeeMallIndexConfigService.getConfigGoodsesForIndex(IndexConfigTypeEnum.INDEX\_GOODS\_HOT.getType(), Constants.INDEX\_GOODS\_HOT\_NUMBER);

List<NewBeeMallIndexConfigGoodsVO> newGoodses = newBeeMallIndexConfigService.getConfigGoodsesForIndex(IndexConfigTypeEnum.INDEX\_GOODS\_NEW.getType(), Constants.INDEX\_GOODS\_NEW\_NUMBER);

List<NewBeeMallIndexConfigGoodsVO> recommendGoodses = newBeeMallIndexConfigService.getConfigGoodsesForIndex(IndexConfigTypeEnum.INDEX\_GOODS\_RECOMMOND.getType(), Constants.INDEX\_GOODS\_RECOMMOND\_NUMBER);

indexInfoVO.setCarousels(carousels);

indexInfoVO.setHotGoodses(hotGoodses);

indexInfoVO.setNewGoodses(newGoodses);

indexInfoVO.setRecommendGoodses(recommendGoodses);

**return** ResultGenerator.genSuccessResult(indexInfoVO);

}

}

方法名称为 indexInfo()，请求类型为 GET，映射的路径为 /api/v1/index-infos，响应结果类型为 Result，实际的 data 属性类型为之前已经介绍的 IndexInfoVO 对象。

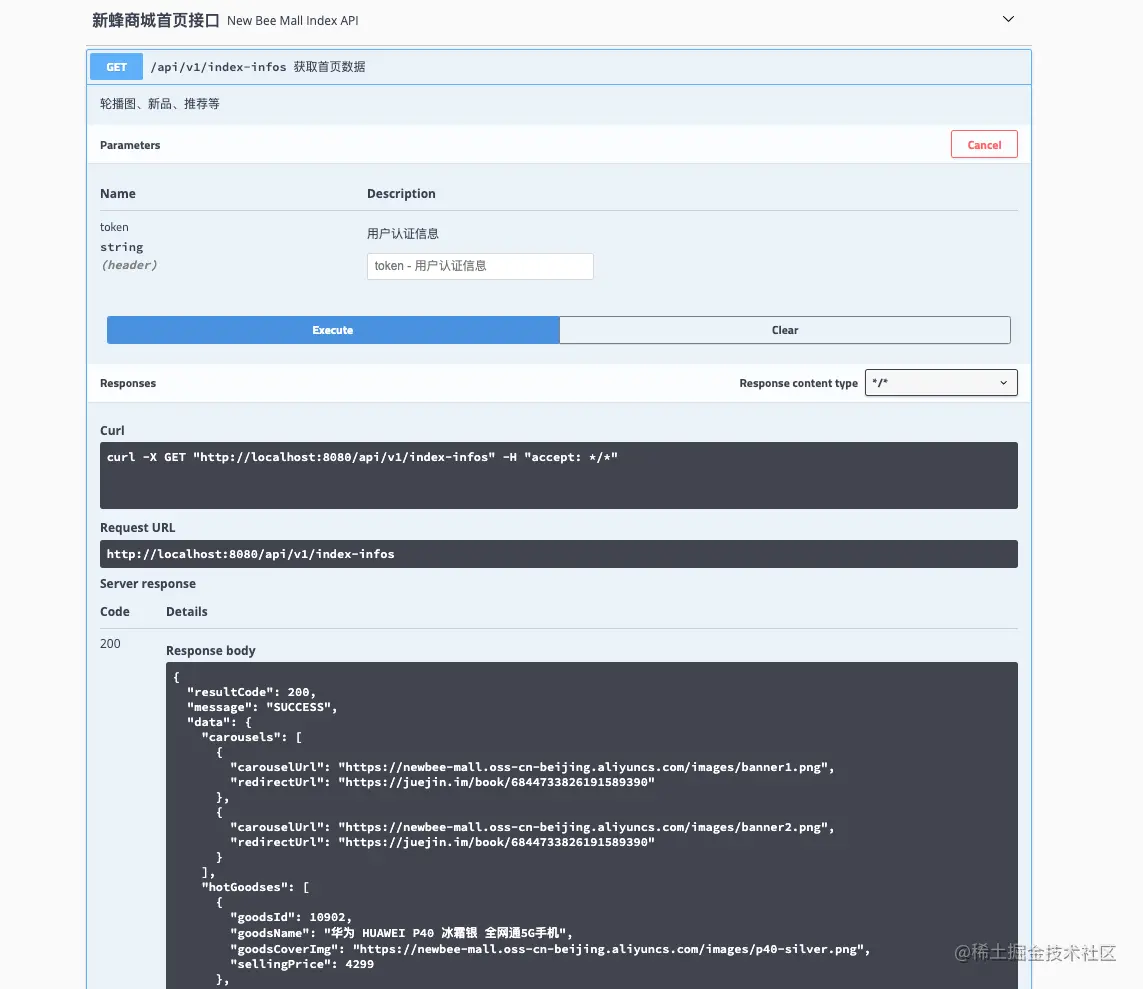
实现逻辑就是分别调用轮播图业务实现类 NewBeeMallCarouselService 中的查询方法和首页配置业务员实现类 NewBeeMallIndexConfigService 中的查询方法，将所需的数据查询到并且设置到 IndexInfoVO 对象中，最后响应给前端，因为商品推荐有热销商品、新品上线、推荐商品三种类型，所以 getConfigGoodsesForIndex() 方法也调用了三次，只是每次传入的参数并不相同。

还有一点需要注意，首页内容可以直接查看，并不需要登录认证，因此首页数据接口中并没有使用权限认证的注解 @TokenToMallUser。

#### 接口测试

最后，我们通过 Swagger 页面来测试一下首页数据接口。

重启项目，打开 swagger-ui 页面，点开“获取首页数据”，由于不需要身份验证，所以 token 输入框中不填入 token 值也可以获取到首页数据，之后点击”Execute“按钮，即可获取到首页展示所需要的数据，如下图所示：



接下来，我们打开控制台，并查看该接口返回的数据格式，如下图所示：



接收到的响应数据，是一个标准的 Result 对象，为 json 格式，字段分别为 resultCode、message 和 data，首页数据都在 data 中，有轮播图数据和推荐商品数据。

再次点开，可以看到列表格式的内容，如下图所示：



首页接口实现成功！

如果想要更改这部分配置内容，可以在新蜂商城后台管理系统中去操作。