31 搜索热词、搜索历史和联想搜索

更新时间：2019-08-30 11:36:32

> 不想当将军的士兵，不是好士兵。
>
> ——拿破仑

上一节我们在商城首页实现了分类展示商品列表，本节我们将在商品搜索页面实现商品搜索功能。

在第二节中，我们已经设计了商品搜索页面的页面效果，如图 11 所示。

图 11 商品搜索页面

根据第一节的业务设计，和第二节的功能设计、页面效果图，按照“分类拆解法”的步骤进行商品搜索页面开发。

1. 拆解步骤

在本节仅给出拆解结果，拆解过程请回顾第二章第三节对“分类拆解法”的详细讲解内容。

请各位同学自己动手实践，按照拆解步骤拆解，结合第一节的业务设计、第二节的功能设计，拆解本节图 11、第二节图 2、第二节图 4 的商品搜索页面，然后将自己的拆解结果与本节列出的拆解结果进行对照总结。这是本节内容的实践环节之一。

商品搜索页面可以由上到下拆解为 4 个子部件，具体拆解结果如下：

子部件 1 ：顶部搜索栏

子部件 1 的显示元素包括：

搜索框

输入界面，事件为键盘输入事件、输入框聚焦事件、输入框失去焦点事件，数据为用户输入关键词和输入框是否聚焦的标志

键盘输入事件的事件响应为：记录用户输入关键词并在联想词列表中显示与输入内容相关的关键词

输入框聚焦事件的事件响应为：设置输入框是否聚焦的标志为 `true`，显示联想词列表，并在联想词列表中显示与输入框中内容相关的关键词

输入框失去焦点事件的事件响应为：设置输入框是否聚焦的标志为 `false`

清空搜索框内容按钮

单条内容交互界面，事件为用户点击事件，数据为用户输入关键词

清空搜索框内容按钮仅在输入框中有内容时显示

用户点击事件的事件响应为：清空用户输入关键词和联想关键词数组

联想词列表

多条内容交互界面，事件为用户点击事件，数据为用户输入关键词和联想关键词数组

当用户在搜索框输入内容时显示联想词列表，联想关键词数组为搜索联想词库中包含用户输入关键词的 8 条联想关键词 **（与用户输入的关键词相关的联想词可能有很多个，显示太多的联想词会让用户无法选择，只显示最相关的几个联想词的用户体验较好）**

用户点击事件的事件响应为：在搜索框中设置搜索内容为用户点击的关键词，并触发搜索按钮点击事件显示搜索结果

搜索按钮

静态界面，事件为用户点击事件，数据为用户输入关键词、搜索结果列表和历史搜索关键词列表

用户点击事件的事件响应为：在历史搜索关键词列表中添加当前用户输入关键词，并在子部件 4 中显示商品标题包含用户输入关键词的搜索结果列表

子部件 2 ：历史搜索记录栏

子部件 1 的显示元素包括：

标题

静态界面，无事件，无数据

清空按钮

静态界面，事件为用户点击事件，数据为历史搜索关键词列表

历史搜索关键词列表存储在微信客户端缓存中

用户点击事件的事件响应为：在微信客户端缓存中清空历史搜索关键词列表

标签栏

多条内容交互界面，事件为页面加载事件与用户点击事件，数据为历史搜索关键词列表

在页面加载时，从微信客户端缓存中读取历史搜索关键词列表，显示到标签栏中 **（使用微信客户端缓存保存历史搜索关键词列表才能实现：当用户第二次打开商品搜索页面时，能看到以前在商品搜索页页面搜索的历史记录）**

用户点击事件的事件响应为：在搜索框中设置搜索内容为用户点击的标签，并触发搜索按钮点击事件显示搜索结果

子部件 3 ：热门搜索词栏

子部件 3 的显示元素包括：

标题

静态界面，无事件，无数据

标签栏

多条内容交互界面，事件为页面加载事件与用户点击事件，数据为热门搜索关键词列表

在页面加载时，从热搜词库中读取热门搜索关键词列表，显示到标签栏中

用户点击事件的事件响应为：在搜索框中设置搜索内容为用户点击的标签，并触发搜索按钮点击事件显示搜索结果

子部件 4 ：搜索结果栏

子部件 4 的显示元素包括：

商品列表

多条数据的交互界面，事件为用户屏幕滑动事件与用户点击屏幕事件，数据为：商品标题包含用户输入关键词的商品信息数组

点击搜索按钮、点击搜索联想词、点击历史搜索记录标签或点击热搜词标签后，从商品信息表中查找商品标题包含关键词的商品信息列表

用户下滑屏幕到页面底部事件的事件响应为：从商品信息表中分页获取商品标题包含关键词的下一页商品信息数据，并追加到商品信息数组末尾。如果商品信息表中商品标题包含关键词的所有数据都获取完毕，用户下滑屏幕将不再获取分页数据

