

阅读页面实现



pages.json 配置阅读页面搜索框

1. 在 pages.json 文件中针对阅读页面配置导航栏搜索框

```
{
  "path" : "pages/article/article",
  "style": {
    "navigationBarTitleText": "阅读",
    "navigationBarTextStyle": "white",
    "navigationBarBackgroundColor": "#345DC2",
    "app-plus": {
      "titleNView": { //导航配置
        "searchInput": { // 搜索框
          "align": "center",
          "placeholder": "搜索你想要的内容",
          "borderRadius": "30rpx",
          "backgroundColor": "#F0F1F2",
          "placeholderColor": "#979C9D", //提示字体颜色
        }
      }
    }
  }
}
```

```
      "disabled": true //禁止输入，点击进入新搜索页面
    }
  }
}
},
```

2. 在 article.vue 页面文件中，监听点击搜索框跳转到搜索页

```
// 监听原生输入框点击事件
onNavigationBarSearchInputClicked() {
  // 在 /common/mixin/mixin.js 中定义了
  this.navTo('/pages/search/search')
},
```

优化标签栏组件实现滚动效果 tab-bar.vue

需求

- 当前 tab-bar.vue 组件中，如果超过5个标签，会有拥挤，这样不美观。
- 解决上面问题，使用 scroll-view 组件将标签栏包裹起来，超过5个标签时，实现左右滑动。

优化标签栏代码

基于原有的 tab-bar.vue 组件进行优化如下：

1. 模板代码中将 <view class="bar-view center"> 替换为 <scroll-view> 左右滚动组件，在其子节点view上，动态绑定每个标签宽度。

```
<template>
  <view class="tab-bar" @touchmove.stop.prevent="()=>{}">
    <!-- <view class="bar-view center"> -->
    <scroll-view id="nav-bar" class="noScroll bar-view" scroll-x scroll-with-animation
      :scroll-left="scrollLeft">
      <!-- 动态绑定每个标签宽度 -->
      <view :style="{width: `${itemWidth}px}`"
        class="bar-item" :class="{current: index === value}"
        v-for="(item, index) in tabs" :key="index"
        @click="changeTab(index)"
      >
        {{item.name}}
      </view>
    </scroll-view>
    <!-- </view> -->
  </view>
</template>
```

2. 样式修改看下面注释处

```
<style lang="scss">
.tab-bar {
  width: 100%;
  height: 80rpx;
  background-color: #FFFFFF;
  border-bottom: 1px solid #efff4;
  .bar-view {
    width: 100%;
    text-align: center;
    /* 1 ++++ */
    white-space: nowrap;
    .bar-item {
      /* 2 ++++ */
      display: inline-block;
      width: 150rpx;
      /* flex: 1; */

      font-size: 30rpx;
      line-height: 80rpx;
      position: relative;
      &:after {
```

3. 优化js业务逻辑

- o let 声明滚动定时器变量，用于判断短时间内是否点击多个标签
- o data 选项声明属性，用于接收每个tab占的宽度和左右滑动距离
- o watch 监听标签数组变化，来计算每个tab占的宽度
- o changeTab 方法当tab切换时触发，计算是否将标签滚动到中间

```
<script>
  /* 1. 滚动定时器 */
  let scrollTimer = null
  export default {
    props: {
      value: { // vue 语法糖，v-model双向绑定：1. props声明value，2. 修改它时触发input事件传递
        type: Number,
        default: 0
      },
      tabs: { // 标签选项数组
        type: Array,
        default: () => [
          {
            id: 1,
            name: '课程'
          },
          {
```

```
        id: 2,
        name: '文章'
      },
      {
        id: 3,
        name: '问答'
      }
    ]
  },
},

/* 2. 声明属性 */
data() {
  return {
    scrollLeft: 0, //顶部选项卡左滑距离
    windowWidth: uni.getSystemInfoSync().windowWidth, //屏幕宽度
    itemWidth: uni.upx2px(150), //每个标签的宽度 默认150rpx
  }
},

/* 3. 监听标签数组变化，来计算每个标签占的宽度 */
watch: {
  tabs: {
    handler(newVal) {
      if(newVal && newVal.length < 5) {
        // 小于5个，则平均分配每个标签宽度， px
        this.itemWidth = this.windowWidth / newVal.length
      }
    },
    immediate: true
  }
},

methods: {

  /* changeTab(index) {
    // console.log('index', index)
    //改变tab，点击不同的标签，才更新值
    if(this.value !== index) {
      this.$emit('input', index)
    }
  } */

  // 4. tab切换，计算是否将标签滚动到中间
  changeTab(index) {

    if(scrollTimer) {
      //多次切换只执行最后一次
      clearTimeout(scrollTimer);
      scrollTimer = false;
    }

    //获取可滑动总宽度
```

```
let width = this.itemWidth * (index+1)

//延迟400ms,防止切换太快
scrollTimer = setTimeout(() => {
  if (width - this.itemWidth/2 > this.windowWidth / 2) {
    // 如果当前项越过中心点, 将其放在屏幕中心
    this.scrollLeft = width - this.itemWidth/2 - this.windowWidth / 2;
  }else{
    this.scrollLeft = 0;
  }
  // 绑定当前点击下标
  this.$emit('input', index)
}, 200)
},
}
}
</script>
```

4. 测试:

- 原引用了 tab-bar.vue 的父组件不需要做任何修改。
- tabs 数组中添加大于5个标签元素，然后测试是否可以左右滑动，是否被点击的会居中（计算允许居中的才会居中）

阅读列表页面效果实现

参考: <http://www.mescroll.com/uni.html>

第三方 mescroll 下拉刷新组件的【mescroll-swiper.vue 和 mescroll-swiper-item.vue】

阅读页面实现 article.vue

1. article.vue 页面模板代码

```
<template>
  <view class="container-list">
    <!-- 标签导航栏 -->
    <tab-bar ref="tabBar" v-model="tabIndex" :tabs="tabs"></tab-bar>
    <!-- 列表区域 -->
    <swiper :style="{height: height}" :current="tabIndex" @change="swiperChange">
      <swiper-item v-for="(tab, i) in tabs" :key="i">
        <mescroll-item ref="mescrollItem" :i="i" :index="tabIndex" :tabs="tabs">
        </mescroll-item>
      </swiper-item>
    </swiper>
  </view>
</template>
```

2. article.vue页面业务逻辑

```
<script>
import tabBar from '@components/common/tab-bar.vue'
import mescrollItem from './components/article-swiper-item.vue'

export default {
  components: {tabBar, mescrollItem},

  data() {
    return {
      height: "400px", // 需要固定swiper的高度
      tabs: [{name: '推荐'}, {name: 'Java'}, {name: '前端'}, {name: '测试'}, {name: '云计算'}, {name: 'UI设计'}, {name: '人工智能'}],
      tabIndex: 0 // 当前tab的下标
    }
  },

  onLoad() {
    // 需要固定swiper的高度，注意减掉标签栏高度80rpx
    this.height = uni.getSystemInfoSync().windowHeight + 'px'
  },

  methods: {

    // 轮播菜单
    swiperChange(e){
      this.tabIndex = e.detail.current
      // 触发标签栏移动
      this.$refs.tabBar.changeTab(this.tabIndex)
    },

  },
}
</script>
```

3. article.vue页面样式, 超出部分隐藏, 这样页面无回弹, 使用 mescroll 列表的回弹

```
<style lang="scss">
page, .container-list {
  height: 100%;
  /* 超出隐藏*/
  overflow: hidden;
}
</style>
```

阅读列表子组件实现 article-swiper-item.vue

1. 创建列表子组件 pages/article/components/article-swiper-item.vue，参考：mescroll-swiper-item.vue

