# 单元1 需求分析

## 任务1.1 功能介绍

1. 首页

1.1首页主体内容共展示3个模块，分别为电影，电视剧，综艺。每个模块可左右滑动查看最多7条数据，如果需要查看更多，点击模块右上方更多按钮即可。

1.2 首页上方为搜索功能，点击跳转搜索页面。

1. 搜索页

2.1可实现实时数据搜索功能。

2.2 对搜索过的数据，有历史记录存储功能。

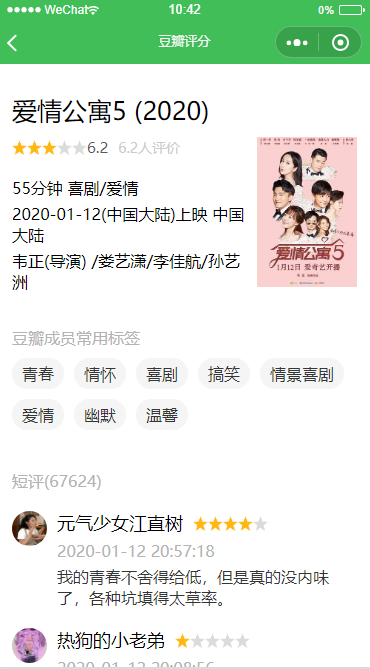
1. 列表页

由首页三大模块更多按钮进入，可查看当前模块所有数据。以列表形式展现数据，可上下滚动查看所有数据。



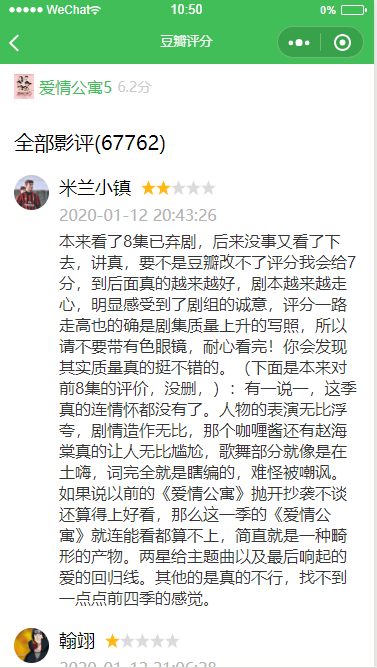
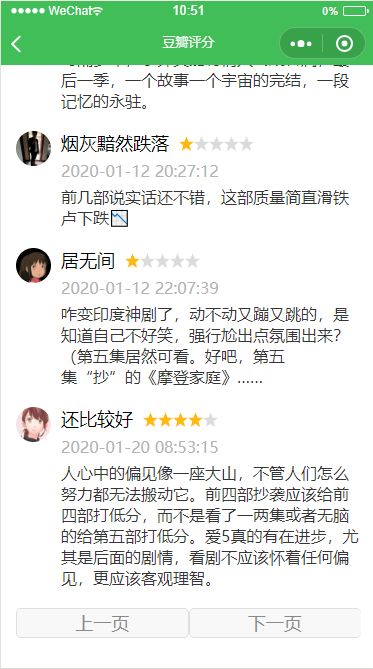
1. 详情页