商品信息数组中的商品数据以左右两列卡片式瀑布流的方式显示，一个卡片显示一个商品信息，每个卡片显示商品的缩略图、商品名称（黑色）、商品简介（灰色）、商品原价（红色）

用户点击屏幕事件的事件响应为：在用户点击商品卡片后，页面跳转到商品详情页面，需要向商品详情页面传递商品 ID

商品信息表的数据需要从云数据库中获取

2. 数据服务

在商品搜索页面需要读取搜索联想词库与热搜词库，读取方法应该在数据服务 `ProductService` 中实现，将业务与数据操作分离：

```
/**
 * 产品数据操作类
 */
class ProductService {

  constructor() {
    this.productData = require("../data/json_data.js")
  }

  /**
   * 从热搜词库获取热搜产品关键词
   * @return {array} 热搜产品关键词string数组
   */
  getHotSearchProducts() {
    return this.productData.hotSearchProducts
  }

  /**
   * 从搜索联想词库获取搜索联想词
   * @param {string} keyword 用户输入的关键词
   * @return {array} 联想词string数组
   */
  getSuggestWords(keyword) {
    keyword = keyword.toLowerCase()
    //将用户输入的关键词与联想词库进行小写匹配，返回 8 个联想词
    return this.productData.suggestWords.filter(e => e.word.toLowerCase().indexOf(keyword) >= 0).slice(0, 8)
  }

}

export default ProductService
```

ProductService 的完整代码请参考专栏源代码 ProductService.js ，具体代码位置见本节末尾图 12。

在页面 JS 逻辑中引用 ProductService 即可调用读取搜索联想词库与热搜词库的方法：

```
//引用商品信息的数据库访问类
import ProductService from '../../../../../dataservice/ProductService.js'
var productService = new ProductService()
```

3. 编程步骤

在本章第二节的图 2 、图 3 、图 4 中分别展示了不同情况下各个显示元素的显示与隐藏，这主要与输入框是否聚焦的标志 inputFocused 和输入框中是否有内容（即用户输入关键词的字符串长度 inputVal.length 是否大于零）有关。

各个子部件与显示元素的具体显示与隐藏规则如下：

输入框中有内容时显示：

子部件 1 的清空搜索框内容按钮

输入框中无内容时显示：

子部件 2 ：历史搜索记录栏

子部件 3 ：热门搜索词栏

输入框中有内容，且输入框聚焦时显示：

子部件 1 的联想词列表

输入框中有内容，且输入框未聚焦时显示：

子部件 4 ：搜索结果栏

3.1 定义页面子部件及其排列顺序

首先在 data 中定义与界面显示隐藏相关的数据：

```
data: {
  inputVal: "", //用户输入关键词
  inputFocused: false, //输入框是否聚焦的标志
},
```

然后在 WXML 页面模板中定义 3 个子部件的容器，并定义显示隐藏的条件：

```
<!-- 子部件 1 ：顶部搜索栏 可直接使用 WeUI 的 SearchBar 组件实现-->
<view class="weui-search-bar">
</view>
<!-- 子部件 1 的联想词列表 当输入框中有内容，且输入框聚焦时显示 -->
<view class="weui-cells searchbar-result" hidden="{{inputVal.length <= 0 || !inputFocused}}">
</view>

<!-- 输入框中无内容时显示 -->
<view hidden="{{inputVal.length > 0}}">
  <!-- 子部件 2 ： 历史搜索记录栏 -->
  <view class="weui-panel padding">
  </view>
  <!-- 子部件 3 ： 热门搜索词栏 -->
  <view class="weui-panel padding">
  </view>
</view>

<!-- 子部件 4 ： 搜索结果栏 当输入框中有内容，且输入框未聚焦时显示-->
<view hidden="{{inputVal.length <= 0 || inputFocused}}">
</view>
```

