黑马商城实战项目

项目搭建

- 利用HBuilder X创建基本项目结构
- 运行项目
- 整理基本项目结构，并修改窗口外观

```
"globalStyle": {
  "navigationBarTextStyle": "white",
  "navigationBarTitleText": "黑马商城",
  "navigationBarBackgroundColor": "#1989fa",
  "backgroundColor": "#F8F8F8"
}
```

配置**tabbar**

- 创建tabbar对应的四个页面和图标准备好
- 将页面路径配置到pages.json中的pages数组中

```
"pages": [ //pages数组中第一项表示应用启动页，参考：
https://uniapp.dcloud.io/collocation/pages
        {
            "path": "pages/index/index"
        },
        {
            "path": "pages/member/member"
        },
        {
            "path": "pages/cart/cart"
        },
        {
            "path": "pages/search/search"
        }
    ]
```

- 配置tabbar

```
{
    "tabBar": {
        "list": [
            {
                "pagePath":"pages/index/index",
                "text":"首页",
                "iconPath":"static/icon/home.png",
                "selectedIconPath":"static/icon/home-
active.png"
            },
            {
                "pagePath":"pages/member/member",
                "text":"会员",
                "iconPath":"static/icon/member.png",
                "selectedIconPath":"static/icon/member-
active.png"
            },
            {
                "pagePath":"pages/cart/cart",
                "text":"购物车",
                "iconPath":"static/icon/cart.png",
                "selectedIconPath":"static/icon/cart-
active.png"
            },
            {
                "pagePath":"pages/search/search",
                "text":"搜索",
                "iconPath":"static/icon/search.png",
                "selectedIconPath":"static/icon/search-
active.png"
            }
        ]

    }
}
```

获取轮播图数据

- 封装uni.request请求，并挂在到全局
    - 创建util》api.js

```
// 封装get请求
```

```
const baseUrl = "http://localhost:8082"
export const myRequest = (options)=>{
    return new Promise((resolve,reject)=>{
        uni.request({
            method: options.method,
            data: options.data,
            url: baseUrl+options.url,
            success(res) {
                if(res.data.status !== 0) {
                    return uni.showToast({
                        title: '获取数据失败'
                    })
                }
                resolve(res)
            },
            fail(err) {
                uni.showToast({
                    title: '获取数据失败'
                })
                reject(err)
            }
        })
    })
}
```

- 在main.js中导入并挂载到全局

```
import { myRequest } from './util/api.js'
Vue.prototype.$myRequest = myReques
```

- 获取轮播图的数据
  - 定义获取轮播图的方法

```
methods: {
  async getSwipers () {
    const res = await this.$myRequest({
      method: 'GET',
      url: '/api/getlunbo'
    })
    this.swipers = res.data.message
  }
}
```

- 在onLoad中调用该方法

```
this.getSwipers()
```

实现轮播图的结构和数据渲染

- 定义轮播图的基本结构

```
<swiper class="swiper" indicator-dots :autoplay="true"
:interval="2000" circular>
  <swiper-item v-for="item in swipers" :key="item.id">
    <image :src="item.img"></image>
  </swiper-item>
</swiper>
```

- 样式，在工具中安装scss

```
<style lang="scss">
    .home{
        swiper{
            height: 380rpx;
            image{
                width: 750rpx;
                height: 380rpx;
            }
        }
    }
</style>
```