展示数据详细信息，包括电影简介、所属标签、短评等信息。页面底部可以跳转到更多短评信息页面。

1. 短评列表页

由详情页查看更多短评按钮进入，展示该条数据所有短评信息，页面底部可进行翻页操作。页面顶部为数据基本信息，点击可返回上一层页面。

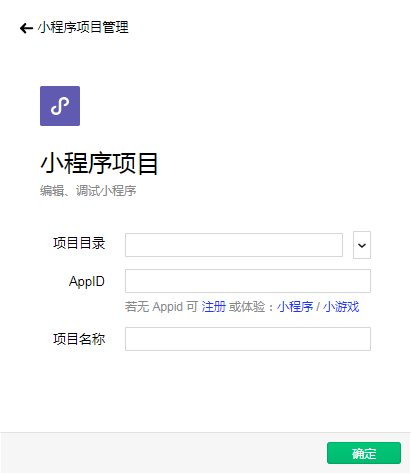
 

## 任务1.2 项目准备

1.选择小程序项目。



2.根据提示填写项目信息。



**注意：AppID可以直接选择小程序/小游戏选项，但不能进行小程序发布的操作。**

查看AppID，进入开发者公众平台，点击如下图所示按钮。



下图所示就是我们需要用的AppID。

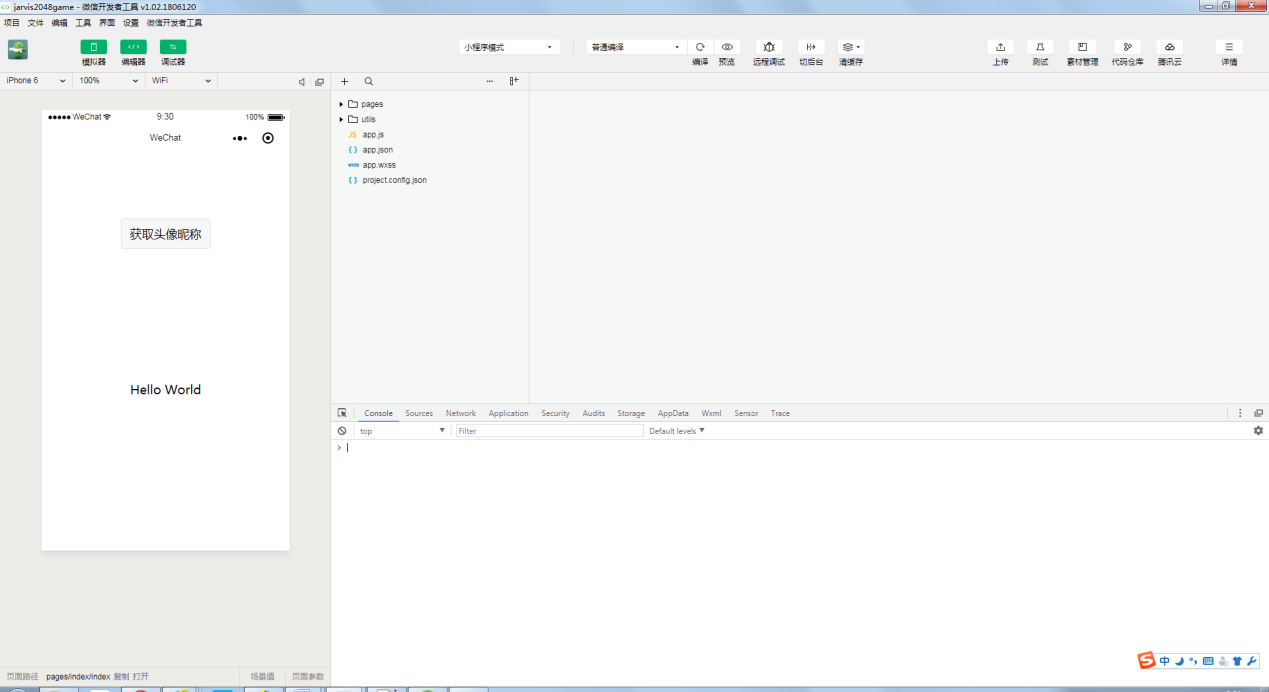


3.当填写完所有信息之后，会出现下图所示样子。



**注意：勾选并且选择简历普通快速启动模板。**

4.点击确认按钮，新建的第一个项目就完成了。



5. 整理初始代码，删除部分文件及代码。

5.1 打开app.js文件，删除自带代码，只保留以下代码。

//app.js

App({

})

5.2 删除logs文件包。

5.3 打开app.json文件，删除"pages/logs/logs"，保留以下代码。

{

"pages":[

"pages/index/index"

],

"window": {

"backgroundTextStyle": "light",

"navigationBarBackgroundColor": "#41be57",

"navigationBarTitleText": "豆瓣评分",

"navigationBarTextStyle": "white"

},

}

5.4 删除index.wxml、index.wxss和app.wxss中所有代码。

5.5 打开index.js文件，删除自带代码，只保留以下代码。

//index.js

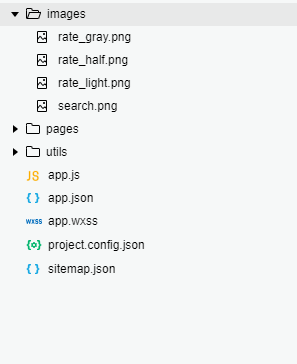
//获取应用实例

const app = getApp()

Page({

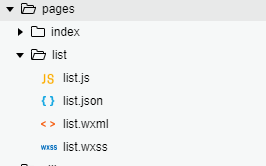
})

5.6 将附件中的imgage文件夹拷贝进项目，放在与pages同级目录下。

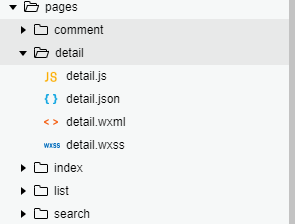


6新建所需页面

6.1新建pages/list目录，在该目录下新建list页面。项目结构如下图：



6.2 相同方法，新建search、detail、comment页面，项目 结构如下图：



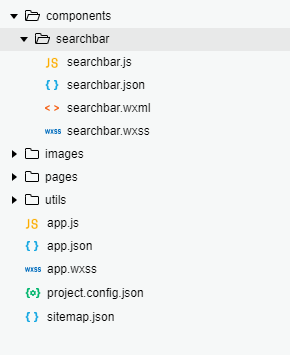
# 单元2 自定义组件封装

## 任务2.1 搜索栏实现

该视图应在多个页面多次使用，所以我们直接将它封装成自定义组件，方便调用。



1.新建components/searchbar目录，在该目录下新建components，命名为searchbar。项目结构如下图：



2.搜索栏的视图层渲染

2.1打开searchbar.wxml文件，添加如下代码：

<view class="searchbar">

<navigator wx:if="{{isnavigator}}" url='/pages/search/search' class='search-navigator'></navigator>

<view wx:else class="search-input-group">

<input class='search-input' placeholder='搜索' bindinput='onInputEvent'></input>

</view>

</view>

2.2 打开searchbar.wxss文件,为视图层添加样式文件，代码如下：

备注：标记为base64代码字样部分，需将搜索图标转换为base64格式展示，提供在线转换工具：<http://tool.chinaz.com/tools/imgtobase/>

.searchbar{

background-color: #41be57;

padding: 20rpx;

}

.search-navigator{

width: 100%;

height: 60rpx;

background-color: #fff;

border-radius: 10rpx;

background-image: url("base64代码");

background-position: center center;

background-repeat: no-repeat;

background-size: 8%;

}

.search-input-group{

width: 100%;

height: 60rpx;

background-color: #fff;

border-radius: 10rpx;

padding: 10rpx 20rpx;

box-sizing: border-box;

}

.search-input{

min-height: 40rpx;

height: 40rpx;

font-size: 12px;

}

3 打开searchbar.js文件，为该自定义组件添加属性值，在properties对象下添加如下代码：

isnavigator: {

type: Boolean,

value: false

}

4 为该自定义组件添加onInputEvent方法，实时监听输入框属性值变化，在methods对象下添加如下代码：

onInputEvent: function(event){

var value = event.detail.value;

var detail = {"value": value};

var options = {};

this.triggerEvent("searchinput",detail,options);

}

5.组件测试

5.1打开index.json文件，在usingComponents对象下添加组件引用代码：

"searchbar": "/components/searchbar/searchbar",

5.2打开index.wxml文件，添加搜索栏组件代码：

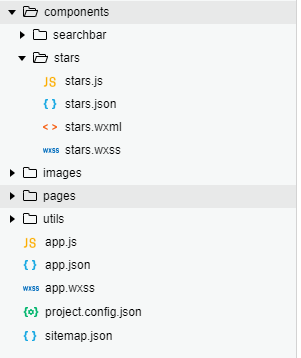
<searchbar isnavigator="{{true}}"></searchbar>