子部件 1 可直接使用 WeUI 的 SearchBar 组件实现。

需要注意的 wx:if 与 hidden 的不同使用场景， wx:if 与 hidden 都可以控制界面的显示与隐藏，但他们的性能开销不同，在需要频繁切换显示隐藏的场景下，用 hidden 更好，否则 wx:if 较好。

wx:if 与 hidden 的详细区别请参考微信官方 “小程序开发文档” 中的说明，具体位置为：“框架” -> “WXML 语法参考” -> “条件渲染”。

3.2 实现子部件 1 ：顶部搜索栏

顶部搜索栏可以直接修改 WeUI Demo 小程序的 SearchBar 组件源代码实现，需要进行修改的内容包括：

1. 在输入框增加输入框聚焦事件 bindfocus="inputFocus"、输入框失去焦点事件 bindblur="inputBlur" 以编写事件响应函数
2. 使用 hidden 来控制界面隐藏、显示
3. 使用列表渲染 wx:for 来展示联想词列表，并定义联想词点击事件 bindtap='onClickTags' 以编写事件响应函数
4. 将搜索框右侧的取消按钮修改为搜索按钮，并定义搜索按钮点击事件 bindtap="onClickSearchButton" 以编写事件响应函数

```
<!-- 子部件 1 ：顶部搜索栏 可直接使用 WeUI 的 SearchBar 组件实现 -->
<view class="weui-search-bar">
 <view class="weui-search-bar__form">
  <view class="weui-search-bar__box">
   <icon class="weui-icon-search_in-box" type="search" size="14"></icon>
   <!-- 1.在输入框增加输入框聚焦事件、输入框失去焦点事件 -->
   <input type="text" class="weui-search-bar__input" placeholder="搜索" value="{{inputVal}}" bindfocus="inputFocus" bindblur="inputBlur" bindinput="in
putTyping" />
   <!-- 2.使用 hidden 来控制界面隐藏、显示 -->
   <view class="weui-colorui-clear" hidden="{{inputVal.length <= 0}}" bindtap="clearInput">
    <icon type="clear" size="14"></icon>
   </view>
  </view>
 </view>
 <!-- 4.在搜索框右侧的取消按钮修改为搜索按钮 -->
 <view class="weui-search-bar__cancel-btn" bindtap="onClickSearchButton">搜索</view>
</view>
<!-- 2.子部件 1 的联想词列表 当输入框中有内容，且输入框聚焦时显示 使用 hidden 来控制界面隐藏、显示 -->
<view class="weui-cells searchbar-result" hidden="{{inputVal.length <= 0 || !inputFocused}}">
 <!-- 3.使用 列表渲染来展示联想词列表 -->
 <block wx:for="{{suggestKeywords}}" wx:key="{{suggestKeyword.word}}" wx:for-item="suggestKeyword">
  <view class="weui-cell" hover-class="weui-cell_active" id="{{ suggestKeyword.word }}" bindtap='onClickTags'>
   <view>{{suggestKeyword.word}}</view>
  </view>
 </block>
</view>
```

WeUI 的 SearchBar 组件是灰色背景白色搜索框，要实现图 11 中的白色背景灰色搜索框效果，还需要对样式进行调整，调整后的样式请参考专栏源代码 search.wxss 中注释为 “/* 修改 weui 搜索栏样式 */” 的部分。

在 data 中定义相关数据并实现各个事件响应函数

数据：

```
data: {
  searchedKeywords: [], //历史搜索关键词列表
  suggestKeywords: [], //联想关键词数组
},
```

