1.商品详情页面概述

效果

业务逻辑

1. 渲染商品详情数据
2. 点击图片，调出图片画廊，进行预览
3. 点击收藏
4. 联系客服
5. 分享功能
6. 加入购物车

接口

1. 获取详情数据接口
https://api-hmugo-web.itheima.net/api/public/v1/goods/detail

2. 加入购物车接口 使用本地存储来维护购物车数据

3. 立即购买接口 (相当于是 创建订单接口)
https://api-hmugo-web.itheima.net/api/public/v1/my/orders/create

关键技术

swiper组件
本地存储实现 收藏功能
联系客服 小程序管理后台中 直接添加即可
富文本标签 渲染 富文本
小程序 预览图片接口

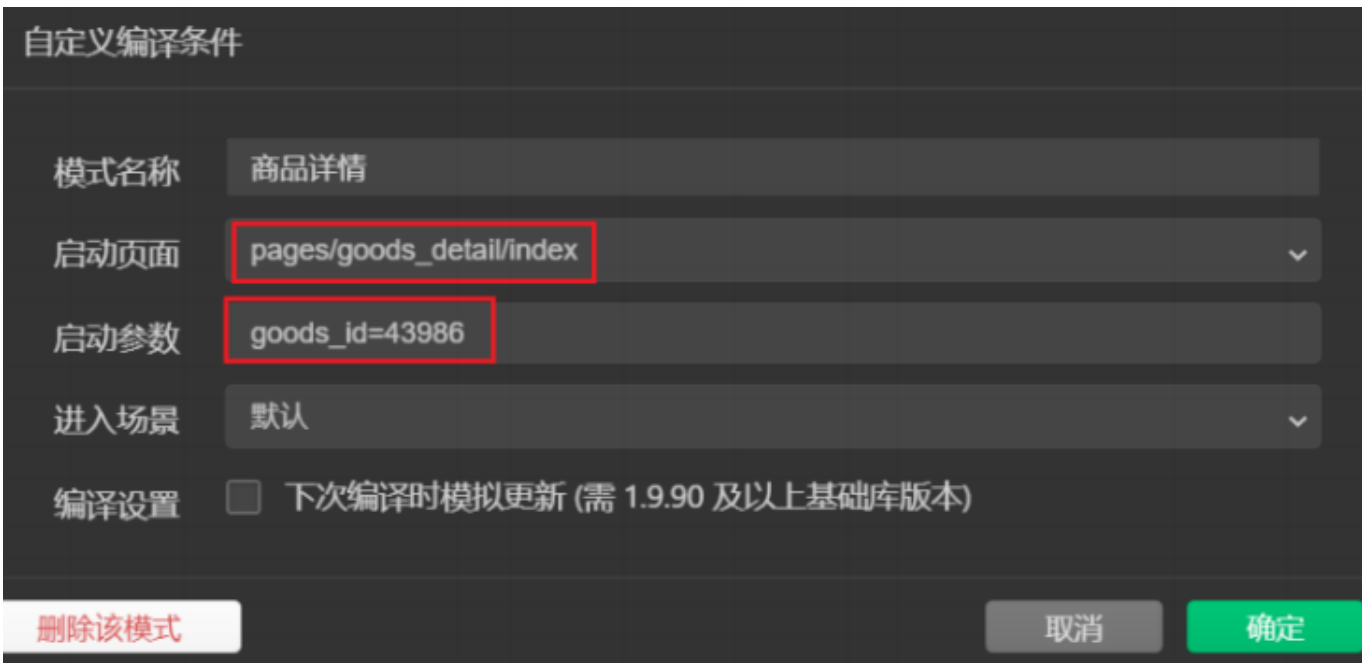
2.页面跳转及获取商品详情数据

a.商品列表页面传参及设置启动页面

商品列表的index.wxml

```
<Tabs tabs="{{tabs}}" bindtabsItemChange="handleTabsItemChange">
    <block wx:if="{{tabs[0].isActive}}">
        <view class="first_tab">
            <navigator class="goods_item" wx:for="{{goodsList}}"
            wx:key="{{goods_id}}"
            url="/pages/goods_detail/index?goods_id={{item.goods_id}}"
```