实现菜单导航结构

引入字体图标初始化样式

```
<style>
    @font-face {font-family: "iconfont";
      src: url('~@/static/fonts/iconfont.eot?t=1576335469449'); /*
IE9 */
      src: url('~@/static/fonts/iconfont.eot?
t=1576335469449#iefix') format('embedded-opentype'), /* IE6-IE8 */
      url('data:application/x-font-woff2;charset=utf-
8;base64,d09GMgABAAAAAAVMAAsAAAAAChwAAAT/AAEAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAHEIGVgCDKAqHQIYhATYCJAMUCwwABCAFhG0HUBurCFGUTk6f7EeBu6eIlFFaj
BY208XLlGHx/6/hiYfvx76d+2xFJLlUn75pp4tCoyQ8NgiJUJjuJfzNpSrpXIDoeDbs
3GjyH7Lfv+zTy1mok+GyEhYAW8AKaPvz/6eeG1QvDUo6SQW8bE4KYTDSMRj1R/5x7/T
P88Dmo7Jc/misYXOelwWYYEBjbGxrAQ7KEvQW8SwuJBO4nkCvRQ21q/PbB5hXQLdAvD
U6gHmbRQnhhG598xlb1lEvsu70PvEBfFU/H/+hF1IkNYO+149nMRz+6v7/fwrv4JAqI
6T5uXCbRMYWUIhnX8+tKT22ZUrvJ3PaCdCNryR+Sb2RfRPT+c9kbdF0s788siIRDWj6
GXkWL/ySEsVkfsmyftoYKiIZsiQfJ+Y2ll7MgdgDiE/AMF+CumLWsCSpdFXpjsNkefo
atW8WSY8xapRhreWm1a9vWvVEM6edaQdPapImTzTLLG4BwEBG/HJsK4bVEKkStJbxbU
Yy3othvO2ofJA7jBARcUZ4illDuEL29TB2C/3AGWj4WUfGYbG25LTICaEk1jWBR8KWC
0bqQJ+EjJYmOnvisiHzqtifvZO8X7NYzBaJWAJBglDIRIi3Fd/C3cbZjqEKvkh9Qmx2
SKCgMAhD15VHVkCQMCWOP72YnLB0euHUpROzJ/PEYiR6wxK8ShC+ZsryVibGlbIzorO
jqRKnhwyOKTIlMRoPSLgMlYvInDIKdwsQ8K2YbO2PReRsyxzZvrK0dWxUYUrIE2sPj8
+sHNw6tUVhj0AgslvelUfhDnNG4k0YwgedbyX/ovPRL8SnqwOt/atMII/RGGMacyXGJ
Oaq+MtGYDFMOL1Hk8OSJdfqARNMZkPmnDVh5DP0bjDpx6XAB6ROOJkqM4F0fNSNZMKX
B20QqI//n+swdzCvnv1/JszOzMEszPsskD8RZtrarNB14gGhubjWlduokVnWD9e9PyC
Qd1jtvPjStKTGs6VZFT/xjhhW0hpkG6PoC0wW8iGUrPK9prFY4AM+9GF8APhKfIvr5t
ct+IoAxU/eE3ILa4yrDxTgUfkrG0lPTtshU4pGtXT9UJqywWkDhzyi/5ka15D+WlPn1
Gd+QYF7s0j3dXTuS0oStZLum6on86NOwIbPcydIP3+STiDFaYk1Z6WFLPt7b7M6EhMA
+Gr+qtDilfia4mPFr0p8JZUS973QHulzYJxsPjJ6xdeW7Wd75WHHrZsGDXaUb1hacCu
QVmrOkt1DjdgTxSeY3EcWDqCzRLxB9PmH2DNplYhzhCDbNLfzY7dKy1jb/dqNFfBTXb
Wg8vhWsH1JP9IN0VQmIcif4isHay1vtVDzNIiyxeirbpZ0JKEXg3QAAqN+8jFMLN9E6
DZhQtJlAbJuS2ght6DqcwRNt1Potelicp8xbEqUHmz4IBCGfUIy6DtkwyZoIb+hmvQL
zXAQoddNbM3ZZyWKFk+YAqEYwxGayJZKsMuiVnxH2ucBW+Uq94m4MkHstjrO5AWVxHk
MqPq6J6JQsS3wnGxGeW7RsU0pklYi4rbbbeV3plZkC1h0iZGAIDEUGkFGxCopubpYdL
3/DtG8XIBDSn4knxBWMbWjrpZODPRCU8YqWZfGlT6tRwjFOynMKqBz6iI5xizk/FulS
ES0JCkizrY2q6Tialrl64ppLD0VysKcNVLkKFGjae8S40w5K14OB+WS9lKjkR/YgsrZ
sRnZzAAA') format('woff2'),
      url('~@/static/fonts/iconfont.woff?t=1576335469449')
format('woff'),
      url('~@/static/fonts/iconfont.ttf?t=1576335469449')
format('truetype'), /* chrome, firefox, opera, Safari, Android, iOS
4.2+ */
      url('~@/static/fonts/iconfont.svg?t=1576335469449#iconfont')
format('svg'); /* iOS 4.1- */
    }

    .iconfont {
      font-family: "iconfont" !important;
      font-size: 16px;
      font-style: normal;
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
    }

    .icon-shipin:before {
      content: "\f0024";
    }

    .icon-tupian:before {
      content: "\e650";
    }

    .icon-guanyuwomen:before {
      content: "\e608";
    }

    .icon-ziyuan:before {
      content: "\e60d";
    }
</style>
```

完成菜单导航基本结构

```
<view class="nav">
            <view class="item">
                <view class="iconfont icon-ziyuan"></view>
                <text>黑马超市</text>
            </view>
            <view class="item">
                <view class="iconfont icon-tupian"></view>
                <text>联系我们</text>
            </view>
            <view class="item">
                <view class="iconfont icon-guanyuwomen"></view>
                <text>社区图片</text>
            </view>
            <view class="item">
                <view class="iconfont icon-shipin"></view>
                <text>视频专区</text>
            </view>
        </view>
```