## 任务2.2 评分星星组件实现

该视图应在多个页面多次使用，所以我们直接将它封装成自定义组件，方便调用。



1.新建components/stars目录，在该目录下新建components，命名为stars。项目结构如下图：



2.评分星星组件的视图层渲染

2.1打开stars.wxml文件，添加如下代码：

<view class='rate-group'>

<image style='width:{{starsize}}rpx;height:{{starsize}}rpx;' wx:for="{{lights}}" wx:key="\*this" src="/images/rate\_light.png"></image>

<image style='width:{{starsize}}rpx;height:{{starsize}}rpx;' wx:for="{{halfs}}" wx:key="\*this" src='/images/rate\_half.png'></image>

<image style='width:{{starsize}}rpx;height:{{starsize}}rpx;' wx:for="{{grays}}" wx:key="\*this" src='/images/rate\_gray.png'></image>

<text wx:if="{{istext}}" style='font-size:{{fontsize}}rpx;color:{{fontcolor}};'>{{ratetext}}</text>

</view>

2.2 打开stars.wxss文件,为视图层添加样式文件，代码如下：

.rate-group{

display: flex;

justify-content: center;

align-items: center;

font-size: 20rpx;

color: #ccc;

}

.rate-group image{

width: 20rpx;

height: 20rpx;

}

3 打开stars.js文件，为该自定义组件添加updateRate方法，根据传入分值计算出满星星、半星星和灰色星星的个数，在methods对象下添加如下代码：

updateRate: function(){

var that = this;

var rate = that.properties.rate;

var intRate = parseInt(rate);

var light = parseInt(intRate / 2);

var half = intRate % 2;

var gray = 5 - light - half;

var lights = [];

var halfs = [];

var grays = [];

for (var index = 1; index <= light; index++) {

lights.push(index);

}

for (var index = 1; index <= half; index++) {

halfs.push(index);

}

for (var index = 1; index <= gray; index++) {

grays.push(index);

}

var ratetext = rate && rate > 0 ? rate.toFixed(1) : "未评分"

that.setData({

lights: lights,

halfs: halfs,

grays: grays,

ratetext: ratetext

});

}

4为该自定义组件添加属性值，在properties对象下添加如下代码：

rate: {

type: Number,

value: 0,

observer(newVal, oldVal, changedPath) {

this.updateRate();

}

},

starsize: {

type: Number,

value: 20 //rpx

},

fontsize: {

type: Number,

value: 20 // rpx

},

fontcolor: {

type: String,

value: "#ccc"

},

istext: {

type: Boolean,

value: true

}

5 添加生命周期函数，在组件挂载时调用星星个数统计，添加代码如下：

lifetimes: {

attached: function(){

this.updateRate();

}

}

6.组件测试

6.1打开index.json文件，在usingComponents对象下添加组件引用代码：

"stars": "/components/stars/stars",

6.2打开index.wxml文件，添加搜索栏组件代码：

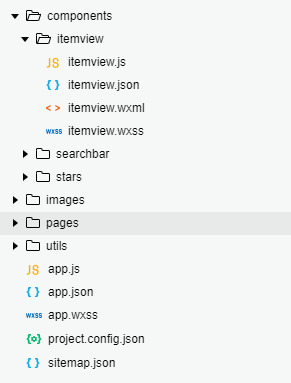
<stars rate="6"></stars>

## 任务2.3 数据模块组件实现

该视图应在多个页面多次使用，所以我们直接将它封装成自定义组件，方便调用。



1.新建components/ itemview目录，在该目录下新建components，命名为itemview。项目结构如下图：



2.数据模块组件的视图层渲染

2.1打开itemview.json文件，在usingComponents对象下添加组件引用代码：

"stars": "/components/stars/stars"

2.2打开itemview.wxml文件，添加如下代码：

<navigator wx:if="{{item}}" class='item-navigator' url="{{itemurl}}">

<view class='item-group'>

<view class='thumbnail-group'>

<image class='thumbnail' src='{{item.cover.url}}'></image>

</view>

<view class='item-title'>{{item.title}}</view>

<stars rate="{{item.rating.value}}"></stars>

</view>

</navigator>

<view wx:else class="item-navigator"></view>

2.3 打开itemview.wxss文件,为视图层添加样式文件，代码如下：

.item-navigator{

width: 200rpx;

margin-right: 20rpx;

display: inline-block;

}

.item-navigator .item-group{

width: 100%;

}

.item-group .thumbnail-group{

width: 100%;

height: 284rpx;

}

.thumbnail-group .thumbnail{

width: 100%;

height: 100%;

}

.item-group .item-title{

font-size: 28rpx;

text-align: center;

margin-top: 20rpx;

text-overflow: ellipsis;

overflow: hidden;

margin-bottom: 20rpx;

}

3打开itemview.js文件为该自定义组件添加属性值，在properties对象下添加如下代码：

item: {

type: Object,

value: {}

},

itemurl: {

type: String,

value: ""

}

6.组件测试

6.1打开index.json文件，在usingComponents对象下添加组件引用代码：

"itemview": "/components/itemview/itemview"

6.2打开index.wxml文件，添加搜索栏组件代码：

<itemview item="{{item}}" itemurl="/pages/detail/detail?type={{type}}&id={{item.id}}"></itemview>

6.3 打开index.js文件，为页面添加测试数据，代码如下：

data:{

item:{

cover:{

url:"https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg"

},

rating:{

value:6.7

},

title:"jarvis"

}

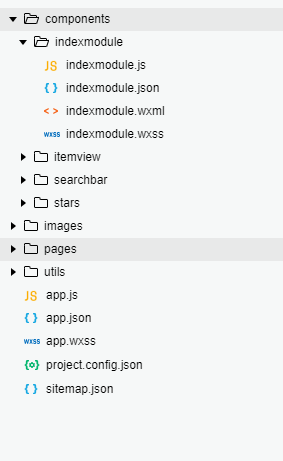
}

## 任务2.4 首页模块组件实现

该视图应在多个页面多次使用，所以我们直接将它封装成自定义组件，方便调用。



1.新建components/ indexmodule目录，在该目录下新建components，命名为indexmodule。项目结构如下图：



2.数据模块组件的视图层渲染

2.1打开indexmodule.json文件，在usingComponents对象下添加组件引用代码：

"itemview": "/components/itemview/itemview"