注意：upOption.auto: false 才不会被初始化调用多次 upCallback

```
<template>
  <!--
    @init="mescrollInit" 初始化mescroll后是否自动加载数据
    bottom="80" 在底部空出标签栏高度
    offset="0" 距底部多远时,触发upCallback,默认150
  -->
  <mescroll-uni :ref="'mescrollRef'+i"
    @init="mescrollInit" height="100%" bottom="80" offset="0"
    :down="downOption" @down="downCallback" :up="upOption"
    @up="upCallback" >
    <!-- 数据列表 -->
    <article-item v-for="i in 12"></article-item>
  </mescroll-uni>
</template>

<script>
  import MescrollMixin from "@components/mescroll-uni/mescroll-mixins.js";
  import MescrollMoreItemMixin from "@components/mescroll-uni/mixins/mescroll-more-item.js";
  import articleItem from '@components/common/article-item.vue'

  export default {
    mixins: [MescrollMixin,MescrollMoreItemMixin], // 注意此处还需使用
    MescrollMoreItemMixin (必须写在MescrollMixin后面)
    components: {
      articleItem
    },
    data() {
      return {
        downOption:{
          auto:false // 不自动加载 (mixin已处理第一个tab触发downCallback)
        },
        upOption:{
          auto: false, // 不自动加载
          noMoreSize: 4, //如果列表已无数据,可设置列表的总数量要大于半页才显示无更多数据;避免列表数据过少(比如只有一条数据),显示无更多数据会不好看;默认5
        },
      }
    },
    props:{
      i: Number, // 每个tab页的专属下标 (除了支付宝小程序必须在这里定义, 其他平台都可不用写, 因为已在MescrollMoreItemMixin定义)
      index: { // 当前tab的下标 (除了支付宝小程序必须在这里定义, 其他平台都可不用写, 因为已在MescrollMoreItemMixin定义)
        type: Number,
        default(){
          return 0
        }
      }
    }
  }
}
```

```
    }  
  },  
  tabs: { // 为了请求数据,演示用,可根据自己的项目判断是否要传  
    type: Array,  
    default(){  
      return []  
    }  
  },  
  methods: {  
    /*  
      下拉刷新的回调, 可以不写, mixins已默认:  
      下拉刷新和上拉加载调同样的接口, 直接mescroll.resetUpScroll()即可  
      重置列表为第一页 (自动执行 page.num=1, 再触发upCallback方法 )  
    */  
    // downCallback() { },  
  
    /*上拉加载的回调: 其中page.num:当前页 从1开始, page.size:每页数据条数,默认10 */  
    upCallback(page) {  
      //联网加载数据  
      // 传入0就没有数据  
      // this.mescroll.endSuccess(0)  
      // 查询数据后, 传递 (列表展示总记录数list.length, 后台响应总记录total)  
      this.mescroll.endBySize(10, 10);  
    }  
  }  
}  
</script>
```

2. 如果数据为空时, ~空空如也~ 显示的图片不居中:

- 找到: /mxg-education-app/components/mescroll-uni/components/mescroll-empty.vue
- 然后针对 image 元素添加样式

```
image {  
  display:inline-block;  
}
```

重构搜索页列表组件

问题

重构 search.vue 搜索页的列表子组件, 第一次进入页面时, 在父组件search.vue中手动触发第一页数据。

前面讲解将 /components/mescroll-uni/mixins/mescroll-more-item.js 中的 mescrollInit 方法中的 this.mescroll.triggerDownScroll(); 注释掉了, 不然进入页面时会触发两次查询操作: 一次自动调用查询, 一次手动触发查询。

优化

现在不要注释 `this.mescroll.triggerDownScroll();`

找到 `mescroll-more-item.js` 中的 `mescrollInit` 方法，不要将 `this.mescroll.triggerDownScroll();` 注释掉

```
mescrollInit(mescroll) {  
  this.mescroll = mescroll;  
  this.mescrollInitByRef && this.mescrollInitByRef(); // 兼容字节跳动小程序  
  // 自动加载当前tab的数据  
  if(this.i === this.index){  
    this.isInit = true; // 标记为true  
    this.mescroll.triggerDownScroll();  
  }  
},
```

找到搜索页中的课程列表组件 `pages/search/components/course-list.vue`、文章列表 `article-list.vue`、问答列表 `question.list`，将 `mescroll-body` 组件元素上的 `@init="mescrollInit"` 删除掉，这样就第一次不会自动加载。

```
<!-- 不要加 @init="mescrollInit" -->  
<mescroll-body :ref="'mescrollRef'+i" @down="downCallback"  
  @up="upCallback" :down="downOption" :up="upOption">
```

调用数据接口渲染标签栏

- 标签栏显示的分类数据接口方法，封装在 `course.js` 中
- 通过分类id分页查询文章列表数据接口方法，封装在 `article.js` 中

实现：

1. 在 `article.vue` 页面中导入 `course.js`，调用查询分类方法查询数据

```
import courseApi from '@api/course.js'  
  
export default {  
  onLoad() {  
    // 需要固定swiper的高度，注意减掉标签栏高度80rpx  
    this.height = uni.getSystemInfoSync().windowHeight + 'px'  
  
    // 查询分类  
    this.loadData()  
  },  
  methods: {  
    async loadData() {  
      uni.showLoading({title: '加载中', mask: true})  
      //查询分类  
      const {data} = await courseApi.getCategoryList()  
      data.unshift({id: null, name: '推荐'}) //添加`推荐`到第1个元素
```

```
      this.tabs = data
      uni.hideLoading()
    },
  }
}
```

2. 在 article-swiper-item.vue 列表组件中

- 导入 article.js
- data中声明 list 数组，接收查询到的列表数据
- upCallback 方法中调用查询文章列表数据接口

```
import api from '@api/article.js'

export default {
  data() {
    return {
      list: [], //列表数据
    },
  },
  methods: {
    /*上拉加载的回调：其中page.num:当前页 从1开始， page.size:每页数据条数,默认10 */
    async upCallback(page) {
      // 根据分类id分页查询列表数据
      const query = { categoryId: this.tabs[this.index].id }
      const {data} = await api.getList(query, page.num, page.size)
      // 注意：数据在 records 中，
      const list = data.records

      // 如果是第一页需手动置空列表,回到顶部
      if(page.num == 1) {
        this.list = []
        this.mescroll.scrollTo(0,0)
      }
      //追加新数据
      this.list = this.list.concat(list)

      // 请求成功，隐藏加载状态，判断是否数据全部加载完（后台接口有返回列表的总数据量 total）
      this.mescroll.endBySize(list.length, data.total)

      // 传入0就没有数据
      // this.mescroll.endSuccess(0)
    }
  }
}
```

问答页面实现



pages.json 配置问答页面搜索框

1. 在 pages.json 文件中针对问答页面配置导航栏搜索框

```
{
  "path": "pages/question/question",
  "style": {
    "navigationBarTitleText": "问答",
    "navigationBarTextStyle": "white",
    "navigationBarBackgroundColor": "#345DC2",
    "app-plus": {
      "titleNView": { // 导航配置
        "searchInput": { // 搜索框
          "align": "center",
          "placeholder": "搜索你想要的内容",
          "borderRadius": "30rpx",
          "backgroundColor": "#F0F1F2",
```

```
        "placeholderColor": "#979C9D", //提示字体颜色
        "disabled": true //禁止输入，点击进入新搜索页面
    }
    }
    }
    },
},
```

2. 在 question.vue 页面文件中，监听点击搜索框跳转到搜索页

```
// 监听原生输入框点击事件
onNavigationBarSearchInputClicked() {
    // 在 /common/mixin/mixin.js 中定义了
    this.navTo('/pages/search/search')
},
```

创建标签栏数据文件

1. 创建 /config/question-tabs.js 用于渲染在标签栏

```
export default [
  {
    id: '1',
    name: '热门回答'
  }, {
    id: '2',
    name: '最新问题',
  }, {
    id: '3',
    name: '等待回答'
  }
]
```

创建与封装查询问题数据列表接口

1. EasyMock 创建分页查询热门回答列表接口 (按回复数降序)

- o 描述：分页查询热门回答列表接口
- o URL: `/question/api/question/hot`
- o 请求方式: `POST`
- o mock.js 语法:

```
{
  "code": 20000,
```

```
"message": "查询成功",
"data": {
  "total": "@integer(30, 60)", // 总记录数
  "records|20": [{
    "id|+1": 10, //初始值10开始, 每条+1
    "userId": "@integer(10, 30)",
    "nickName": "@cname",
    "userImage": "@image",
    "title": "@csentence", // 标题
    "viewCount": "@integer(5, 300)", // 浏览数
    "thumhup": "@integer(2, 20)", // 点赞数
    "reply": "@integer(1, 10)", // 回复数
    "status|1": [1, 2], // 1: 未解决, 2: 已解决
    "createDate": "@datetime",
    "updateDate": "@datetime"
  }]
}
```

2. EasyMock 创建分页查询最新问题列表接口 (按更新时间降序)

- 描述: 分页查询最新问题列表接口
- URL: `/question/api/question/new`
- 请求方式: `POST`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "查询成功",
  "data": {
    "total": "@integer(30, 60)", // 总记录数
    "records|20": [{
      "id|+1": 10, //初始值10开始, 每条+1
      "userId": "@integer(10, 30)",
      "nickName": "@cname",
      "userImage": "@image",
      "title": "@csentence", // 标题
      "viewCount": "@integer(5, 300)", // 浏览数
      "thumhup": "@integer(2, 20)", // 点赞数
      "reply": "@integer(1, 10)", // 回复数
      "status|1": [1, 2], // 1: 未解决, 2: 已解决
      "createDate": "@datetime",
      "updateDate": "@datetime"
    }]
  }
}
```

3. EasyMock 创建分页查询等待回答列表接口 (0回复数)

- 描述: 分页查询等待回答列表接口
- URL: `/question/api/question/wait`

- 请求方式: POST
- mock.js 语法:

