

1 首页整体介绍

1-1 效果图



1-2 业务逻辑

- 1. 使用自定义组件的方式 实现 头部搜索框
- 2. 加载 轮播图 数据
- 3. 加载 分类导航 数据
- 4. 加载 楼层 数据

1-3 接口

1. 获取首页轮播图数据
`https://api-hmugo-web.itheima.net/api/public/v1/home/swiperdata`
2. 获取首页分类菜单数据
`https://api-hmugo-web.itheima.net/api/public/v1/home/catitems`
3. 获取首页 楼层数据
`https://api-hmugo-web.itheima.net/api/public/v1/home/floordata`

1-4 关键技术

小程序内置 `request` API

小程序 `swiper` 组件

自定义组件 实现 搜索框

2.自定义首页搜索框组件

==新增组件：`SearchInput`==

==自定义组件内容设计==

```
<view class="search_input">
  <navigator url="/pages/search/search" >搜索</navigator>
</view>

<!--
navigator微信文档 https://developers.weixin.qq.com/miniprogram/dev/component/navigator.html
navigator -- 页面链接。
-->
```

==自定义组件`SearchInput`样式设计==

```
.search_input{
  height: 90rpx;
  padding: 10rpx;
  background-color: var(--themeColor);
}

navigator{
  height: 100%;
  display: flex;
```

```
justify-content:center;
align-items: center;
background-color: #fff;
border-radius: 15rpx;
color: #666;
}
```

==在首页中声明引用组件==

```
// index.json

{
  "usingComponents": {
    "SearchInput": "../components/SearchInput/SearchInput"
  },
  "navigationBarTitleText": "商城首页"
}
```

```
<!-- 首页 index.wxml -->

<view class="gpmu_index">
  <!-- 搜索框开始 -->
  <SearchInput></SearchInput>
  <!-- 搜索框结束 -->
</view>
```

3.轮播图展示

获取首页轮播图数据

注意： 轮播图中，数据获取优先级更高些--先确立数据来源

```
Page({
  data: {
    // 轮播图数组
    swiperList:[],
  },

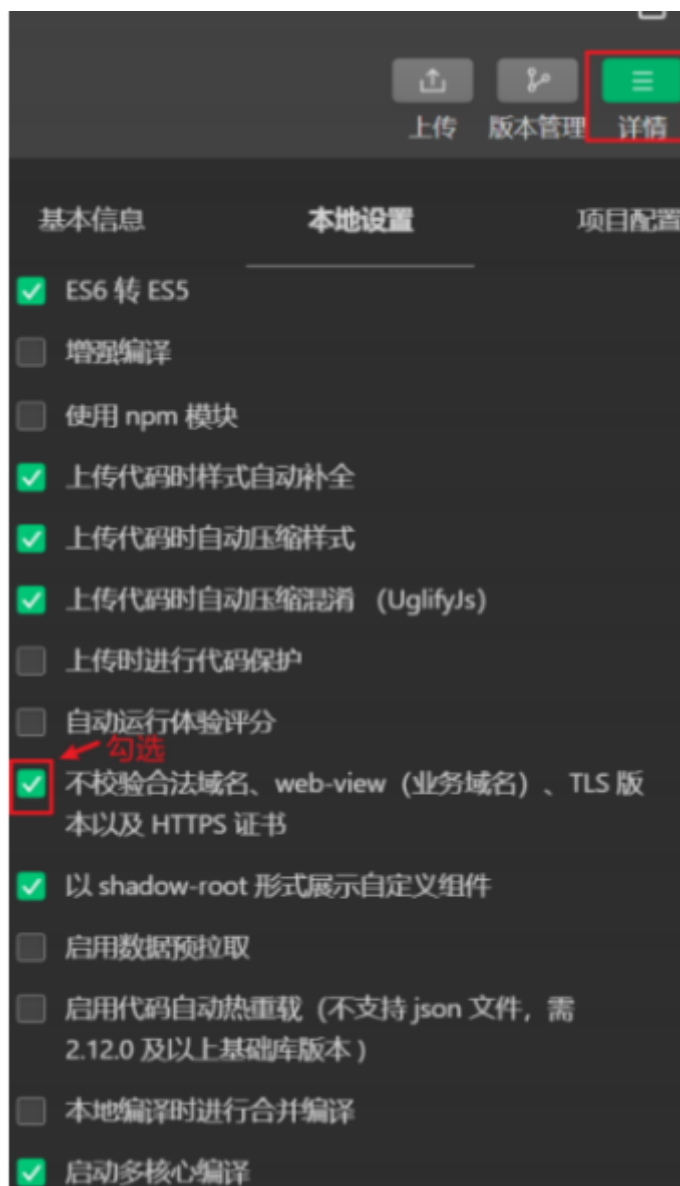
  //页面开始时加载， 就会触发
  onLoad: function (options) {
    // 1. 发送异步请求获取轮播图数据
    wx.request({
      url: 'https://api-hmugo-web.itheima.net/api/public/v1/home/swiperdata',
      success: (result) => {
        // console.log(result.data.message[0]); // 可以输出一下响应数据，获取目标数据对象
        this.setData({ // 通过setData方法更新 自定义属性 swiperList数据 --- setData 函数
          // 用于将数据从逻辑层发送到视图层，同时改变对应的 this.data 的值
          swiperList:result.data.message
        });
      },
    });
  },
});
```

```
},  
})
```