2.2打开indexmodule.wxml文件，添加如下代码：

<view class='module-group'>

<view class='module-top'>

<view class='module-title'>{{title}}</view>

<navigator url="{{moreurl}}" class='module-more'>更多</navigator>

</view>

<scroll-view class='module-scroll-view' scroll-x="{{true}}">

<itemview wx:for="{{items}}" wx:key="{{item.title}}" item="{{item}}" itemurl="/pages/detail/detail?type={{type}}&id={{item.id}}"></itemview>

</scroll-view>

</view>

2.3 打开indexmodule.wxss文件,为视图层添加样式文件，代码如下：

.module-group{

padding: 40rpx;

background-color: #fff;

}

.module-group .module-top{

font-size: 36rpx;

display: flex;

justify-content: space-between;

}

.module-top .module-title{

color: #494949;

}

.module-top .module-more{

color: #41be57;

}

.module-scroll-view{

margin-top: 40rpx;

width: 100%;

height: 400rpx;

white-space: nowrap;

}

3打开indexmodule.js文件为该自定义组件添加属性值，在properties对象下添加如下代码：

title: {

type: String,

value: ""

},

moreurl: {

type: String,

value: ""

},

items: {

type: Array,

value: []

},

type: {

type: String,

value: ""

}

6.组件测试

6.1打开index.json文件，在usingComponents对象下添加组件引用代码：

"indexmodule": "/components/indexmodule/indexmodule"

6.2打开index.wxml文件，添加搜索栏组件代码：

<indexmodule title="电影" items="{{movies}}" moreurl="/pages/list/list?type=movie" type="movie"></indexmodule>

6.3 打开index.js文件，为页面添加测试数据，在data对象下添加代码如下：

movies: [{

cover: {

url: "https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg"

},

rating: {

value: 6.7

},

title: "jarvis"

}, {

cover: {

url: "https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg"

},

rating: {

value: 6.7

},

title: "jarvis"

}, {

cover: {

url: "https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg"

},

rating: {

value: 6.7

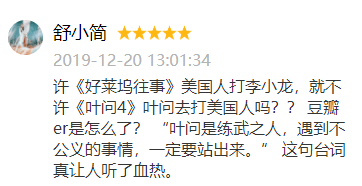
},

title: "jarvis"

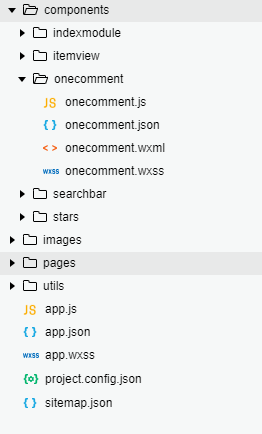
}]

## 任务2.5 短评组件实现

该视图应在多个页面多次使用，所以我们直接将它封装成自定义组件，方便调用。



1.新建components/ onecomment目录，在该目录下新建components，命名为onecomment。项目结构如下图：



2.数据模块组件的视图层渲染

2.1打开onecomment.json文件，在usingComponents对象下添加组件引用代码：

"stars": "/components/stars/stars"

2.2打开onecomment.wxml文件，添加如下代码：

<view class='comment-group'>

<view class='left-comment'>

<image class="avatar" src="{{item.user.avatar}}"></image>

</view>

<view class='right-comment'>

<view class='username-rate'>

<text class='username'>{{item.user.name}}</text>

<stars rate="{{item.rating.value\*2}}" starsize="30" istext='{{false}}'></stars>

</view>

<view class="release-time">{{item.create\_time}}</view>

<view class='content'>{{item.comment}}</view>

</view>

</view>

2.3 打开onecomment.wxss文件,为视图层添加样式文件，代码如下：

.comment-group{

display: flex;

justify-content: flex-start;

padding-top: 40rpx;

}

.comment-group .left-comment{

width: 70rpx;

margin-right: 20rpx;

}

.left-comment .avatar{

width: 70rpx;

height: 70rpx;

border-radius: 50%;

}

.comment-group .right-comment{

flex: 1;

}

.right-comment .username-rate{

display: flex;

justify-content: flex-start;

align-items: center;

}

.username-rate .username{

font-size: 36rpx;

margin-right: 20rpx;

}

.release-time{

color: #b3b3b3;

font-size: 32rpx;

margin-top: 10rpx;

}

.content{

font-size: 32rpx;

color: #353535;

margin-top: 10rpx;

}

3打开onecomment.js文件为该自定义组件添加属性值，在properties对象下添加如下代码：

item: {

type: Object,

value: {}

}

6.组件测试

6.1打开index.json文件，在usingComponents对象下添加组件引用代码：

"onecomment": "/components/onecomment/onecomment"

6.2打开index.wxml文件，添加搜索栏组件代码：

<onecomment wx:for="{{comments}}" item="{{item}}"></onecomment>

6.3 打开index.js文件，为页面添加测试数据，在data对象下添加代码如下：

comments:[

{

user:{

avatar: "https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg",

name:'jarvis'

},

rating:{

value:6.3

},

create\_time:"2019-12-11 12:32:11",

comment:"这个电影太好看了"

},

{

user: {

avatar: "https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1581654839849&di=7daafbe7d5a57121340d08c8ffd73632&imgtype=0&src=http%3A%2F%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201802%2F20%2F20180220165946\_RiGPS.thumb.700\_0.jpeg",

name: 'jarvis'

},

rating: {

value: 6.3

},

create\_time: "2019-12-11 12:32:11",

comment: "这个电影太好看了"

}

]

# 单元3 首页功能

## 任务3.1 功能实现

1.为首页添加页面渲染

2.1 打开index.json文件，在usingComponents对象下添加组件引用代码：

"searchbar": "/components/searchbar/searchbar",

"indexmodule": "/components/indexmodule/indexmodule"