```
{
  "code": 20000,
  "message": "查询成功",
  "data": {
    "total": "@integer(30, 60)", // 总记录数
    "records|20": [{
      "id|+1": 10, // 初始值10开始, 每条+1
      "userId": "@integer(10, 30)",
      "nickName": "@cname",
      "userImage": "@image",
      "title": "@csentence", // 标题
      "viewCount": "@integer(5, 300)", // 浏览数
      "thumbup": "@integer(2, 20)", // 点赞数
      "reply": 0, // 回复数
      "status": 1, // 1: 未解决, 2: 已解决
      "createDate": "@datetime",
      "updateDate": "@datetime"
    }]
  }
}
```

4. 在 `/api/question.js` 文件封装调用数据接口方法

```
// 分页查询热门问答列表
getHotList(current = 1, size = 20) {
  return request({
    url: '/question/api/question/hot',
    method: 'POST',
    data: {current, size}
  })
},

// 分页查询最新问题列表
getNewList(current = 1, size = 20) {
  return request({
    url: '/question/api/question/new',
    method: 'POST',
    data: {current, size}
  })
},

// 分页查询等待回答列表
getWaitList(current = 1, size = 20) {
  return request({
    url: '/question/api/question/wait',
    method: 'POST',
    data: {current, size}
  })
}
```

```
    })  
  },
```

问答列表页面效果实现

问答页面实现 question.vue

参考阅读页面 `article.vue`

1. 导入 `question-swiper-item.vue` 组件
2. 导入 `question-tabs.js` 标签栏数据文件，并引用

```
<template>  
  <view class="container-list">  
    <!-- 标签导航栏 -->  
    <tab-bar ref="tabBar" v-model="tabIndex" :tabs="tabs"></tab-bar>  
    <!-- 列表区域 -->  
    <swiper :style="{height: height}" :current="tabIndex" @change="swiperChange">  
      <swiper-item v-for="(tab, i) in tabs" :key="i">  
        <mescroll-item ref="mescrollItem" :i="i" :index="tabIndex" :tabs="tabs">  
        </mescroll-item>  
      </swiper-item>  
    </swiper>  
  </view>  
</template>  
  
<script>  
  import tabBar from '@components/common/tab-bar.vue'  
  // 导入 question-swiper-item.vue  
  import mescrollItem from './components/question-swiper-item.vue'  
  // 导入标签栏数据  
  import tabs from '@config/question-tabs.js'  
  
  export default {  
    components: {tabBar, mescrollItem},  
  
    data() {  
      return {  
        height: "400px", // 需要固定swiper的高度  
        tabs, // 标签栏数据  
        tabIndex: 0 // 当前tab的下标  
      }  
    },  
  
    onLoad() {  
      // 需要固定swiper的高度，注意减掉标签栏高度80rpx  
      this.height = uni.getSystemInfoSync().windowHeight + 'px'  
    },  
  },  
</script>
```

```
methods: {  
  // 轮播菜单  
  swiperChange(e){  
    this.tabIndex = e.detail.current  
    // 触发标签栏移动  
    this.$refs.tabBar.changeTab(this.tabIndex)  
  },  
  
},  
  
}  
</script>  
  
<style lang="scss">  
  page, .container-list {  
    height: 100%;  
    /* 超出隐藏*/  
    overflow: hidden;  
  }  
</style>
```

问答列表子组件实现 question-swiper-item.vue

1. 创建 question-swiper-item.vue 组件，完成以下js业务逻辑：

- 导入并引用 question-item.vue
- 导入 /api/question.js
- upCallback方法中使用 switch 判断标签下标后，调用对应的数据接口

```
<script>  
  // 1. 导入  
  import questionItem from '@/components/common/question-item.vue'  
  import api from '@/api/question.js'  
  
  export default {  
  
    components: {  
      questionItem // 修改为问题item  
    },  
  
    methods: {  
      /*上拉加载的回调：其中page.num:当前页 从1开始， page.size:每页数据条数,默认10 */  
      async upCallback(page) {  
        let response = null // 查询响应结构  
        switch (this.index){  
          case 0: // 热门回答  
            response = await api.getHotList(page.num, page.size)  
            break;  
          case 1: // 最新问题  
            response = await api.getNewList(page.num, page.size)
```



```
        break;
      case 2: // 等待回答
        response = await api.getWaitList(page.num, page.size)
        break;
      default:
        break;
    }
    // 响应数据
    const data = response.data
    // 列表数据
    const list = data.records

    // 如果是第一页需手动置空列表,回到顶部
    if(page.num == 1) {
      this.list = [];
      this.mescroll.scrollTo(0,0)
    }
    //追加新数据
    this.list = this.list.concat(list);

    // 请求成功, 隐藏加载状态, 判断是否数据全部加载完 (后台接口有返回列表的总数据量
total)

    this.mescroll.endBySize(list.length, data.total);

    // 传入0就没有数据
    // this.mescroll.endSuccess(0)
  }
}
}
</script>
```

2. 模板代码, 引用 `question-item` 问题列表组件元素

```
<template>
  <!--
    @init="mescrollInit" 初始化mescroll后是否自动加载数据
    bottom="80" 在底部空出标签栏高度
    offset="0" 距底部多远时,触发upCallback,默认150
  -->
  <mescroll-uni :ref="'mescrollRef'+i"
    @init="mescrollInit" height="100%" bottom="80" offset="0"
    :down="downOption" @down="downCallback" :up="upOption"
    @up="upCallback">
    <!-- 数据列表 -->
    <question-item v-for="(item, index) in list"
      :key="index" :item="item">
    </question-item>
  </mescroll-uni>
</template>
```

文章详情页实现



创建配置文章详情页

1. 创建文章详情页文件 `/pages/article/details.vue`
2. 配置 `page.json`

```
{
  "path": "pages/article/details",
  "style": {
    "navigationBarTitleText": "",
    "backgroundColor": "#FFFFFF", // 回弹颜色
    "app-plus": {
      "titleNView": { // 导航配置
        // #ifdef APP-PLUS
      }
    }
  }
}
```

```
        {  
            "type": "share" // 分享按钮  
        }  
    ]  
    // #endif  
}  
}  
},  
},
```

3. 在 /components/common/article-item.vue 添加点击事件跳转文章详情页

```
<view class="article-item" @click="navTo(`/pages/article/details?id=${item.id}`)">
```

编写静态页面代码

1. 将 `hello-uni-app` 项目中的标签组件 `/components/uni-tag/uni-tag.vue` 导入到 `mxg-education-app` 项目的相同目录下。
2. 页面模板代码
 - **v-html指令**: App端和H5端支持 `v-html`；其他端不支持 `v-html`，需要使用 `rich-text` 组件的 `nodes` 属性渲染 `html`。

```
<template>  
  <view>  
    <!-- 所属标签 -->  
    <view class="tag-list row">  
      <uni-tag class="tag-view" text="vue.js" :inverted="true" :circle="true"  
type="primary" size="small" />  
      <uni-tag class="tag-view" text="html" :inverted="true" :circle="true"  
type="primary" size="small" />  
    </view>  
  
    <!-- 标题与内容 -->  
    <view class="content-main">  
      <text class="title">  
        问答列表页面功能实现  
      </text>  
      <view class="info">  
        <view class="author center">  
          <image src="/static/logo.png"></image>  
          <text>梦老师</text>  
        </view>  
        <text> · 2023-12-30</text>  
        <text> · 100人阅读</text>  
      </view>  
  
    <!-- 主体内容 -->
```

```
<!-- #ifdef MP-->
<!-- 小程序端使用rich-text组件的nodes渲染html代码,
      selectable="true" 只有百度小程序才可以复制 -->
<rich-text class="markdown-body" selectable="true"
      nodes="<div style='color: red'>渲染html代码</div>"></rich-text>
<!--#endif-->
<!-- #ifndef MP-->
<text class="markdown-body" selectable="true"
      v-html="`<div style='color: red'>渲染html代码</div>`" ></text>
<!--#endif-->
</view>

<!-- 热门评论 -->
<view class="footer">
  <!-- 评论 -->
  <view class="comment">
    <view class="footer-header">热门评论</view>
    <view class="comment-item row">
      <image src="/static/logo.png" ></image>
      <view class="comment-right column">
        <view class="info space-between center">
          <text>梦学谷</text>
          <text>2008-12-12</text>
        </view>
        <text class="content">用户权限中的内容的错误以及访问串行数据</text>
      </view>
    </view>
    <view class="comment-item row">
      <image src="/static/logo.png" ></image>
      <view class="comment-right column">
        <view class="info space-between center">
          <text>梦小二</text>
          <text>刚刚</text>
        </view>
        <text class="content">好文章值得学习! </text>
      </view>
    </view>
  </view>
</view>

<!-- 评论框 -->
<view class="bottom center" @touchmove.stop.prevent="()=>{}">
  <textarea class="textarea" placeholder="有何高见，展开讲讲..." />
  <button size="mini" class="btn">提交</button>
</view>

<!-- 分享 -->
<mxg-share ref="share"></mxg-share>
</view>
</template>
```

3. 页面样式代码