报错提示

▶ <https://api-hmugo-web.itheima.net> 不在以下 request 合法域名列表中,

解决方案1:



解决方案2

在微信公众平台的开发设置中添加服务器域名



==首页-轮播图-动态渲染==

```
<!-- index.wxml -->

<!-- 轮播图开始 -->
<view class="index_swiper">
  <!--
    1 swiper标签存在默认的宽度和高度100% * 150px
    2 image标签也存在默认的宽度和高度320px * 240px
    3 设计图片和轮播图
      1 先看一下原图的宽高 750 * 340
      2 让图片的高度自适应 宽度 等于100%
      3 让swiper标签的高度 变成和图片的高一样即可
    4 图片标签
      mode属性 渲染模式 ----- mode模式说明参考文章
      https://developers.weixin.qq.com/miniprogram/dev/component/image.html
      widthFix 让图片的标签宽高 和 图片标签的内容的宽高都等比例的变化
    5 wx:key详解
      https://www.jianshu.com/p/e81d29b01fbe
  -->

  <swiper autoplay indicator-dots circular>
    <swiper-item wx:for="{{swiperList}}" wx:key="goods_id">
      <navigator url="{{item.navigator_url}}">
        <image mode="widthFix" src="{{item.image_src}}"></image>
      </navigator>
    </swiper-item>
  </swiper>
</view>
<!-- 轮播图 结束 -->
```

```
<!-- index.wxss -->

.index_swiper swiper {
  width: 750rpx;
  height: 340rpx;
}

.index_swiper swiper image {
  width: 100%;
```

```
}
```

4. 首页分类导航

了解easy-less: <https://marketplace.visualstudio.com/items?itemName=mrcrowl.easy-less>