菜单导航样式

```
.nav{
  display: flex;
  align-items: center;
  .item{
    width: 25%;
    text-align: center;
    view{
      background: $shop-color;
      line-height: 120rpx;
      width: 120rpx;
      height: 120rpx;
      border-radius: 90px;
      margin:10px auto;
    }
    text{
      font-size: 15px;
    }
  }
  .iconfont{
    font-size: 25px;
    color: #fff;
    height: 50px;
  }
  .icon-tupian{
    font-size: 20px;
  }
}
```

实现推荐商品列表

定义基本结构

```
<view class="hot_goods">
  <view class="tit">推荐商品</view>
  <!-- 一般用法 -->
  <view class="goods_list">
    <view class="goods_item">
      <image></image>
      <view class="price">
        <text>1299</text>
        <text>12990</text>
      </view>
      <view class="name">华为（HUAWEI）荣耀6Plus 16G双4G版</view>
    </view>
  </view>
</view>
```

美化样式

```
.hot_goods {
  background: #eee;
  .tit{
    border-top: 2px solid #eee;
    border-bottom: 2px solid #eee;
    margin-top: 20px;
    margin-bottom: 3px;
    color: $shop-color;
    height: 50px;
    line-height: 50px;
    text-align: center;
    letter-spacing: 20px;
    background: #fff;
  }
  .goods_list {
    display: flex;
    padding: 0 15rpx;
    justify-content: space-between;
    overflow: hidden;
    flex-wrap: wrap;
```

```
      .goods_item {
        width: 355rpx;
        margin-bottom: 15rpx;
        background: #fff;
        padding: 10px;
        box-sizing: border-box;
        image{
          height: 150px;
          width: auto;
          mix-width:160px;
          margin: 10px auto;
        }
        .price{
          font-size: 18px;
          color: red;
          padding: 8px 0;
          text:nth-child(2){
            color: #ccc;
            text-decoration: line-through;
            margin-left: 10px;
            font-size: 13px;
          }
        }
        .name {
          font-size: 14px;

        }
      }
    }
  }
```

获取数据

- 定义获取数据的方法

  ```
  // 获取推荐商品
  async getGoods () {
    const res = await this.$myRequest({
      url: '/api/getgoods?pageindex=1'
    })
    this.goods = res.data.message
  }
  ```

- 在onLoad生命周期中调用该方法

```
this.getGoods()
```

渲染数据

- 通过v-for渲染数据

```
<view class="hot_goods">
  <view class="tit">推荐商品</view>
  <!-- 一般用法 -->
  <view class="goods_list">
    <view class="goods_item" v-for="item in goods"
:key="item.id">
      <image :src="item.img_url"></image>
      <view class="price">
        <text>{{item.sell_price}}</text>
        <text>{{item.market_price}}</text>
      </view>
      <view class="name">{{item.title}}</view>
    </view>
  </view>
</view>
```

完成黑马超市页面

改造导航菜单

- 定义数据

```
navs: [
  {
    icons: "iconfont icon-ziyuan",
    title: "黑马超市",
    path: "/pages/goods/list"
  },
  {
    icons: "iconfont icon-tupian",
    title: "社区图片",
    path: "/pages/pics/pics"
  },
  {
    icons: "iconfont icon-guanyuwomen",
    title: "联系我们",
    path: "/pages/contact/contact"
  },
  {
    icons: "iconfont icon-shipin",
    title: "学习视频",
    path: "/pages/videos/videos"
  }
]
```

- 渲染数据

```
<view class="nav">
  <view class="item" v-for="(item,index) in navs"
:key="index">
    <view :class="item.icons"></view>
    <text>{{item.title}}</text>
  </view>
</view>
```

- 给导航菜单注册点击事件

```
<view class="goods_item" v-for="item in goods"
:key="item.id">
```