```
<style lang="scss">
.tag-list {
  // 一行排列不下，换行
  flex-wrap: wrap;
  padding: 10rpx 25rpx;
  font-size: 14px;
  background-color: #ffffff;
  .tag-view {
    margin-top: 15rpx;
    margin-right: 20rpx;
  }
}

.content-main {
  padding: 25rpx;
  .title {
    font-size: 45rpx;
    line-height: 55rpx;
    font-weight: bold;
  }
  .info {
    display: flex;
    align-items: center;
    margin: 30rpx 0;
    .author {
      font-size: 30rpx;
      color: #303133;
      image {
        width: 45rpx;
        height: 45rpx;
        border-radius: 100%;
        margin-right: 10rpx;
      }
    }
    &>text {
      margin-left: 10rpx;
      font-size: 25rpx;
      color: #999;
    }
  }
}

.footer {
  background-color: #F1F1F1;
  padding-top: 10rpx;
  /* 标题 */
  .footer-header{
    font-size: 30rpx;
    color: #303133;
    font-weight: bold;
    padding: 25rpx;
    &:before{
      content: '';
```

```
        width: 0;
        height: 40rpx;
        margin-right: 25rpx;
        border-left: 6rpx solid $mxg-color-primary;
    }
}

/* 评论 */
.comment {
    background-color: #FFFFFF;
    margin-top: 10rpx;
    // 最下方有评论按钮,
    padding-bottom: 120rpx;
    .comment-item {
        padding: 20rpx 25rpx;
        image{
            width: 50rpx;
            height: 50rpx;
            border-radius: 50rpx;
            margin-right: 20rpx;
        }

        .comment-right{
            width: calc(100% - 80rpx);
            font-size: 25rpx;
            color: $mxg-text-color-grey;
            .content{
                font-size: 30rpx;
                color: $mxg-text-color-black;
                margin: 10rpx 0;
            }
        }
    }
}

/* 底部 */
.bottom {
    position: fixed;
    left: 0;
    right: 0;
    bottom: 0;
    white-space: nowrap;// 不换行
    padding: 20rpx;
    background-color: #FFFFFF;
    border-top: $mxg-underline;
    .textarea {
        font-size: 30rpx;
        padding: 15rpx 20rpx;
        width: 100%;
        height: 70rpx;
        background-color: #F8F9FB;

        border-radius: 30rpx;
```

```
    }  
    .btn {  
      padding: 0 20rpx;  
      margin-left: 15rpx;  
    }  
  }  
</style>
```

4. 在 article/details.vue 页面导入主体内容需要的 markdown 样式文件:

将 03-配套资料\common\css\github-xxx.css，复制到 /common/css 目录下

```
<style lang="scss">  
  @import url('@/common/css/github-min.css');  
  @import url('@/common/css/github-markdown.css');
```

创建文章详情相关接口

1. EasyMock 创建查询文章详情数据接口

- 描述: 查询文章详情数据接口
- URL: `/article/api/article/{articleId}`
- 请求方式: GET
- mock.js 语法:

```
{  
  "code": 20000,  
  "message": "成功",  
  "data": {  
    "id": "10",  
    "userId": "1",  
    "nickName": "@cname",  
    "userImage|1": [  
      null,  
      "https://fuss10.elemecdn.com/e/5d/4a731a90594a4af544c0c25941171jpeg.jpeg"  
    ],  
    "title": "@csentence", // 标题  
    "summary": "@csentence(50, 100)",  
    "viewCount": "@integer(0, 100)",  
    "thumbup": "@integer(0, 10)",  
    "createDate": "@datetime",  
    "updateDate": "@datetime", // 详情显示的时间  
    "ispublic|1": [0, 1], // 0: 不公开 1: 公开  
    "labelIds|1-5": ['@integer(10, 24)'], // 随机产生1到5个元素的数字数组，数字取值10到24间
```

```
"labelList|3": [{ // 所属标签集合
  "id|+1": 10,
  "name": "@word" // 标签名
}],
"imageUrl|1": [
  null,
  "https://img.alicdn.com/bao/uploaded/i2/3603079088/01CN01rGCKfb2H0M107Lj45_!!0-
item_pic.jpg"
],
"mdContent": "# 第二章 初始化项目\n![image20200509092757510.png]
(https://mengxuegu.oss-cn-
shenzhen.aliyuncs.com/article/20200729/4882eecb1bb44802b2752e74eef0e7a9.png)\n## 重命名
项目\n\n1. 将目录名 `vue-admin-template-master` 重命名为 `mengxuegu-blog-admin` \n\n2. 将
`mengxuegu-blog-admin/package.json` 中的 `name` 值改为 `mengxuegu-blog-admin`
1\n\n``json\n{\n  \"name\": \"mengxuegu-blog-admin\", \n  \"version\":
\"1.0.1\", \n  \"description\": \"mengxuegu blog\", \n  \"author\":
\"mengxuegu.com\", \n  // ... \n}\n\n3. 将 `mengxuegu-blog-admin/package-
lock.jsom` 中的 `name` 值改为 `mengxuegu-blog-admin` 2\n\n``json\n{\n  \"name\":
\"mengxuegu-blog-admin\", \n  \"version\": \"1.0.1\", \n  \"lockfileVersion\":
1, \n}\n\n``\n\n## Eslint 语法规则检查\n\nESLint 是一个用来按照规则给出报告的代码检测工具，使用
它可以避免低级错误和统一代码的风格，也保证了代码的可读性。 \n\n### 取消 ESLint 校验\n\n在工程
根目录下的 `vue.config.js` 中将 lintOnSave 指定为 false 即可。 \n\n``js\nlintOnSave:
false, // process.env.NODE_ENV === 'development', \n\n``\n",
  "htmlContent": "<h1><a id=\"__0\"></a>第二章 初始化项目</h1>\n<p><img
src=\"https://mengxuegu.oss-cn-
shenzhen.aliyuncs.com/article/20200729/4882eecb1bb44802b2752e74eef0e7a9.png\"
alt=\"image20200509092757510.png\" /></p>\n<h2><a id=\"_2\"></a>重命名项目
</h2>\n<ol>\n<li>\n<p>将目录名 <code>vue-admin-template-master</code> 重命名为
<code>mengxuegu-blog-admin</code></p>\n</li>\n<li>\n<p>将 <code>mengxuegu-blog-
admin/package.json</code> 中的 <code>name</code> 值改为 <code>mengxuegu-blog-
admin</code> 1</p>\n</li>\n</ol>\n<pre><div class=\"hljs\"><code class=\"lang-json\">
{\n  <span class=\"hljs-attr\">\"name\"</span>: <span class=\"hljs-
string\">\"mengxuegu-blog-admin\"</span>, \n  <span class=\"hljs-attr\">\"version\"</span>: <span class=\"hljs-string\">\"1.0.1\"</span>, \n  <span class=\"hljs-attr\">\"description\"</span>: <span class=\"hljs-string\">\"mengxuegu blog\"</span>, \n  <span class=\"hljs-attr\">\"author\"</span>: <span class=\"hljs-string\">\"mengxuegu.com\"</span>, \n  // ... \n}\n</code></div></pre>\n<ol
start=\"3\">\n<li>将 <code>mengxuegu-blog-admin/package-lock.jsom</code> 中的
<code>name</code> 值改为 <code>mengxuegu-blog-admin</code> 2</li>\n</ol>\n<pre><div
class=\"hljs\"><code class=\"lang-json\">{\n  <span class=\"hljs-attr\">\"name\"</span>: <span class=\"hljs-string\">\"mengxuegu-blog-admin\"</span>, \n  <span class=\"hljs-attr\">\"version\"</span>: <span class=\"hljs-string\">\"1.0.1\"</span>, \n  <span class=\"hljs-attr\">\"lockfileVersion\"</span>: <span class=\"hljs-number\">1</span>, \n}\n</code></div></pre>\n<h2><a id=\"Eslint__28\"></a>ESLint 语法规则检查</h2>\n<p>ESLint 是一个用来按照规则给出报告的代码检测工具，使用它可以避免低级错误和统一代码的风格，也保证了代码的可读性。</p>\n<h3><a id=\"_ESLint__32\"></a>取消 ESLint 校验</h3>\n<p>在工程根目录下的 <code>vue.config.js</code> 中将 lintOnSave 指定为 false 即可。</p>\n<pre><div class=\"hljs\"><code class=\"lang-js\">lintOnSave: <span class=\"hljs-literal\">false</span>, <span class=\"hljs-comment\">// process.env.NODE_ENV === 'development',</span>\n</code></div></pre>\n"
}
```


2. EasyMock 创建文章评论列表数据接口

- 描述：查询文章评论列表数据接口
- URL: `/article/api/comment/list/{articleId}`
- 请求方式: `GET`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "成功",
  "data|2-5": [{
    "id|+1": 1,
    "parentId": "-1", //父评论id
    "userId|+1": 1, // 评论者id
    "nickName": "@cname",
    "userImage|1": [
      null,
      "https://fuss10.elemecdn.com/e/5d/4a731a90594a4af544c0c25941171jpeg.jpeg"
    ],
    "articleId": "10",
    "content": "@csentence", // 评论内容
    "createDate": "@datetime"
  }]
}
```