设置编译条件



b.获取商品详情数据

index.js

```
import { request } from "../../request/index.js";

Page({
    /**
    * 页面的初始数据
    */
    data: {
        goodsObj: {}
    },

    /**
    * 生命周期函数--监听页面加载
    */
    onLoad: function (options) {
        // console.log(options);
        const { goods_id } = options;
        this.getGoodsDetail(goods_id)
    },

    // 获取商品详情
    getGoodsDetail(goods_id) {
        request({
            url: "/goods/detail", data: { goods_id }
        })
            .then(res => {
                // console.log(res)
                this.setData({
                    goodsObj: res.data.message
                })
            })
    }
})
```

3.商品详情-接口数据和页面分析

4.商品详情- 轮播图动态渲染

index.wxml

```
<view class="tg_goods_ditail">
    <view class="detail_swiper">
        <swiper autoplay circular indicator-dots>
            <swiper-item wx:for="{{goodsObj.pics}}" wx:key="pics_id">
                <image mode="widthFix" src="{{item.pics_mid}}"></image>
            </swiper-item>
        </swiper>
    </view>
</view>
```

index.less

```
.detail_swiper{
    swiper{
        height: 65vw;
        text-align: center;
        image{
            width: 60%;
```

```
        }
    }
}
```

5.商品详情-价格-名称-图文详情

index.wxml

```
<view class="goods_price">￥{{goodsObj.goods_price}}</view>
<view class="goods_name_row">
    <view class="goods_name">{{goodsObj.goods_name}}{{goodsObj.goods_name}}</view>
    <view class="goods_collect" bindtap="handleCollect" >
        <text class="iconfont {{isCollect?'icon-shoucang1':'icon-shoucang'}} "></text>
        <view class="collect_text">收藏</view>
    </view>
</view>

<view class="goods_info">
    <view class="goods_info_title">图文详情</view>
    <view class="goods_info_content">
        <!-- 富文本 -->
        <!-- {{goodsObj.goods_introduce}} -->
        <rich-text nodes="{{goodsObj.goods_introduce}}"></rich-text>
    </view>
</view>
```

index.less

```
.goods_price{
    padding: 15rpx;
    font-size: 32rpx;
    font-weight: 600;
    color: var(--themeColor);
}

.goods_name_row{
    border-top: 5rpx solid #dedede;
    border-bottom: 5rpx solid #dedede;
    padding: 10rpx 0;
    display: flex;
    .goods_name{
        flex: 5;
        color: #000;
        font-size: 30rpx;
        padding: 0 10rpx;
        display: -webkit-box;
        overflow: hidden;
        -webkit-box-orient: vertical;
        -webkit-line-clamp:2;
    }
```

```
    .goods_collect{
        flex: 1;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        border-left: 1rpx solid #000;
        .iconfont{}
        .icon-shoucang1{
            color: orangered;
        }
        .collect_text{}
    }
}

.goods_info{
    .goods_info_title{
        font-size: 32rpx;
        color: var(--themeColor);
        font-weight: 600;
        padding: 20rpx;
    }
    .goods_info_content{}
}
```

6.动态渲染数据优化

需求： 剔除不需要渲染的数据

```
// 获取商品详情
getGoodsDetail(goods_id) {
    request({
        url: "/goods/detail", data: { goods_id }
    })
    .then(res => {
        console.log(res);
        const result = res.data.message;
        this.setData({
            goodsObj: {
                goods_price: result.goods_price,
                goods_name: result.goods_name,
                goods_introduce: result.goods_introduce,
                pics: result.pics
            }
        })
    })
}
```

7.点击轮播图，预览大图

```
思路
    给轮播图绑定点击事件
    调用小程序的api previewImage
```

index.wxml

```
<swiper autoplay circular indicator-dots>
    <swiper-item wx:for="{{goodsObj.pics}}" wx:key="pics_id" bindtap="handlePreViewImage" data-
url="{{item.pics_mid}}">
        <image mode="widthFix" src="{{item.pics_mid}}"></image>
    </swiper-item>
</swiper>
```

index.js

```
// 商品对象
GoodsInfo: {},
// 点击轮播图放大预览
handlePreViewImage(e) {
    // 1.先构造要预览的图片数组
    const urls = this.GoodsInfo.pics.map(v => v.pics_mid);
    // 2.接受传递过来的url
    const current = e.currentTarget.dataset.url;
    wx.previewImage({
        urls,
        current,
    })
}
```