2.2 为首页界面的视图层添加页面渲染，打开index.wxml文件，添加如下代码：

<searchbar isnavigator="{{true}}"></searchbar>

<!-- 电影 -->

<indexmodule title="电影" items="{{movies}}" moreurl="/pages/list/list?type=movie" type="movie"></indexmodule>

<!-- 电视剧 -->

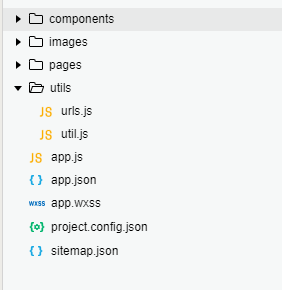
<indexmodule title="电视剧" items="{{tvs}}" moreurl="/pages/list/list?type=tv" type="tv"></indexmodule>

<!-- 综艺 -->

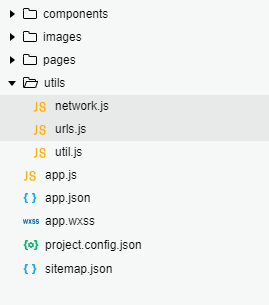
<indexmodule title="综艺" items="{{shows}}" moreurl="/pages/list/list?type=show" type="show"></indexmodule>

2. 添加工具类，方便网络请求地址集中管理及网络接口统一调用。

2.1在utils包下新建js文件，命名为urls，目录结构如下图：



2.2相同办法，新建network.js文件，目录结构如下图：



2.3 打开urls.js文件，添加首页需要使用的网络请求地址，添加如下代码：

const globalUrls = {

movieList: "https://m.douban.com/rexxar/api/v2/subject\_collection/movie\_showing/items",

tvList: "https://m.douban.com/rexxar/api/v2/subject\_collection/tv\_hot/items",

showList: "https://m.douban.com/rexxar/api/v2/subject\_collection/tv\_variety\_show/items",

}

export { globalUrls }

2.4 为首页需要的网络请求封装函数，方便请求调用。

2.4.1 在network.js文件中引入urls.js文件，在network.js文件顶部添加如下代码：

import { globalUrls } from "urls.js";

2.4.2 打开network.js文件，封装getItemList函数，传入参数分别为type（请求类型），count（请求个数）。顺序添加如下代码：

const network = {

// 获取item列表

getItemList: function (params) {

var url = "";

if (params.type === 'movie') {

url = globalUrls.movieList;

} else if (params.type === 'tv') {

url = globalUrls.tvList;

} else {

url = globalUrls.showList;

}

var count = params.count ? params.count : 7;

wx.request({

url: url,

data: {

count: count

},

success: function (res) {

var items = res.data.subject\_collection\_items;

var itemsCount = items.length;

var left = itemsCount % 3;

if (left === 2) {

items.push(null);

}

if (params && params.success) {

params.success(items);

}

}

});

}

}

export { network }

2.4.3 分别添加获取电影列表，电视剧列表，综艺节目列表封装代码，在network.js文件中顺序添加如下代码：

// 获取电影列表

getMovieList: function (params) {

params.type = "movie";

this.getItemList(params);

},

// 获取电视剧列表

getTVList: function (params) {

params.type = "tv";

this.getItemList(params);

},

// 获取综艺列表

getShowList: function (params) {

params.type = "show";

this.getItemList(params);

},

3.为首页页面添加业务逻辑代码

3.1 打开index.js文件，添加network.js文件的引用代码，在页面最顶部添加如下代码：

import {network} from "../../utils/network.js";

3.2 在生命周期函数中添加获取数据的函数，调用network中封装的函数，在onLoad函数中添加如下代码：

/\*\*

\* 生命周期函数--监听页面加载

\*/

onLoad: function (options) {

// MVC：Model,View,Controller

var that = this;

// 电影

network.getMovieList({

success: function(movies){

that.setData({

movies: movies

});

}

});

// 电视剧

network.getTVList({

success: function(tvs){

that.setData({

tvs: tvs

});

}

});

// 综艺

network.getShowList({

success: function (shows) {

that.setData({

shows: shows

});

}

});

}

# 单元4 列表页功能

## 任务4.1 功能实现

1.为列表页添加页面渲染

1.1 打开list.json文件，在usingComponents对象下添加组件引用代码：

"searchbar": "/components/searchbar/searchbar",

"itemview": "/components/itemview/itemview"

1.2 为列表页的视图层添加页面渲染，打开list.wxml文件，添加如下代码：

<searchbar isnavigator="{{true}}"></searchbar>

<view class='container'>

<itemview wx:for="{{items}}" wx:key="{{item.title}}" item="{{item}}" itemurl="/pages/detail/detail?type={{type}}&id={{item.id}}"></itemview>

</view>

1.3 为页面添加样式文件，打开打开list.wxss文件，添加如下代码:

.container{

display: flex;

justify-content: space-between;

flex-wrap: wrap;

padding: 30rpx;

}

2.为首页页面添加业务逻辑代码

2.1 打开list.js文件，添加network.js文件的引用代码，在页面最顶部添加如下代码：

import {network} from "../../utils/network.js";

2.2 在生命周期函数中添加获取数据的函数，根据页面跳转type类型，调用network中封装的函数，在onLoad函数中添加如下代码：

var that = this;

var type = options.type;

that.setData({

type:type

});

var title = "";

wx.showLoading({

title: '正在加载中...',

})

if(type === "movie"){

// 请求电影的数据

network.getMovieList({

success: function (items) {

that.setData({

items: items

});

wx.hideLoading();

},

count: 1000

});

title = "电影";

}else if(type === 'tv'){

// 请求电视剧的数据

network.getTVList({

success: function (items) {

that.setData({

items: items

});

wx.hideLoading();

},

count: 1000

});

title = "电视剧";

}else{

// 请求综艺的数据

network.getShowList({

success: function (items) {

that.setData({

items: items

});

wx.hideLoading();

},

count: 1000

});

title = "综艺";

}

wx.setNavigationBarTitle({

title: title,

})