3. EasyMock 创建新增文章评论接口数据接口

- 描述：新增文章评论接口
- URL: `/article/comment`
- 请求方式: `POST`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "新增成功"
}
```

封装文章详情接口方法

1. 在 `/api/article.js` 文件封装调用数据接口方法

```
// 通过文章id查询文章详情

getArticleById(articleId) {
```

```
        return request({
          url: `/article/api/article/${articleId}`,
          method: 'GET'
        })
      },

      // 通过文章id查询文章评论列表
      getArticleCommentById(articleId) {
        return request({
          url: `/article/api/comment/list/${articleId}`,
          method: 'GET'
        })
      },

      // 新增文章评论
      addArticleComment(data) {
        return request({
          url: `/article/comment`,
          method: 'POST',
          data
        })
      },
    },
  },
}
```

调用数据接口渲染页面

1. 导入article.js接口文件，查询详情和评论列表数据

```
<script>
  import api from '@api/article.js'

  export default {
    data() {
      return {
        id: null, //文章id
        detailData: {}, // 详情数据
        commentList: [], // 评论列表
      }
    },

    //option为object类型，会序列化上个页面传递的参数
    onLoad(option) {
      // 文章id
      this.id = option.id
      // 查询详情
      this.getDetails();
      // 评论列表
      this.loadCommentList();
    },

    methods: {
```

```
// 查询详情
async getDetails() {
  const {data} = await api.getArticleById(this.id)
  // 将课程名称赋值导航标题
  uni.setNavigationBarTitle({
    title: data.title
  })
  // 图片在 小程序中太大显示 (/gi 全文查找<img、忽略大小写)
  data.htmlContent = data.htmlContent.replace(
    /\<img/gi, '<img style="width:100%; height: auto;" ' )
  this.detailData = data
},

//获取评论列表
async loadCommentList(){
  const {data} = await api.getArticleCommentById(this.id)
  this.commentList = data
},

},
}
</script>
```

2. 页面模板动态渲染数据

```
<template>
  <view>
    <!-- 所属标签 -->
    <view class="tag-list row">
      <uni-tag class="tag-view"
        v-for="(item,index) in detailData.labellist" :key="index" :text="item.name"
        :inverted="true" :circle="true" type="primary" size="small" />
    </view>
    <!-- 标题与内容 -->
    <view class="content-main">
      <text class="title">
        {{ detailData.title }}
      </text>
      <view class="info">
        <view class="author center">
          <image :src="detailData.userImage || '/static/logo.png'"></image>
          <text>{{ detailData.nickName}}</text>
        </view>
        <text> · {{ $util.dateFormat(detailData.updateDate) }}</text>
        <text> · {{detailData.viewCount}}人阅读</text>
      </view>
    <!-- 主体内容 -->
    <!-- #ifdef MP-->
    <!-- 小程序端使用rich-text组件的nodes渲染html代码， 只有百度小程序才可以复制 -->
    <rich-text class="markdown-body" selectable="true">
```

```
        :nodes="detailData.htmlContent"></rich-text>
      <!--#endif-->
      <!-- #ifndef MP-->
      <text class="markdown-body" selectable="true"
        v-html="detailData.htmlContent" ></text>
      <!--#endif-->
    </view>

    <!-- 热门评论 -->
    <view class="footer">
      <!-- 评论 -->
      <view class="comment">
        <view class="footer-header">热门评论</view>
        <view class="comment-item row"
          v-for="(item, index) in commentList" :key="index"
        >
          <image :src="item.userImage || '/static/logo.png'"></image>
          <view class="comment-right column">
            <view class="info space-between center">
              <text>{{item.nickName}}</text>
              <text>{{ $util.dateFormat(item.createDate) }}</text>
            </view>
            <text class="content">{{item.content}}</text>
          </view>
        </view>
      </view>
    </view>

    <view class="bottom center" @touchmove.stop.prevent="()=>{}">
      <textarea class="textarea" placeholder="有何高见，展开讲讲..." />
      <button size="mini" class="btn" >提交</button>
    </view>

    <!-- 分享 -->
    <mxg-share ref="share"
      :shareData="{title: detailData.title, mainImage: detailData.imageUrl}">
    </mxg-share>

  </view>
</template>
```

分享文章功能

1. 检查引用分享组件元素上是否有属性 `ref="share"`

```
<mxg-share ref="share"
  :shareData="{title: detailData.title, mainImage: detailData.imageUrl}">
</mxg-share>
```

2. 分享逻辑实现，下面两个都是页面生命周期函数，与 onLoad 同级：

```
// 点击导航按钮触发
onNavigationBarButtonTap(e) {
  // App端分享
  if(e.type === 'share') {
    this.$refs.share.showHandler()
  }
},

// 小程序端分享给好友
onShareAppMessage(res) {
  return {
    title: this.detailData.title,
    path: this.$util.routePath()
  }
},
```

3. 当前文章可能没有图片，所以要修改分享组件 **mxg-share.vue** 中的监听器，判断不为空才取值，因为如果图片为空会报错

```
watch: {
  shareData(newVal) {
    if(newVal.mainImage) this.image = newVal.mainImage
    if(newVal.title) this.title = newVal.title
    this.href = this.$env.HOST_H5 + this.$util.routePath()
  }
},
```

评论功能实现

需求：

1. 双向绑定获取评论框输入的内容 `v-model="content"` ；

禁用评论框，进入页面第1次点击评论框时，先判断是否已经登录，登录过取消禁用和获取焦点，未登录跳转登录页。

```
<textarea v-model="content"
  :disabled="disabled" :focus="focus" @click="clickComment"
  class="textarea" placeholder="有何高见，展开讲讲..." />
```

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

```
    return {
      content: '', //评论内容
      disabled: true, // 默认禁用评论框，只有登录过才可点击
      focus: false, //获取评论框焦点
    }
  },

  methods: {
    // 点击评论框，判断是否登录
    clickComment() {
      //如果不是禁用状态，直接结束
      if(!this.disabled) return

      // 判断是否已登录，没有登录过：方法中会跳转到登录页
      const isLogin = this.$util.isLogin()
      if(isLogin) {
        // 登录过，取消不可输入
        this.disabled = false
        this.$nextTick(()=>{
          // 获取焦点
          this.focus = true
        })
      }
    },
  },
},
}
</script>
```

2. 提交评论内容

```
<!-- 未登录或输入框为空时，禁用按钮 -->
<button :disabled="disabled || !content.trim()"
  size="mini" class="btn" @click="addComment">提交</button>
```

```
<script>
import api from '@api/article.js'

export default {
  data() {
    return {
      content: '', //评论内容
      disabled: true, // 默认禁用评论框，只有登录过才可点击
      focus: false, //获取评论框焦点
    }
  },

  methods: {

    // 提交评论信息
```

```
async addComment() {  
  const content = this.content.trim()  
  if(!content) return  
  uni.showLoading({title: '提交中', mask: true})  
  
  const data = {content, articleId: this.id}  
  const res = await api.addArticleComment(data)  
  
  uni.hideLoading()  
  if(res.code === 20000) {  
    // 重新查询评论列表  
    this.loadCommentList()  
    this.$util.msg('评论成功', {icon: 'success'})  
    //清除评论框内容  
    this.content = ''  
  }else {  
    this.$util.msg('评论失败, 请重试!')  
  }  
}  
  
},  
}  
</script>
```

问答详情页实现



ttafgp

iwky

jxlyx

年例能热走连律风会高专派反技
听。

胡秀兰 · 1979年02月24日

参考官网配置：

<https://cli.vuejs.org/zh/guid...>

需要设置BASE_URL

```
data () { return { publicPath: process
```

然后

```
<img :src=~`${publicPath}my-image.png`">
```

2个回答



武都

2004年07月26日

联织直值变事三得学现干目候部区为。



江军

2011年05月06日

改成相对路径试试

```
publicPath: ". /"
```

关注问题

回答问题

ttafgp

iwky

jxlyx

年例能热走连律风会高专派反技
听。

胡秀兰 · 1979年02月24日

参考官网配置：

<https://cli.vuejs.org/zh/guid...>

需要设置BASE_URL

```
data () { return { publicPath: process
```

然后

```
<img :src=~`${publicPath}my-image.png`">
```



回答问题

提交

有何高见，展开讲讲...

创建配置问答详情页

1. 创建问答详情页文件 /pages/question/details.vue
2. 配置page.json

```
{
  "path": "pages/article/details",
  "style": {
    "navigationBarTitleText": "",
    "backgroundColor": "#FFFFFF", //回弹颜色
    "app-plus": {
      "titleNView": { //导航配置
        // #ifdef APP-PLUS
        "buttons": [
          {
            "type": "share" // 分享按钮
          }
        ]
        // #endif
      }
    }
  }
}
```