less语法了解: <https://lesscss.org/#overview>

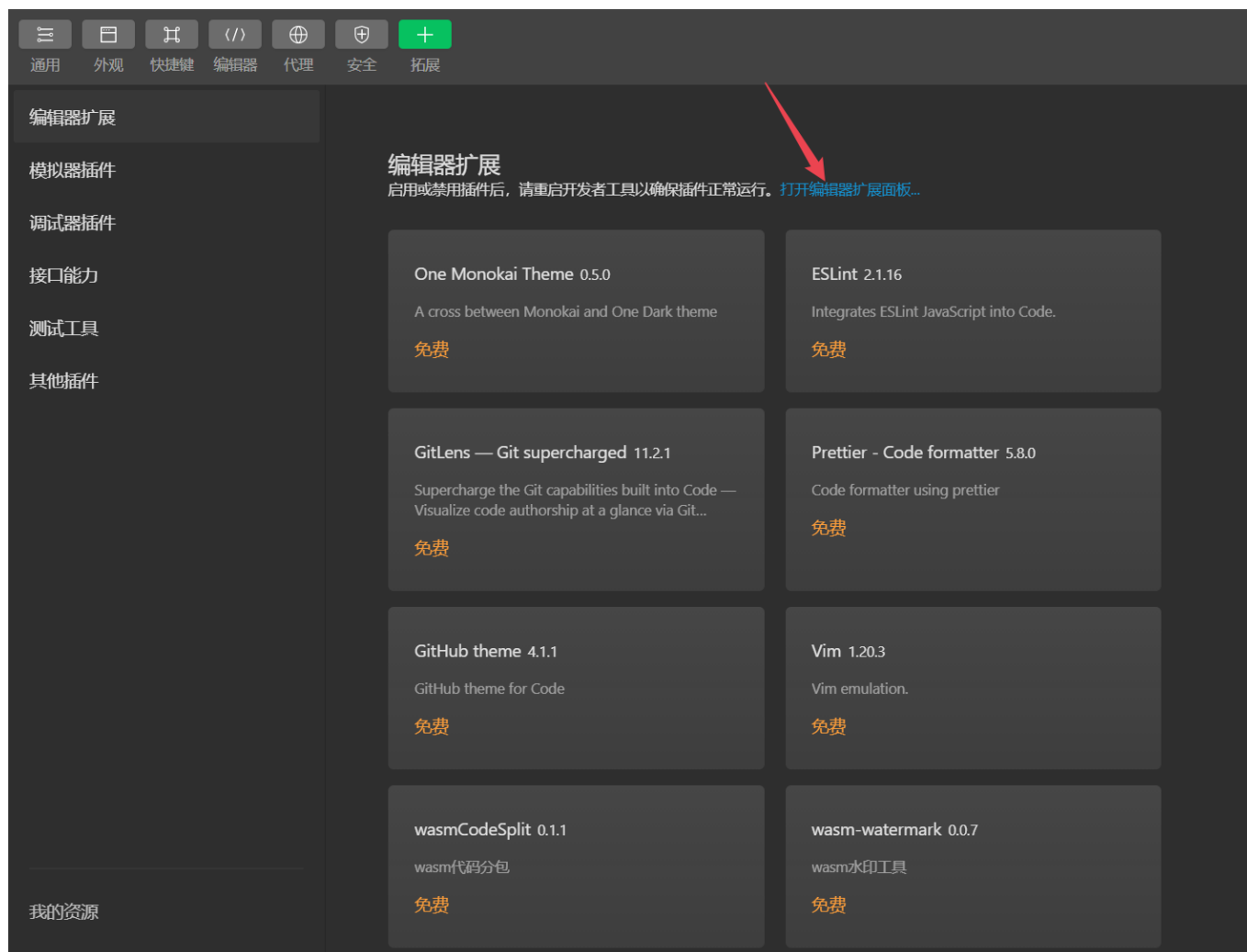
==配置less语法支持==

参考操作文:

<https://developers.weixin.qq.com/community/develop/article/doc/0008a475b40fd0c53c4bd0f905bc13>

准备好 easy-less插件, 且拷贝到以下目录(参考): C:\Users\lijun\AppData\Local\微信开发者工具\User Data\f13f1c30f325514de496c9db6e27b110\Default\Editor\User\extensions

设置--》扩展设置--》编辑器自定义扩展--》 打开easy less 开关 (新版本只需重启一下自动启用)



在settings.json配置 设置--》编辑器设置--》更多及工作区的设置--》搜索less 打开文档编辑 --- 选择 setting.json中编辑 --- 加入下述数据

```
"less.compile":{
  "outExt": ".wxss"
}
```

重启工具

"""

默认输出扩展名为 `.css`

这允许您指定替代输出文件扩展名（例如 `.less` 而不是 `.wxss.css`）

这也适用于文件（例如 `.map.wxss.map`）

"""

less

找到 27 个设置

用户 工作区

▼ 扩展 (27)

Easy LESS configura... (1)

Emmet (2)

JavaScript Debugger (1)

LESS (23)

LESS: Compile

Configuration options for Easy LESS extension.

在 [settings.json](#) 中编辑

```
<!-- index.js -->
```

```
data: {
  //轮播图数据，数组
  swiperList: [],
  //分类导航数据
  catesList:[]
},
```

```
onLoad: function (options) {
  // 1.获取轮播图数据
  this.getSwiperList();

  // 2.获取分类导航数据
  this.getCateList();
},
```

```
// 1.获取轮播图数据
getSwiperList(){
  // 1. 发送异步请求获取轮播图数据
  wx.request({
    url: 'https://api-hmugo-web.itheima.net/api/public/v1/home/swiperdata',
    success: (result) => {
```

```

        // console.log(result.data.message[0]);
        this.setData({
            swiperList:result.data.message
        });
    },
    });
},

// 2. 获取分类导航数据
getCateList(){
    wx.request({
        url: 'https://api-hmugo-web.itheima.net/api/public/v1/home/catitems',
        success: (result) => {
            this.setData({
                catesList:result.data.message
            });
        },
    });
}
}