# 单元5 搜索功能

## 任务5.1 功能实现

1.为搜索页添加页面渲染

2.1 打开search.json文件，在usingComponents对象下添加组件引用代码：

"searchbar": "/components/searchbar/searchbar"

2.2 为搜索界面的视图层添加页面渲染，打开search.wxml文件，添加如下代码：

<searchbar bindsearchinput="onSearchInputEvent"></searchbar>

<view class='history-list-group' wx:if="{{histories && !subjects}}">

<view class='history-title'>

<view class='title'>历史记录</view>

<view class='clear' bindtap="onClearEvent">清除</view>

</view>

<navigator wx:for="{{histories}}" wx:key="{{item.id}}" url='/pages/detail/detail?type={{item.type}}&id={{item.id}}' class='history-group'>{{item.title}}</navigator>

</view>

<view class='item-list-group'>

<view wx:for="{{subjects}}" class='item-group' wx:key="{{item.id}}" bindtap="onItemTapEvent" data-id="{{item.id}}" data-title="{{item.title}}" data-type="{{item.type}}">

<image src='{{item.pic.normal}}' class='thumbnail'></image>

<view class='info-group'>

<view class='title'>{{item.title}}</view>

<view class='rate-year'>{{item.rating.value}}分/{{item.year}}</view>

</view>

</view>

</view>

1.3 为页面添加样式文件，打开打开search.wxss文件，添加如下代码:

.item-list-group{

padding: 10rpx 20rpx;

}

.item-list-group .item-group{

padding: 10rpx 0;

border-bottom: 1px solid #e4e4e4;

display: flex;

}

.item-group .thumbnail{

width: 80rpx;

height: 100rpx;

margin-right: 20rpx;

}

.item-group .info-group{

flex: 1;

display: flex;

flex-direction: column;

justify-content: space-between;

}

.info-group .title{

font-size: 32rpx;

}

.info-group .rate-year{

font-size: 28rpx;

color: #7b7b7b;

}

.history-list-group{

padding: 10rpx 20rpx;

}

.history-list-group .history-title{

display: flex;

justify-content: space-between;

padding: 20rpx 0;

background: #f9f9f9;

font-size: 28rpx;

color: #9e9e9e;

}

.history-list-group .history-group{

font-size: 32rpx;

padding: 20rpx 0;

border-bottom: 1px solid #e4e4e4;

}

2. 修改工具类，方便网络请求地址集中管理及网络接口统一调用。

2.1 打开urls.js文件，添加搜索页需要的网络请求地址searchUrl函数，传入参数为搜索信息内容。顺序添加如下代码：

searchUrl: function (q) {

return "https://m.douban.com/rexxar/api/v2/search? q=" + q

}

2.2 打开network.js文件，封装getSearch函数，传入参数为搜索信息内容。顺序添加如下代码：

// 搜索item

getSearch: function(params){

var q = params.q;

var url = globalUrls.searchUrl(q);

wx.request({

url: url,

success: function(res){

var subjects = res.data.subjects;

if(params.success){

params.success(subjects);

}

}

})

}

3.为搜索页面添加业务逻辑代码

3.1 打开search.js文件，添加network.js文件的引用代码，在页面最顶部添加如下代码：

import {network} from "../../utils/network.js";

3.2在onLoad生命周期函数中，获取搜索内容缓存数据，添加如下代码：

onLoad: function (options) {

var that = this;

wx.getStorage({

key: 'searched',

success: function(res) {

var data = res.data;

that.setData({

histories: data

})

},

})

},

3.3 为输入框添加数据变化监听onSearchInputEvent事件，实时请求网络数据并且将搜索内容展示在页面中，添加如下代码：

onSearchInputEvent: function(event){

var that = this;

var value = event.detail.value;

if(!value || value === ""){

that.setData({

subjects: null

});

return;

}

network.getSearch({

q: value,

success: function(subjects){

that.setData({

subjects: subjects

})

}

})

},

3.4 为搜索内容项添加单击onItemTapEvent事件，判断该项是否已存在于缓存中，如不存在，则将该条记录存入缓存中。添加如下代码：

onItemTapEvent: function(event){

var that = this;

var id = event.currentTarget.dataset.id;

var type = event.currentTarget.dataset.type;

var title = event.currentTarget.dataset.title;

var histories = that.data.histories;

var isExisted = false;

if(histories){

for(var index=0;index<histories.length;index++){

var movie = histories[index];

if(movie.id === id){

isExisted = true;

break;

}

}

}

if(!isExisted){

if(!histories){

histories = [];

}

histories.push({ title: title, id: id, type: type});

wx.setStorage({

key: 'searched',

data: histories,

success: function () {

console.log("保存成功！");

}

})

}

wx.navigateTo({

url: "/pages/detail/detail?type=" + type+"&id="+id,

});

},

3.5 为清除按钮添加onClearEvent事件，移除缓存中的内容，添加如下代码：

onClearEvent: function(event){

wx.removeStorage({

key: 'searched',

success: function(res) {

console.log("删除成功！");

},

});

this.setData({

histories: null

});

}

# 单元6 详情页功能

## 任务6.1 基础信息展示

1.为详情页添加页面渲染

2.1 打开detail.json文件，在usingComponents对象下添加组件引用代码：

"stars": "/components/stars/stars"

2.2 为详情页的视图层添加基础信息展示部分的渲染，打开detail.wxml文件，添加如下代码：

<view class="item-header">

<view class="item-title">{{item.title}} {{item.original\_title}}({{item.year}})</view>

<view class='item-detail'>

<view class="left-detail">

<view class="item-rate">

<stars rate="{{item.rating.value}}" starsize="30" fontsize="30" fontcolor="#595959"></stars>

<text class="comment-persons">{{item.rating.value}}人评价</text>

</view>

<view class="item-sub-detail">

<view class="item-type">

{{item.durations[0]}} {{item.genres}}

</view>

<view class="item-show">

{{item.pubdate[0]}}上映 {{item.countries[0]}}

</view>

<view class="item-actors">

{{item.authors}}

</view>

</view>

</view>

<view class='right-detail'>

<image src="{{item.cover.image.small.url}}"></image>

</view>

</view>

</view>

1.3 为页面添加样式文件，打开打开detail.wxss文件，添加如下代码:

.item-header{

padding: 60rpx 30rpx;

}

.item-header .item-title{

font-size: 50rpx;

}

.item-header .item-detail{

display: flex;

justify-content: space-between;

margin-top: 20rpx;

}

.item-detail .left-detail{

flex: 1;

margin-right: 20rpx;

}

.left-detail .item-rate{

display: flex;

justify-content: flex-start;

align-items: center;

}

.item-rate .comment-persons{

font-size: 28rpx;

color: #ccc;

margin-left: 20rpx;

}

.item-detail .right-detail{

width: 200rpx;

height: 300rpx;

}

.right-detail image{

width: 100%;

height: 100%;

}

.item-sub-detail{

margin-top: 40rpx;

font-size: 32rpx;

}

.item-sub-detail view{

margin-bottom: 10rpx;

}

2. 修改工具类，方便网络请求地址集中管理及网络接口统一调用。

2.1 打开urls.js文件，添加详情页需要的网络请求地址。顺序添加如下代码：

movieDetail: "https://m.douban.com/rexxar/api/v2/movie/",

tvDetail: "https://m.douban.com/rexxar/api/v2/tv/",

showDetail: "https://m.douban.com/rexxar/api/v2/tv/",

2.2 打开network.js文件，封装getItemDetail函数，传入参数为模块类型和影视数据id。顺序添加如下代码：

// 获取详情

getItemDetail: function(params){

var type = params.type;

var id = params.id;

var url = "";

if(type === 'movie'){

url = globalUrls.movieDetail + id;

}else if(type === 'tv'){

url = globalUrls.tvDetail + id;

}else{

url = globalUrls.showDetail + id;

}

wx.request({

url: url,

success: function(res){

var item = res.data;

if(params.success){

params.success(item);

}

}

})

},

3.为详情页页面添加业务逻辑代码

3.1 打开detail.js文件，添加network.js文件的引用代码，在页面最顶部添加如下代码：

import {network} from "../../utils/network.js";

3.2在onLoad生命周期函数中，获取详情页基础信息内容数据，根据页面展示要求拼接数据，添加如下代码：

onLoad: function (options) {

console.log(options)

var that = this;

var type = options.type;

var id = options.id;

that.setData({

id: id,

type: type

});

network.getItemDetail({

type: type,

id: id,

success: function(item){

console.log(item)

var genres = item.genres;

genres = genres.join("/");

item.genres = genres;

var actors = item.actors;

var actorNames = [];

if(actors.length > 3){

actors = actors.slice(0,3);

}

for(var index=0; index<actors.length; index++){

var actor = actors[index];

actorNames.push(actor.name);

}

var authors=""

actorNames = actorNames.join("/");

if(type=="show"){

authors = actorNames;

}else{

var director = item.directors[0].name;

authors = director + "(导演) /" + actorNames;

}

item.authors = authors;

that.setData({

item: item

});

}

});

}

## 任务6.2 标签信息展示

1.为详情页添加页面渲染

1.1 为详情页的视图层添加标签信息展示部分的渲染，打开detail.wxml文件，顺序添加如下代码：

<view class='item-tags'>

<view class='item-tags-title'>豆瓣成员常用标签</view>

<view class='item-tags-list'>

<text wx:for="{{tags}}" wx:key="\*this">{{item}}</text>

</view>

</view>

1.2 为页面添加样式文件，打开打开detail.wxss文件，添加如下代码:

.item-tags{

padding: 0 30rpx;

}

.item-tags .item-tags-title{

color: #b3b3b3;

font-size: 32rpx;

margin-bottom: 20rpx;

}

.item-tags .item-tags-list{

display: flex;

justify-content: flex-start;

flex-wrap: wrap;

}

.item-tags-list text{

padding: 10rpx 20rpx;

background: #f5f5f5;

font-size: 32rpx;

color: #353535;

text-align: center;

border-radius: 40rpx;

margin-right: 20rpx;

margin-bottom: 20rpx;

}

2. 修改工具类，方便网络请求地址集中管理及网络接口统一调用。

2.1 打开urls.js文件，添加获取标签需要的网络请求地址函数，传入参数为视频数据id。顺序添加如下代码：

movieTags: function(id){

return "https://m.douban.com/rexxar/api/v2/movie/"+ id +"/tags?count=8"

},

tvTags: function(id){

return "https://m.douban.com/rexxar/api/v2/tv/"+id+"/tags?count=8"

},

showTags: function(id){

return this.tvTags(id);

},

2.2 打开network.js文件，封装获取标签信息的getItemTags函数，传入参数为模块类型和影视数据id。顺序添加如下代码：

// 获取标签

getItemTags: function(params){

var type = params.type;

var id = params.id;

var url = "";

if(type === 'movie'){

url = globalUrls.movieTags(id);

}else if(type === 'tv'){

url = globalUrls.tvTags(id);

}else{

url = globalUrls.showTags(id);

}

wx.request({

url: url,

success: function(res){

var tags = res.data.tags;

if(params.success){

params.success(tags);

}

}

})

},

3.为详情页页面添加业务逻辑代码

3.1在onLoad生命周期函数中，获取详情页标签内容数据，顺序添加如下代码：

network.getItemTags({

type: type,

id: id,

success: function(tags){

that.setData({

tags: tags

});

}

});

## 任务6.3 短评信息展示

1. 为详情页添加页面渲染

1.1 打开detail.json文件，在usingComponents对象下添加组件引用代码：

"onecomment": "/components/onecomment/onecomment"

1.2 为详情页的视图层添加短评信息展示部分的渲染，打开detail.wxml文件，顺序添加如下代码：

<view class='comment-list-group'>

<view class='comment-title'>短评({{totalComment}})</view>

<onecomment wx:for="{{comments}}" item="{{item}}"></onecomment>

</view>

<navigator class='more-comment' url='/pages/comment/comment?id={{id}}&type={{type}}&thumbnail={{item.cover.image.small.url}}&title={{item.title}}&rate={{item.rating.value}}'>查看更多短评</navigator>