```
}  
},
```

3. 在 /components/common/question-item.vue 添加点击事件跳转问答详情页

```
<view class="question-item" @click="navTo(`/pages/question/details?id=${item.id}`)">
```

编写静态页面代码

1. 页面模板代码

```
<template>  
  <view >  
    <!-- 所属标签 -->  
    <view class="tag-list row">  
      <uni-tag class="tag-view" text="vue.js" :inverted="true" :circle="true"  
type="primary" size="small" />  
      <uni-tag class="tag-view" text="html" :inverted="true" :circle="true"  
type="primary" size="small" />  
    </view>  
  
    <!-- 标题与内容 -->  
    <view class="content-main">  
      <text class="title">  
        问答列表页面功能实现  
      </text>  
      <view class="info">  
        <view class="author center">  
          <image src="/static/logo.png"></image>  
          <text>梦老师</text>  
        </view>  
        <text> · 2023-12-30</text>  
      </view>  
  
      <!-- 主体内容 -->  
      <!-- #ifdef MP-->  
      <!-- 小程序端使用rich-text组件的nodes渲染html代码，  
selectable="true" 只有百度小程序才可以复制 -->  
      <rich-text class="markdown-body" selectable="true"  
        nodes="<div style='color: red'>渲染html代码</div>"></rich-text>  
      <!--#endif-->  
      <!-- #ifndef MP-->  
      <text class="markdown-body" selectable="true"  
        v-html="`<div style='color: red'>渲染html代码</div>`"></text>  
      <!--#endif-->  
    </view>  
  
    <!-- 精彩回答 -->
```

```
<view class="footer">
  <view class="comment">
    <view class="footer-header">2个回答</view>
    <view class="comment-item row">
      <image src="/static/logo.png" ></image>
      <view class="comment-right column">
        <view class="info space-between center">
          <text>梦学谷</text>
          <text>2008-12-12</text>
        </view>
        <text class="content">用户权限中的内容的错误以及访问串行数据</text>
      </view>
    </view>
    <view class="comment-item row">
      <image src="/static/logo.png" ></image>
      <view class="comment-right column">
        <view class="info space-between center">
          <text>梦小二</text>
          <text>刚刚</text>
        </view>
        <text class="content">好文章值得学习! </text>
      </view>
    </view>
  </view>
</view>

<!-- 底部按钮 -->
<view class="question-option row">
  <text class="one iconfont icon-jiaguanzhu">关注问题</text>
  <!-- <text class="one grey">已关注问题</text> -->
  <text class="one iconfont icon-edit">回答问题</text>
</view>

<!-- 回答问题输入框 -->
<view v-if="false" class="answer-box" @touchmove.stop.prevent="()=>{}">
  <!-- @touchend.prevent="()=>{}" 禁止一点击就收起键盘 -->
  <view class="title center" @touchend.prevent="()=>{}">
    <view class="one">
      <!-- one样式占用太宽了，只针对x按钮绑定事件 -->
      <text class="iconfont icon-close"></text>
    </view>
    <text class="one">回答问题</text>
    <button class="btn" type="primary" size="mini">提交</button>
  </view>
  <textarea class="textarea" focus maxlength="200"
    placeholder="有何高见，展开讲讲..." />
</view>

<!-- 分享 -->
<mxg-share ref="share"></mxg-share>
</view>
</template>
```

2. 页面样式代码

```
<style lang="scss">
  @import url('@common/css/github-min.css');
  @import url('@common/css/github-markdown.css');

  .tag-list {
    // 一行排列不下，换行
    flex-wrap: wrap;
    padding: 10rpx 25rpx;
    font-size: 14px;
    background-color: #ffffff;
    .tag-view {
      margin-top: 15rpx;
      margin-right: 20rpx;
    }
  }

  .content-main {
    padding: 25rpx;
    .title {
      font-size: 45rpx;
      line-height: 55rpx;
      font-weight: bold;
    }
    .info {
      display: flex;
      align-items: center;
      margin: 30rpx 0;
      .author {
        font-size: 30rpx;
        color: #303133;
        image {
          width: 45rpx;
          height: 45rpx;
          border-radius: 100%;
          margin-right: 10rpx;
        }
      }
      &>text {
        margin-left: 10rpx;
        font-size: 25rpx;
        color: #999;
      }
    }
  }

  .footer {
    background-color: #F1F1F1;
    padding-top: 10rpx;
    /* 标题 */
    .footer-header {
      font-size: 30rpx;
    }
  }
}
```

```
color: #303133;
font-weight: bold;
padding: 25rpx;
&:before{
    content: '';
    width: 0;
    height: 40rpx;
    margin-right: 25rpx;
    border-left: 6rpx solid $mxg-color-primary;
}
}
}

/* 回答列表 */
.comment {
    background-color: #FFFFFF;
    margin-top: 10rpx;
    // 最下方有评论按钮,
    padding-bottom: 120rpx;
    .comment-item {
        padding: 20rpx 25rpx;
        image{
            width: 50rpx;
            height: 50rpx;
            border-radius: 50rpx;
            margin-right: 20rpx;
        }

        .comment-right{
            width: calc(100% - 80rpx);
            font-size: 25rpx;
            color: $mxg-text-color-grey;
            .content{
                font-size: 30rpx;
                color: $mxg-text-color-black;
                margin: 10rpx 0;
            }
        }
    }
}

.question-option {
    position: fixed;
    left: 0;
    right: 0;
    bottom: 0;
    z-index: 99;
    text-align: center;
    border-top: $mxg-underline;
    background-color: $mxg-color-grey;
    text {
        color: $mxg-text-color-blue;

        font-size: 30rpx;
```

```
        font-weight: bold;
        line-height: 90rpx;
        &:first-child {
            border-right: $mxg-underline;
        }
    }
    .grey {
        color: $mxg-text-color-grey
    }
}

/* 底部 */
.answer-box {
    z-index: 100;
    position: fixed;
    left: 0;
    right: 0;
    bottom: 0;
    padding: 20rpx;
    background-color: #FFFFFF;
    border-radius: 20rpx 20rpx 0 0;
    box-shadow: 0 0 5px $mxg-text-color-grey;
    .title {
        font-size: 35rpx;
        font-weight: bold;
        padding: 20rpx 0;
    }
    .iconfont {
        padding: 20rpx;
        color: $mxg-color-primary;
    }
    .btn {
        padding: 0 20rpx;
        margin-left: 15rpx;
    }
    .textarea {
        height: 350rpx;
        font-size: 30rpx;
        padding: 15rpx 20rpx;
        width: 100%;
        background-color: #F8F9FB;
    }
}
</style>
```

创建问答详情相关接口

1. EasyMock 创建查询问题详情数据接口

- 描述：查询问题详情数据接口
- URL: `/question/api/question/{questionId}`
- 请求方式: `GET`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "成功",
  "data": {
    "id": "10",
    "userId": "1",
    "nickName": "@cname",
    "userImage": "https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif",
    "title": "@csentence", // 标题
    "viewCount": "@integer(0,100)",
    "reply": "@integer(0,3)",
    "star|1": [0, 1], // 0 未关注, 1已关注
    "status|1": [1, 2], //1: 未解决, 2: 已解决
    "createDate": "@datetime",
    "updateDate": "@datetime", // 详情显示的时间
    "labelIds|1-5": ['@integer(10, 24)'], //随机产生1到5个元素的数字数组, 数字取值10到24间
    "labelList|3": [{ // 所属标签集合
      "id|+1": 10,
      "name": "@word" // 标签名
    }],
    "mdContent": '参考官网配置: [https://cli.vuejs.org/zh/guid...]  
(https://cli.vuejs.org/zh/guide/html-and-static-assets.html#public-文件夹)需要设置  
`BASE_URL` ``jsdata () { return { publicPath: process.env.BASE_URL }}``然后  
``html``,  
  
    "htmlContent": '<p>参考官网配置: <br /><a href="https://cli.vuejs.org/zh/guide/html-  
and-static-assets.html#public-%E6%96%87%E4%B6%E5%A4%B9"  
target="_blank">https://cli.vuejs.org/zh/guid...</a></p><p>需要设置<code>BASE_URL</code>  
</p><pre><div class="hljs"><code class="lang-js">data () { <span class="hljs-  
keyword">return</span> { <span class="hljs-attr">publicPath</span>:  
process.env.BASE_URL }</code></div></pre><p>然后</p><pre><div class="hljs"><code  
class="lang-html"><span class="hljs-tag">&lt;<span class="hljs-name">img</span> <span  
class="hljs-attr">:src</span>=<span class="hljs-string">"${publicPath}my-image.png`"  
</span>&gt;</span></code></div></pre>'
  }
}
```