根据拆解结果，逐一实现子部件 1 的各个事件响应函数：

```
 /**
  * 输入框聚焦事件：设置输入框是否聚焦的标志为 true，根据输入框中的关键词刷新联想关键词数组
  */
 inputFocus: function() {
   this.setData({
     inputFocused: true
   });
   this.refreshSuggestWords()
 },

 /**
  * 输入框失去焦点事件：设置输入框是否聚焦的标志为 false
  */
 inputBlur: function() {
   this.setData({
     inputFocused: false
   });
 },

 /**
  * 清空搜索框内容按钮点击事件：清空用户输入关键词和联想关键词数组
  */
 clearInput: function() {
   this.setData({
     inputVal: "",
     suggestKeywords: [],
```

```
  });
 },

 /**
  * 输入框键盘输入事件：记录用户输入关键词，根据输入框中的关键词刷新联想关键词数组
  */
 inputTyping: function(e) {
  this.setData({
   inputVal: e.detail.value,
  });
  this.refreshSuggestWords()
 },

 /**
  * 搜索按钮点击事件：在历史搜索关键词列表中添加当前用户输入关键词，
  * 并在子部件 4 中显示商品标题包含用户输入关键词的搜索结果列表
  */
 onClickSearchButton: function() {
  //在历史搜索关键词列表中添加当前用户输入关键词
  this.addSearchedKeyword(this.data.inputVal)

  //在用户点击搜索后，刷新商品列表显示
  //... 略，参考第三节 refreshProductList 方法，或阅读专栏源代码
 },

 /**
  * 联想词列表中联想词点击事件：设置输入框内容为点击关键词，并触发搜索按钮点击事件
  * 点击历史记录的标签、点击热门搜索标签的事件响应函数同样为该方法
  */
 onClickTags: function(e) {
  this.setData({
   inputVal: e.currentTarget.id,
  });
  this.onClickSearchButton()
 },

 /**
  * 根据输入框中的关键词刷新联想关键词数组
  */
 refreshSuggestWords: function() {
  //调用数据服务，从搜索联想词库获取搜索联想词
  var suggestWords = productService.getSuggestWords(this.data.inputVal)
  this.setData({
   suggestKeywords: suggestWords
  });
 },

 /**
  * 在历史搜索关键词列表中添加关键词
  * 使用小程序官方提供的数据缓存 API 实现历史搜索关键词列表的读取与存储
  */
 addSearchedKeyword: function(text) {
  //输入参数的合法性校验
  if (typeof(text) == "undefined" || text.length == 0) {
   return;
  }
  //从缓存读取历史搜索关键词列表
  var value = wx.getStorageSync('searchedKeywords');
  if (value) {
   //如果缓存中已经存在历史搜索关键词列表
   if (value.indexOf(text) < 0) {
    //如果历史搜索关键词列表中没有关键词，就将关键词添加到列表末尾
    value.unshift(text);
   }
  } else {
   //如果缓存中不存在历史搜索关键词列表，新建列表并添加关键词进列表
   value = [];
   value.push(text);
  }
  //更新历史搜索关键词列表的显示数据
  this.setData({
   searchedKeywords: value
  });
```

```
    //将最新的历史搜索关键词列表更新到本地缓存
    wx.setStorage({
      key: "searchedKeywords",
      data: value,
    })
  },
```

历史搜索关键词列表的存储与读取使用使用小程序官方提供的数据缓存 API 实现，数据缓存 API 的官方文档位置为：“API” -> “数据缓存”，使用 `setStorage` 向微信客户端写入搜索记录，`getStorage` 从微信客户端读取搜索记录，`removeStorage` 从缓存中清除历史搜索关键词列表。

请注意 `clearStorage` 是清空微信客户端的所有缓存，不要将它与 `removeStorage` 的用法搞反了。

3.3 实现子部件 2：历史搜索记录栏与子部件 3：热门搜索词栏

子部件 2 与 子部件 3 的实现基本一致，区别仅在于数据源不同，以及子部件 2 多了一个清空按钮。

子部件 3 的显示数据需要定义：

```
  data: {
    hotSearchProducts: [], //热门搜索关键词列表
  },
```

子部件 2 的界面需要使用 WeUI 的 Flex 组件来实现标题与清空按钮的水平左右显示。

历史搜索记录标签，热门搜索词标签使用 WXML 的列表渲染显示。

```
<!-- 输入框中无内容时显示 -->
<view hidden="{{inputVal.length > 0}}">
  <!-- 子部件 2：历史搜索记录栏 使用 WeUI 的 Panel 组件-->
  <view class="weui-panel padding">
    <view class="weui-panel__bd">
      <!-- 使用 WeUI 的 Flex 组件实现 标题与清空按钮左右显示 -->
      <view class="weui-flex">
        <view class="weui-flex__item padding-bottom-sm">
          <view class="text-lg">历史记录</view>
        </view>
        <!-- 清空按钮 -->
        <view class="weui-flex__item text-right">
          <text class="icon-delete" bindtap='clearSearchedKeywords'></text>
        </view>
      </view>
      <!-- 使用列表渲染显示历史搜索记录标签 -->
      <view class="search_tags text-df">
        <block wx:for="{{searchedKeywords}}" wx:key="{{hotSearchProducts}}" wx:for-item="keyword">
          <view id="{{ keyword }}" class="fl padding-right-sm" bindtap='onClickTags'>{{ keyword }}</view>
        </block>
      </view>
    </view>
  </view>
  <!-- 子部件 3：热门搜索词栏 使用 WeUI 的 Panel 组件-->
  <view class="weui-panel padding">
    <view class="weui-panel__bd">
      <view class="text-lg padding-bottom-sm">热门搜索</view>
      <!-- 使用列表渲染显示热门搜索词标签 -->
      <view class="search_tags text-df">
        <block wx:for="{{hotSearchProducts}}" wx:key="{{hotSearchProducts}}" wx:for-item="keyword">
          <view id="{{ keyword }}" class="fl padding-right-sm" bindtap='onClickTags'>{{ keyword }}</view>
        </block>
      </view>
    </view>
  </view>
</view>
```