1.3 为页面添加样式文件，打开打开detail.wxss文件，添加如下代码:

.comment-list-group{

padding: 60rpx 30rpx;

}

.comment-list-group .comment-title{

font-size: 32rpx;

color: #b3b3b3;

}

.more-comment{

text-align: center;

font-size: 36rpx;

color: #41be57;

margin-bottom: 60rpx;

}

2. 修改工具类，方便网络请求地址集中管理及网络接口统一调用。

2.1 打开urls.js文件，添加获取短评需要的网络请求地址函数，传入参数为视频数据id，请求个数count和请求开始点start。顺序添加如下代码：

movieComments: function(id,start=0,count=3){

return 'https://m.douban.com/rexxar/api/v2/movie/' + id + '/interests?count=' + count + '&start=' + start;

},

tvComments: function(id,start=0,count=3){

return 'https://m.douban.com/rexxar/api/v2/tv/' + id + '/interests?count=' + count + '&start=' + start;

},

showComments: function(id,start=0,count=3){

return this.tvComments(id,start,count);

},

2.2 打开network.js文件，封装获取标签信息的getItemTags函数，传入参数为模块类型type，视频数据id，请求个数count和请求开始点start。顺序添加如下代码：

// 获取短评

getItemComments: function(params){

var type = params.type;

var id = params.id;

var start = params.start?params.start:0;

var count = params.count?params.count:3;

var url = "";

if(type === 'movie'){

url = globalUrls.movieComments(id,start,count);

}else if(type === 'tv'){

url = globalUrls.tvComments(id,start,count);

}else{

url = globalUrls.showComments(id,start,count);

}

wx.request({

url: url,

success: function(res){

var data = res.data;

if(params.success){

params.success(data);

}

}

})

},

3.为详情页页面添加业务逻辑代码

3.1在onLoad生命周期函数中，获取详情页标签内容数据，顺序添加如下代码：

// 获取评论

network.getItemComments({

type: type,

id: id,

success: function(data){

var totalComment = data.total;

var comments = data.interests;

that.setData({

totalComment: totalComment,

comments: comments

});

}

});

# 单元7 短评列表功能

## 任务7.1 功能实现

1.为短评列表页添加页面渲染

2.1 打开comment.json文件，在usingComponents对象下添加组件引用代码：

"onecomment": "/components/onecomment/onecomment"

2.2 为短评列表页面的视图层添加页面渲染，打开comment.wxml文件，添加如下代码：

<view class="container">

<view class="item-group" bindtap="onItemTapEvent">

<image class="thumbnail" src='{{thumbnail}}'></image>

<text class='item-title'>{{title}}</text>

<text class='item-rate'>{{rate}}分</text>

</view>

<view class='comment-title'>全部影评({{total}})</view>

<onecomment wx:for="{{comments}}" item="{{item}}"></onecomment>

<view class='page-btn-group'>

<button class='page-btn' bindtap='onPrePageTap' disabled='{{start <= 1}}' loading="{{preLoading}}">上一页</button>

<button class='page-btn' bindtap='onNextPageTap' loading='{{nextLoading}}'>下一页</button>

</view>

</view>

1.3 为页面添加样式文件，打开打开comment.wxss文件，添加如下代码:

.container{

padding: 20rpx 30rpx;

}

.item-group{

display: flex;

justify-content: flex-start;

align-items: center;

}

.item-group .thumbnail{

width: 40rpx;

height: 50rpx;

}

.item-group .item-title{

font-size: 32rpx;

color: #41be57;

margin-left: 10rpx;

margin-right: 10rpx;

}

.item-group .item-rate{

font-size: 28rpx;

color: #ccc;

}

.comment-title{

margin-top: 60rpx;

font-size: 40rpx;

}

.page-btn-group{

margin-top: 40rpx;

margin-bottom: 40rpx;

display: flex;

justify-content: flex-start;

align-items: center;

}

.page-btn{

flex: 1;

height: 60rpx;

color: #898989;

border-color: #898989;

line-height: 60rpx;

}

2.为搜索页面添加业务逻辑代码

2.1 打开comment.js文件，添加network.js文件的引用代码，在页面最顶部添加如下代码：

import {network} from "../../utils/network.js";

3.2 为页面添加临时变量，分别为短评总量total、短评请求开始点start、单次请求个数count，在data字段中添加如下代码：

total: 0,

start: 1,

count: 20

3.3 封装getComments函数，获取当前页页面数据，传入短评请求开始点start数据。

getComments: function(start){

var that = this;

var type = that.data.type;

var id = that.data.id;

if(start > that.data.start){

that.setData({

nextLoading: true

});

}else{

that.setData({

preLoading: true

});

}

network.getItemComments({

type: type,

id: id,

start: start,

count: 20,

success: function (data) {

var total = data.total;

var comments = data.interests;

console.log(comments);

that.setData({

total: total,

comments: comments,

start: start,

preLoading: false,

nextLoading: false

});

wx.pageScrollTo({

scrollTop: 0,

})

}

})

},

3.4在onLoad生命周期函数中，获取首页展示短评数据，获取上一个页面传递数据，用来展示页面首行数据。添加如下代码：

onLoad: function (options) {

var that = this;

that.setData(options);

that.getComments(1);

},

3.5为页面首行数据添加onItemTapEvent事件，效果未返回上一页，添加如下代码：

onItemTapEvent: function(event){

wx.navigateBack({});

},

3.6 为上一页按钮添加单击onPrePageTap事件，如果现有起始点大于20，则代表非第一页数据，则可以点击上一页。添加如下代码：

onPrePageTap: function(event){

var that = this;

var oldStart = that.data.start;

var count = that.data.count;

if(oldStart - count > 0){

var start = oldStart - count;

that.getComments(start);

}

},

3.7 为下一页按钮添加单击onNextPageTap事件，起始点为上一页起始点加上20，添加如下代码：

onNextPageTap: function(event){

var that = this;

var oldStart = that.data.start;

var start = oldStart + that.data.count;

that.getComments(start);

}