2. EasyMock 创建通过问题ID查询所有回答接口

- 描述：通过问题ID查询所有回答接口
- URL: `/question/api/reply/list/{questionId}`
- 请求方式: `GET`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "成功",
  "data|2-5": [{
    "id|+1": 1,
    "parentId": "-1", //父评论id
    "userId|+1": 1, // 评论者id
    "nickName": "@cname",
    "userImage|1": [
      null,
      "https://fuss10.elemecdn.com/e/5d/4a731a90594a4af544c0c25941171jpeg.jpeg"
    ],
    "questionId": "10",
    "mdContent|+1": ['@csentence', '改成相对路径试试`shpublicPath: "/"`'],
    "htmlContent|+1": ['@csentence', '<p>改成相对路径试试</p><pre><div class="hljs">
<code class="lang-sh">publicPath:<span class="hljs-string">"/"</span></code></div>
</pre>'],
    "createDate": "@datetime"
  }]
}
```

3. EasyMock 创建新增回答数据接口

- 描述：新增回答数据接口
- URL: `/question/reply`
- 请求方式: `POST`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "新增成功"
}
```

4. EasyMock 创建 关注问题接口（后台会更新 "star"值 0 未关注，1已关注）

- 描述：关注问题接口
- URL: `/question/question/star/{questionId}`
- 请求方式: `PUT`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "成功"
}
```

封装问答详情接口方法

1. 在 `/api/question.js` 文件封装调用数据接口方法

```
// 通过问题id查询问题详情
getQuestionById(questionId) {
  return request({
    url: `/question/api/question/${questionId}`,
    method: 'GET'
  })
},

// 通过问题id查询问题回答列表
getReplyById(questionId) {
  return request({
    url: `/question/api/reply/list/${questionId}`,
    method: 'GET'
  })
},

// 新增回答
addReply(data) {
  return request({
    url: `/question/reply`,
    method: 'POST',
    data
  })
},

// 关注问题接口
starQuestion(questionId) {
  return request({
    url: `/question/question/star/${questionId}`,
    method: 'PUT'
  })
},
```

调用数据接口渲染页面

1. 导入question.js接口文件，查询详情和回答列表数据

```
<script>
import api from '@api/question.js'

export default {
  data() {
    return {
      id: null, //问题id
      data: {}, // 问题数据
    }
  }
}
```



```
        replyList: [], // 回答列表
      },
    },
    onLoad(option) {
      // 文章id
      this.id = option.id
      // 查询问题
      this.getQuestion();
      // 回答列表
      this.loadReplyList();
    },
    methods: {

      // 查询问题
      async getQuestion() {
        const {data} = await api.getQuestionById(this.id)
        // 图片在 小程序中太大显示 (/gi 全文查找<img、忽略大小写)
        data.htmlContent = data.htmlContent.replace(
          /\<img/gi, '<img style="width:100%; height: auto;" ' )
        this.data = data
      },

      //获取回答列表
      async loadReplyList(){
        const {data} = await api.getReplyById(this.id)
        this.replyList = data
      },
    },
  },
</script>
```

2. 页面模板动态渲染数据

```
<template>
  <view>
    <!-- 所属标签 -->
    <view class="tag-list row">
      <uni-tag class="tag-view"
        v-for="(item,index) in data.labellist" :key="index" :text="item.name"
        :inverted="true" :circle="true" type="primary" size="small" />
    </view>

    <!-- 标题与内容 -->
    <view class="content-main">
      <text class="title">
        {{data.title}}
      </text>
      <view class="info">
```

```
<view class="author center">
  <image :src="data.userImage || '/static/logo.png'"></image>
  <text>{{data.nickName}}</text>
</view>
<text> · {{ $util.dateFormat(data.createDate) }}</text>
</view>
<!-- 主体内容 -->
<!-- #ifdef MP-->
<!-- 小程序端使用rich-text组件的nodes渲染html代码,
      selectable="true" 只有百度小程序才可以复制 -->
<rich-text class="markdown-body" selectable="true"
      :nodes="data.htmlContent"></rich-text>
<!--#endif-->
<!-- #ifndef MP-->
<text class="markdown-body" selectable="true"
      v-html="data.htmlContent" ></text>
<!--#endif-->
</view>

<!-- 精彩回答 -->
<view class="footer">
  <view class="comment">
    <view class="footer-header">{{data.reply}}个回答</view>
    <view class="comment-item row"
      v-for="(item, index) in replyList" :key="index">
      <image :src="item.userImage || '/static/logo.png'"></image>
      <view class="comment-right column">
        <view class="info space-between center">
          <text>{{item.nickName}}</text>
          <text>{{ $util.dateFormat(item.createDate) }}</text>
        </view>
        <!-- #ifdef MP-->
        <rich-text class="content markdown-body" selectable="true"
          :nodes="item.htmlContent"></rich-text>
        <!--#endif-->
        <!-- #ifndef MP-->
        <text class="content markdown-body" selectable="true"
          v-html="item.htmlContent" ></text>
        <!--#endif-->
      </view>
    </view>
  </view>
</view>

<!-- 底部按钮 -->
<view class="question-option row">
  <text v-if="data.star" class="one iconfont icon-jiaguanzhu">关注问题</text>
  <text v-else class="one grey">已关注问题</text>
  <text class="one iconfont icon-edit">回答问题</text>
</view>

<!-- 回答问题输入框 -->

<view v-if="false" class="answer-box" @touchmove.stop.prevent="()=>{}">
```

```
<!-- @touchend.prevent="()=>{}" 禁止一点击就收起键盘 -->
<view class="title center" @touchend.prevent="()=>{}">
  <text class="one iconfont icon-close"></text>
  <text class="one">回答问题</text>
  <button class="btn" type="primary" size="mini">提交</button>
</view>
<textarea class="textarea" maxlength="200"
  placeholder="有何高见，展开讲讲..." />
</view>

<!-- 分享 -->
<mxg-share ref="share" :shareData="{title: data.title}"></mxg-share>
</view>
</template>
```

分享文章功能

1. 检查引用分享组件元素上是否有属性 `ref="share"`

```
<mxg-share ref="share" :shareData="{title: data.title}"></mxg-share>
```

2. 分享逻辑实现，下面两个都是页面生命周期函数，与 `onLoad` 同级：

```
// 点击导航按钮触发
onNavigationBarButtonTap(e) {
  // App端分享
  if(e.type === 'share') {
    this.$refs.share.showHandler()
  }
},

// 小程序端分享给好友
onShareAppMessage(res) {
  return {
    title: this.data.title,
    path: this.$util.routePath()
  }
},
```

回答功能实现

需求：

1. 显示和隐藏回答窗口

```
<script>
  export default {
    data() {
      return {
        showAnswer: false, // 显示隐藏回答输入框
        content: '', // 评论内容
      },
    },
    methods: {
      // 显示隐藏回答输入框
      showAnswerHandler(){
        // 判断是否已登录，没有登录过：方法中会跳转到登录页
        const isLogin = this.$util.isLogin()
        if(isLogin) this.showAnswer = !this.showAnswer
      },
    }
  }
</script>
```

- 回答问题 按钮绑定点击事件函数 @click="showAnswerHandler" ,
- 回答问题输入框判断显示隐藏 v-if="showAnswer"
- 左上角 X 绑定触摸事件函数（不能click事件） @touchend.prevent="showAnswerHandler"
- 获取输入框内容 v-model="content"

```
<!-- 底部按钮 -->
<view class="question-option row">

  <text class="one iconfont icon-edit" @click="showAnswerHandler">回答问题</text>
</view>

<!-- 回答问题输入框 -->
<view v-if="showAnswer" class="answer-box" @touchmove.stop.prevent="()=>{}">
  <!-- @touchend.prevent="()=>{}" 禁止一点击就收起键盘 -->
  <view class="title center" @touchend.prevent="()=>{}">
    <view class="one">
      <!-- one样式占用太宽了，只针对按钮绑定事件， -->
      <text class="iconfont icon-close" @touchend.prevent="showAnswerHandler" >
    </text>
    </view>
  </view>

  <!--省略-->
</view>
<textarea v-model="content"
  class="textarea" focus maxlength="200" placeholder="有何高见，展开讲讲..." />
</view>
```

2. 动态获取焦点 :focus="focus" , 提交按钮绑定事件 @touchend.prevent="addReply"

```
<!-- 动态获取焦点 -->
<textarea :focus="focus" v-model="content"
          class="textarea" maxlength="200" placeholder="有何高见，展开讲讲..." />

<!-- 未输入禁用按钮 -->
<button :disabled="!content" @touchend.prevent="addReply"
        class="btn" type="primary" size="mini">提交</button>
```