- 定义跳转的方法

```
goNavigator (url) {
  uni.navigateTo({
    url
  })
}
```

创建黑马超市页面

- 创建页面，goods>list.vue
- 将页面路劲配置到pages文件中，修改标题

封装商品列表组件

- 在components下面创建goods>list.vue

```
<template>
    <view class="goods_list">
        <view class="goods_item" v-for="item in goods"
:key="item.id">
            <image :src="item.img_url"></image>
            <view class="price">
                <text>{{item.sell_price}}</text>
                <text>{{item.market_price}}</text>
            </view>
            <view class="name">{{item.title}}</view>
        </view>
    </view>
</template>

<script>
    export default {
        props:{
            goods:Array
        }
    }
</script>

<style lang="scss">
    .goods_list {
        display: flex;
        padding: 0 15rpx;
        justify-content: space-between;
        overflow: hidden;
        flex-wrap: wrap;
        .goods_item {
            width: 355rpx;
            margin-bottom: 15rpx;
            background: #fff;
            padding: 10px;
            box-sizing: border-box;
            image{
                height: 150px;
                width: 150px;
                display: block;
                margin: 10px auto;
            }
            .price{
                font-size: 18px;
                color: red;
                padding: 8px 0;
                text:nth-child(2){
                    color: #ccc;
                    text-decoration: line-through;
                    margin-left: 10px;
                    font-size: 13px;
                }
            }
            .name {
                font-size: 14px;

            }
        }
    }
</style>
```

- 在首页引入该组件

```
import goodsList from "../../components/goods-
list/index.vue"

components: {
  "goods-list":goodsList
}
```

- 使用组件并将数据传递到组件内部

```
<goods-list :goods="goods"></goods-list>
```

渲染商品列表

- 定义获取商品列表数据的方法并调用

```
<script>
    export default {
        data () {
            return {
                goods: []
            }
        },
        methods: {
            async getGoods () {
                const res = await this.$myRequest({
                    url: '/api/getgoods?pageindex=1'
                })
                this.goods = res.data.message
            },
        },

        onLoad () {
            this.getGoods()
        }
    }
</script>
```

- 引入商品组件并使用

```
<template>
    <view class="goods_list">
        <goods-list :goods="goods"></goods-list>
    </view>
</template>
<script>
    import goodsList from "../../components/goods-
list/index.vue"
    export default {
        components: {
            "goods-list": goodsList
        }
    }
</script>
```

实现上拉加载更多

- 通过onReachBottom来监听触底

  ```
  onReachBottom () {
    this.pageindex++
    this.getGoods()
  }
  ```

- 修改给goods赋值

  ```
  this.goods = [...this.goods,...res.data.message]
  ```

- 动态显示底线
  - 通过onReachBottom监听是否还有更多

    ```
    if(this.pageindex*10>this.goods.length) return
    this.flag = true
    ```

  - 通过v-if控制底线

    ```
    <view class="over_line" v-if="flag">----------我是有底线的----------</view>
    ```

实现下拉刷新

- 通过onPullDownRefresh进行下拉刷新的操作

  ```
  onPullDownRefresh() {
    this.goods = []
    this.pageindex = 1
    this.flag = false
    setTimeout(()=>{
      this.getGoods(()=>{
        uni.stopPullDownRefresh()
      })
    },1000)
  }
  ```

关于我们

实现社区图片

实现资讯列表

实现列表项的结构和样式

结构

```
<view class="news_item">
  <image src="../../static/logo.png"></image>
  <view class="content">
    <view class="tit">1季度多家房企利润跌幅超50% 去库存促销战打响</view>
    <view class="info">
      <text>发表时间：2019-12-23</text>
      <text>浏览：1次</text>
    </view>
  </view>
</view>
```

样式

```
.news{
    .news_item{
        display: flex;
        padding: 5px 10px;
        border-bottom: 3px solid $shop-color;
        image{
            width: 300rpx;
            height: 150rpx;
        }
        .content {
            padding: 5px 10px;
            position: relative;
            .tit{
                font-size: 30rpx;
            }
            .info{
                font-size: 26rpx;
                position: absolute;
                left: 10px;
                bottom: 5px;
                text:nth-child(2){
                    margin-left: 20px;
                }
            }
        }
    }
}
```

封装为组件

创建news-item.vue

```
<template>
    <view>
        <view class="news_item" v-for="item in data"
:key="item.id">
            <image :src="item.img_url"></image>
            <view class="content">
                <view class="tit">{{item.title}}</view>
                <view class="info">
                    <text>发表时间：{{item.add_time | formatDate}}
</text>
                    <text>浏览：{{item.click+123}}次</text>
                </view>
            </view>
        </view>
    </view>
</template>

<script>
    export default {
        props: ['data'],
        filters:{
          formatDate(data){
                const date = new Date(data)
                console.log(date)
                const day =
date.getMonth().toString().padStart(2,'0')+'-
'+date.getDay().toString().padStart(2,'0')
```

```
                return date.getFullYear()+'-'+day
            }
        }
    }
</script>

<style lang="scss">
    .news_item{
        display: flex;
        padding: 5px 10px;
        border-bottom: 1rpx solid $shop-color;
        image{
            max-width: 200rpx;
            min-width: 200rpx;
            height: 150rpx;
        }
        .content {
            padding: 5px 10px;
            position: relative;
            .tit{
                font-size: 30rpx;
            }
            .info{
                font-size: 26rpx;
                position: absolute;
                left: 10px;
                bottom: 5px;
                text:nth-child(2){
                    margin-left: 20px;
                }
            }
        }
    }
</style>
```