```

```

<!-- index.wxml -->
<!-- 分类导航 开始 -->
<view class="index_cate">
    <navigator wx:for="{{catesList}}" wx:key="name" url="/pages/category/index" open-
type="switchTab">
        <image mode="widthFix" src="{{item.image_src}}"></image>
    </navigator>
</view>
<!-- 分类导航 结束 -->

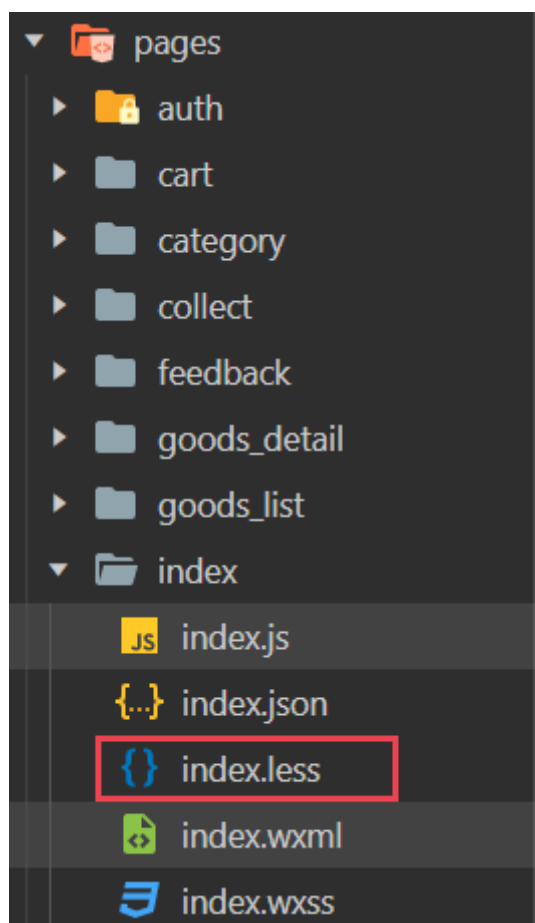
```

<!-- 首页 建立index.less文件 建议将index.wxss中轮播图的样式也转入index.less中，避免覆盖性问题-->

```

.index_cate{
    display: flex;
    navigator{
        padding:20rpx;
        flex:1;
        image{
            width:100%;
        }
    }
}

```

5. 首页楼层

```
<!-- index.js -->

//页面开始时加载， 就会触发
onLoad: function (options) {
  // 1. 获取轮播图数据
  this.getSwiperList();

  // 2. 获取分类导航数据
  this.getCateList();

  //3. 获取楼层
  this.getFloorList();
},

// 3. 获取楼层数据
getFloorList() {
  wx.request({
    url: 'https://api-hmugo-web.itheima.net/api/public/v1/home/floordata',
    success: (result) => {
      this.setData({
        floorList: result.data.message
      });
    },
  });
},
},
```

```

<!-- index.wxml -->

<!-- 楼层 开始 -->
<view class="index_floor">
  <view class="floor_group" wx:for="{{floorList}}" wx:for-item="item1" wx:for-index="index1"
wx:key="floor_title">
    <!-- 标题 -->
    <view class="floor_title">
      <image mode="widthFix" src="{{item1.floor_title.image_src}}"></image>
    </view>

    <!-- 内容 -->
    <view class="floor_content">
      <navigator wx:for="{{item1.product_list}}" wx:for-item="item2" wx:for-index="index2"
wx:key="name" url="{{item2.navigator_url}}">
        <image mode="{{index2===0?'widthFix':'scaleToFill'}}" src="{{item2.image_src}}">
      </image>
    </navigator>
  </view>
</view>
</view>
<!-- 楼层 结束 -->

```

```

<!-- index.less -->
.index_floor{
  .floor_group{
    .floor_title{
      padding: 10rpx 0;
      image{
        width: 100%;
      }
    }
    .floor_content{
      navigator{
        float: left;
        width: 33.33%;
        /* 后四个超链接 nth-last-child--https://developer.mozilla.org/zh-CN/docs/Web/CSS/:nth-last-child*/
        &:nth-last-child(-n+4){
          /* 原图的宽高 232 *386 */
          // 232 / 386 = 33.33vw / height
          // 第一张图片的高度 height:33.33vw * 386 / 232
          height:33.33vw * 386 / 232 / 2 ;
          border-left: 0rpx solid #fff;
        }

        /* 2 3 两个超链接 */
        &:nth-child(2),
        &:nth-child(3){
          border-bottom: 10rpx solid #fff;
        }
        image{
          width: 100%;
          height: 100%;
        }
      }
    }
  }
}

```

```
    }  
  }  
}
```

CSS样式中的 & 是什么作用？

写在前面： 当我在CSS样式中看到 **&**，我是十分懵逼的，不过搜索了一下就觉得不过如此。

贴个CSS代码进行讲解

```
1  .box{  
2      &:before{  
3          border-color: red;  
4      }  
5      &:after{  
6          border-color: green;  
7      }  
8  }
```

解析：

& 表示在嵌套层次中回溯一层，即

&:before相当于**.box:before**

&:after相当于**.box:after**

:before和**:after**是CSS中的伪元素，在这里分别表示为在使用box类的元素前面和后面插入指定内容