提示： GoodsInfo需要先在网络请求request方法中赋值

```
this.GoodsInfo =res.data.message;
```

8.商品详情-底部工具栏

a.界面实现

==index.wxml

```
<view class="btm_tool">
    <view class="tool_item">
        <view class="iconfont icon-kefu"></view>
        <view>客服</view>
    </view>

    <view class="tool_item">
        <view class="iconfont icon-yixianshi-"></view>
        <view>分享</view>
    </view>

    <view class="tool_item">
        <view class="iconfont icon-gouwuche"></view>
        <view>购物车</view>
    </view>

    <view class="tool_item btn_cart ">加入购物车</view>
    <view class="tool_item btn_buy">立即购买</view>
</view>
```

index.less

```
page{
    padding-bottom: 90rpx;
}

.btm_tool{
    border-top: 1rpx solid #ccc;
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 90rpx;
    background-color: #fff;
    display: flex;
    .tool_item{
        flex: 1;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        font-size: 24rpx;
        position: relative;
    }

    .btn_cart{
        flex: 2;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: #ffa500;
        color: #fff;
        font-size: 30rpx;
        font-weight: 600;
    }
```

```
    .btn_buy{
        flex: 2;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: #eb4450;
        color: #fff;
        font-size: 30rpx;
        font-weight: 600;
    }
}
```

b.添加客服，分享，购物车跳转

index.wxml

```
<view class="btm_tool">
    <view class="tool_item">
        <view class="iconfont icon-kefu"></view>
        <view>客服</view>
        <!-- 添加按钮控件 -->
        <button open-type="contact"></button>
    </view>

    <view class="tool_item">
        <view class="iconfont icon-yixianshi-"></view>
        <view>分享</view>
        <button open-type="share"></button>
    </view>

    <!-- view改为navigator按钮，假如跳转到tab栏的url，需设置open-type="switchTab" -->
    <navigator open-type="switchTab" url="/pages/cart/index" class="tool_item">
        <view class="iconfont icon-gouwuche"></view>
        <view>购物车</view>
    </navigator>

    <view class="tool_item btn_cart ">加入购物车</view>
    <view class="tool_item btn_buy">立即购买</view>
</view>
```

index.less

```
.tool_item{
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    font-size: 24rpx;
    position: relative;
    // 设置按钮样式
    button{
        position: absolute;
```

```
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        opacity: 0;
    }
}
```

9.添加购物车

思路分析

1. 先绑定点击事件
2. 获取缓存中的购物车数据 数组格式
3. 先判断 当前的商品是否已经存在于 购物车
4. 已经存在 修改商品数据 执行购物车数量++ 重新把购物车数组 填充回缓存中
5. 不存在于购物车的数组中 直接给购物车数组添加一个新元素 新元素 带上 购买数量属性 num 重新把购物车数组填充回缓存中
6. 弹出提示

index.wxml

```
<view class="tool_item btn_cart " bindtap="handleCartAdd">加入购物车</view>
```

index.less

```
// 点击 加入购物车
handleCartAdd() {
    // 1 获取缓存中的购物车 数组
    let cart = wx.getStorageSync("cart") || [];

    // 2 判断 商品对象是否存在于购物车数组中
    let index = cart.findIndex(v => v.goods_id === this.GoodsInfo.goods_id);

    if (index === -1) {
        //3 不存在 第一次添加
        this.GoodsInfo.num = 1;
        cart.push(this.GoodsInfo);
    } else {
        // 4 已经存在购物车数据 执行 num++
        cart[index].num++;
    }

    // 5 把购物车重新添加回缓存中
    wx.setStorageSync("cart", cart);

    // 6 弹窗提示
    wx.showToast({
        title: '加入成功',
        icon: 'success',
        // true 防止用户 手抖 疯狂点击按钮
```

```
        mask: true
    });
},
```