在新闻页面导入并使用

```
<template>
    <view class="news">
        <new-item :data="newsList"></new-item>
    </view>
```

```
</template>
<script>
    import newItem from '../../components/new-item/new-item.vue'
    export default {
        data() {
            return {
                newsList: []
            }
        },
        methods: {
            async getNewsList () {
                const res = await this.$myRequest({
                    url:'/api/getnewslist'
                })
                this.newsList = res.data.message
            }
        },
        onLoad () {
            this.getNewsList()
        },
        components: {
            "new-item":newItem
        }
    }
</script>

<style lang="scss">
</style>
```

点击列表进入详情

- 点击子组件列表项通过this.$emit触发父组件的方法

  ```
  navigatorTo (item) {
    this.$emit('clickItem',item)
  }
  ```

- 父组件通过注册clickItem事件及事件函数实现跳转操作

  ```
  <template>
      <view class="news">
  ```

```
        <new-item :data="newsList" @clickItem="goDetail">
</new-item>
    </view>
</template>
<script>
    export default {
        methods: {
            goDetail (data) {
                console.log(data)
                uni.navigateTo({
                    url: '/pages/news-detail/news-detail'
                })
            }
        }
    }
</script>
```

- 新建/pages/news-detail/news-detail页面

实现资讯详情

实现基本结构

```
<template>
    <view class="news_detail">
        <view class="news_title">
            标题
        </view>
        <view class="info">
            <text>发表时间：</text>
            <text>浏览：</text>
        </view>
        <view class="content">

        </view>
    </view>
</template>

<script>
    export default {
        data() {
            return {

            }
        },
        methods: {

        }
    }
</script>

<style lang="scss">
.news_detail {
    padding: 15rpx;
    .news_title{
        text-align: center;
        font-size: 32rpx;
    }
    .info{
        font-size: 28rpx;
        display: flex;
        justify-content: space-between;
    }
}
</style>
```

获取详情的数据

```
methods: {
  async getNewsDetail(id){
    const res = await this.$myRequest({
      url: '/api/getnew/'+id
    })
    console.log(res)
    this.newsDetail = res.data.message[0]
  }
},

  onLoad (options){
    this.getNewsDetail(options.id)
  }
```

实现详情的渲染

```
<view class="news_title">
  {{newsDetail.title}}
    </view>
<view class="info">
  <text>发表时间：{{newsDetail.add_time | formatDate}}</text>
<text>浏览：{{newsDetail.click}}</text>
</view>
<view class="content">
  <rich-text :nodes="newsDetail.content"></rich-text>
</view>
```

实现商品详情页

商品列表注册点击事件

```
<template>
    <view class="goods_list">
        <view @click="itemClick(item)" class="goods_item" v-
for="item in goods" :key="item.id">
        </view>
    </view>
</template>
<script>
    export default {
        props:{
            goods:Array
        },
        methods: {
            itemClick(item) {
                this.$emit('itemClick',item)
            }
        }
    }
</script>
```

父组件绑定自定义事件进行跳转

```
<goods-list @itemClick="godetail" :goods="goods"></goods-list>
godetail (item) {
  uni.navigateTo({
     url: '/pages/goods-detail/goods-detail?id='+item.id
  })
}
```

创建商品详情页

获取详情轮播图数据

```
methods: {
  async getDetail(){
    const res = await this.$myRequest({
      url:'/api/getthumimages/'+this.id
    })
    console.log(res)
  }
},
onLoad(options){
    this.id = options.id
    this.getDetail()
}
```

渲染轮播图

```
<swiper indicator-dots>
  <swiper-item v-for="item in swipers" :key="item.src">
    <image :src="item.src"></image>
  </swiper-item>
</swiper>
<style lang="scss">
.goods_detail{
    swiper{
        height: 700rpx;
        image{
            width: 100%;
            height: 100%;
        }
    }
}
</style>
```

获取商品信息

```
methods: {
  async getDetailInfo () {
    const res = await this.$myRequest({
      url:'/api/goods/getinfo/'+this.id
    })
    this.info = res.data.message[0]
  }
},
onLoad(options){
    this.id = options.id
    this.getDetailInfo()
}
```

完成商品信息结构和渲染