在页面加载时需要读取历史搜索记录列表与热门搜索关键词列表：

```
onReady: function() {
  //调用数据服务读取热门搜索关键词列表
  this.setData({
    hotSearchProducts: productService.getHotSearchProducts()
  })
  //从本地缓存读取历史搜索记录列表
  var value = wx.getStorageSync('searchedKeywords');
  if (value) {
    //如果缓存中已经存在历史搜索关键词列表，读取缓存
    this.setData({
      searchedKeywords: value
    });
  }
},
```

历史搜索记录标签与热门搜索词标签的点击事件与联想词点击事件一致，可直接使用同样的的事件响应函数。

子部件 2 的清空按钮需要定义事件响应函数：

```
/**
 * 清空按钮点击事件：在微信客户端缓存中清空历史搜索关键词列表
 */
clearSearchedKeywords: function() {
  //清空历史搜索关键词列表的显示数据
  this.setData({
    searchedKeywords: []
  });
  //缓存中清除历史搜索关键词列表
  wx.removeStorage({
    key: 'searchedKeywords',
  })
},
```

3.4 实现子部件 4：搜索结果栏

搜索结果栏的代码实现与第三节商品列表栏几乎相同，不再复述。

搜索结果栏的实现代码请参考：

- 第三节中商品列表栏的代码，不使用自定义组件，直接在页面中编写全部代码
- 专栏源代码，使用自定义组件实现

由于篇幅所限，包含完整样式的 WXML 页面模板代码请查阅专栏源代码 search.wxml 与 search.wxss，完整的 JS 逻辑代码见 search.js，具体代码位置见本节末尾图 12。

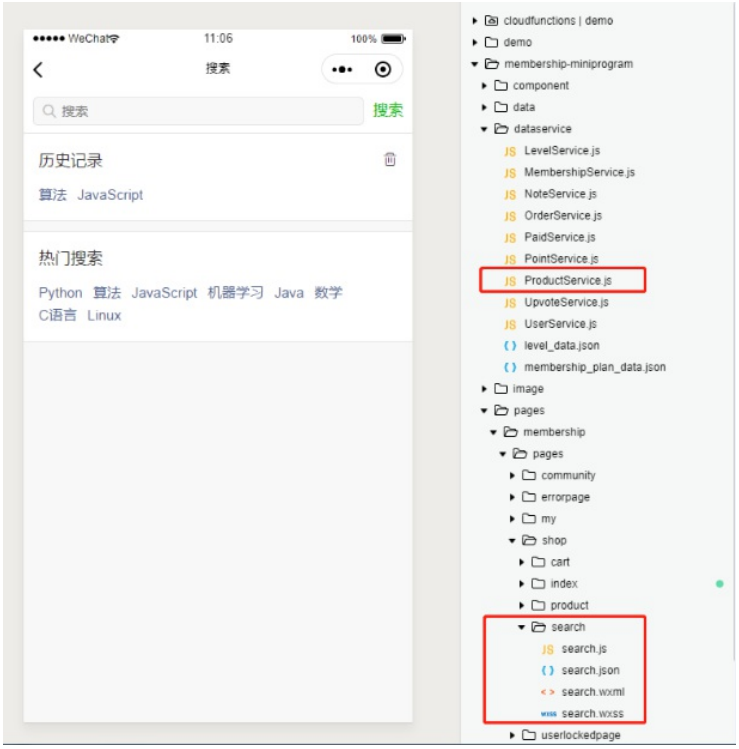
4. 专栏源代码

本专栏源代码已上传到 GitHub，请访问以下地址获取：

https://github.com/liujiec/Membership-ECommerce-Miniprogram

本节源代码内容在图 12 红框标出的位置。

图 12 本节源代码位置



下节预告

下一节，我们将编写云函数实现商品购买的支付接口，然后在商品详情页面和购物车页面中实现商品购买功能。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 请结合第二章第三节对“分类拆解法”的详细讲解内容，拆解本节图 **11**、第二节图 **2**、第二节图 **4** 的商品搜索页面，然后将自己的拆解结果与本节列出的拆解结果进行对照总结。
- 编写代码完成商品搜索页面，如碰到问题，请阅读本专栏源代码学习如何实现。

}

30 现在流行瀑布流：开发实现商城首页面

32 准备好收银机：开发实现商城支付接口