```
<script>
  import api from '@api/question.js'

  export default {
    data() {
      return {
        showAnswer: false, // 显示隐藏回答输入框
        content: '', // 评论内容
        focus: true, // 默认获取焦点，点击提交失去焦点
      }
    },

    methods: {

      // 提交回答内容
      async addReply() {
        const content = this.content.trim()
        if(!content) return

        this.focus = false // 失去焦点
        uni.showLoading({title: '提交中', mask: true})

        const data = {
          mdContent: content,
          htmlContent: content,
          questionId: this.id
        }
        const res = await api.addReply(data)

        uni.hideLoading()
        if(res.code === 20000) {
          this.showAnswer = false // 隐藏弹窗
          // 重新查询回答列表
          this.loadReplyList()
          this.$util.msg('回答成功', {icon: 'success'})
          // 清除输入框内容
          this.content = ''
        } else {
          this.$util.msg('回答成功，请重试!')
        }
      },

    },
  },
}
```

</script>

关注问题功能实现

1. 针对 关注问题 按钮绑定点击事件

```
<view class="question-option row">
  <text v-if="data.star" @click="starQuestionHandler" class="one iconfont icon-
jiaguanzhu">关注问题</text>
  <text v-else @click="starQuestionHandler" class="one grey">已关注问题</text>
  <text class="one iconfont icon-edit" @click="showAnswerHandler">回答问题</text>
</view>
```

2. 调用关注问题接口

```
// 关注问题
async starQuestionHandler() {
  uni.showLoading()
  const res = await api.starQuestion(this.id)
  uni.hideLoading()
  if(res.code === 20000) {
    // 针对原关注状态取反
    this.data.star = !this.data.star
  }else {
    this.$util.msg('关注失败')
  }
},
```

我的页面实现



配置page.json和导入彩色图标

1. pages.json 配置我的页面

```
{
  "path": "pages/my/my",
  "style": {
    "navigationBarTitleText": "我的",
    "navigationBarTextStyle": "white",
    "navigationBarBackgroundColor": "#345DC2",
    "backgroundColorTop": "#345DC2", // 头部回弹色
    "backgroundColorBottom": "#FFFFFF", // 底部回弹色
    "app-plus": {
      "scrollIndicator": "none", // app隐藏页面滚动条
      "titleView": {
        "type": "transparent" // app/h5滚动透明渐变
      }
    }
  }
},
```

2. 在 App.vue 文件导入彩色图标文件

```
/* #ifndef APP-NVUE */
/* 彩色的图标*/
@import url("@/static/icon/iconfont-app-icon.css");
/* #endif */
```

编写我的页面效果

1. 页面模板

```
<template>
  <view class="my-page">
    <!-- #ifndef MP -->
    <!-- 头部空出间隙 -->
    <view class="status_bar"></view>
    <!-- #endif -->

    <!-- 头部用户信息 -->
    <view class="my-header">
      <view class="header-content space-between center">
        <view class="left row center">
          <image class="header-image" src="/static/logo.png"></image>
          <view v-if="true" class="header-info column" >
            <text class="nickname">
              梦学谷
            </text>
            <text class="username">
              用户名: mengxuegu
            </text>
          </view>
          <view v-else class="header-info" >
            请登录
          </view>
        </view>
        <text class="iconfont icon-right"></text>
      </view>
    </view>

    <!-- 功能列表 -->
    <mxg-list></mxg-list>

  </view>
</template>
```


2. 先创建好**功能列表**子组件 `/mxg-education-app/components/mxg-list/mxg-list.vue`
3. 编写**我的**页面样式

```
<style lang="scss">
  .status_bar {
    height: calc(var(--status-bar-height) + 48px);
    width: 100%;
    background-color: $mxg-color-primary;
  }

  .my-header {
    background-color: $mxg-color-primary;
    .header-content {
      padding: 50rpx 39rpx;
      background-color: #FFFFFF;
      border-radius: 30px 30px 0 0;
    }
    .left {
      .header-image {
        width: 150rpx;
        height: 150rpx;
        border-radius: 60px;
      }
      .header-info {
        margin-left: 30rpx;
        .nickname {
          font-size: 39rpx;
          font-weight: 500;
        }
        .username {
          font-size: 33rpx;
          color: $mxg-text-color-grey;
        }
      }
    }
  }
  /* 右侧箭头 */
  .icon-right {
    font-size: 35rpx;
    color: $mxg-text-color-grey;
  }
</style>
```

编写功能列表子组件 mxg-list.vue

1. 编写mxg-list.vue功能列表子组件模板代码：
 - 每行右侧有3种效果：右箭头>，switch开头，显示文字，图片



```
<template>
  <view class="list-box">
    <!-- 一大类列表 -->
    <view class="list">
      <!-- 每一行 -->
      <view class="list-item space-between center"
        hover-class="active" :hover-stay-time="50" >
        <view class="left center">
          <text class="mxg-icon mxg-icon-host-color"></text>
          <view class="title">我的订单</view>
        </view>
        <view class="right center">
          <!-- 右侧图标 -->
          <text class="iconfont icon-right"></text>
        </view>
      </view>

      <view class="list-item space-between center"
        hover-class="active" :hover-stay-time="50" >
        <view class="left center">
          <text class="mxg-icon mxg-icon-host-color"></text>
          <view class="title">清除应用缓存</view>
        </view>
        <view class="right center">
          <!-- 右侧文字 -->
          <text>10KB</text>
        </view>
      </view>
    </view>

    <!-- 一大类列表 -->
    <view class="list">
      <!-- 每一行 -->
      <view class="list-item space-between center"
        hover-class="active" :hover-stay-time="50" >
        <view class="left center">
          <view class="title">允许非WIFI网络播放</view>
        </view>
      </view>
    </view>
  </view>
</template>
```

```
<view class="right center">
  <!-- 右侧开关 -->
  <switch color="#345dc2" checked/>
</view>
</view>

<view class="list-item space-between center"
  hover-class="active" :hover-stay-time="50" >
  <view class="left center">
    <view class="title">头像</view>
  </view>
  <view class="right center">
    <!-- 右侧图片 -->
    <image src="/static/logo.png" lazy-load></image>
    <!-- 右侧图标 -->
    <text class="iconfont icon-right"></text>
  </view>
</view>
</view>
</view>
</template>
```

2. 功能列表子组件样式

```
<style lang="scss">
  /* 列表 */
  .list-box {
    background-color: $mxg-color-grey;
    padding-top: 20rpx;
    .list {
      background-color: #FFFFFF;
      margin-bottom: 20rpx;
      .list-item {
        padding: 26rpx 39rpx;
        border-bottom: $mxg-underline;
        .left {
          font-size: 33rpx;
          text:first-child {
            margin-right: 20rpx;
          }
        }
        .right {
          text{
            font-size: 35rpx;
            color: $mxg-text-color-grey;
            margin-left: 15rpx;
          }
          image {
            width: 120rpx;
            height: 120rpx;
            border-radius: 60px;
          }
        }
      }
    }
  }
}
```

```
    }  
    switch{  
      margin-right: -10rpx;  
    }  
  }  
}  
}  
.active {  
  background-color: #fafafa;  
}  
}  
}  
}  
</style>
```

我的页面动态渲染功能列表数据

1. 创建我的页面功能列表数据配置文件 `/config/my-list-bar.js`

采用二维数据方式，一维是列表块，二维是列表块中每个列表项

```
const rightIcon = 'iconfont icon-right'  
  
export default () => {  
  return [ // 二维数组导航  
    [  
      {  
        title: '我的订单',  
        icon: 'mxg-icon mxg-icon-host-color', // 左侧图标  
        rightIcon, // 右侧图标  
        path: '/pages/order/order', // 目标页面，配置了点击直接跳转  
        login: true, // true 登录后才可访问  
        /* event: 'setWifiPlay', // 点击触发的事件名  
        checked: false // 是否选中 */  
      },  
      {  
        title: '我的余额',  
        icon: 'mxg-icon mxg-icon-lock-color',  
        rightIcon,  
        path: '/pages/order/my-balance',  
        login: true  
      },  
      {  
        title: '我的课程',  
        icon: 'mxg-icon mxg-icon-warn',  
        rightIcon,  
        path: '/acticle',  
        login: true  
      }  
    ],  
  
    [  

```

```
{
  title: '设置',
  icon: 'mxg-icon mxg-icon-set-color',
  rightIcon,
  path: '/pages/my/setting',
},
{
  title: '意见反馈',
  icon: 'mxg-icon mxg-icon-notice-color',
  rightIcon,
  path: '/pages/my/feedback'
}
],
[
  {
    title: '关于我们',
    icon: 'mxg-icon mxg-icon-model-color',
    rightIcon,
    path: '/pages/my/about'
  }
]
]
```

2. 我的页面 my.vue 导入并引用功能列表数据,

```
<!-- 功能列表 -->
<mxg-list :list="list"></mxg-list>
</view>
</template>

<script>
import list from '@config/my-list-bar.js'

export default {
  data () {
    return {
      list(), // 功能列表数据
    },
  },
}
</script>
```

3. mxg-list.vue 接收父组件传递的数据，然后模板中动态渲染数据

```
<script>
  export default {
    props: {
      list: Array,
    },
  }
</script>
```

4. mxg-list.vue 动态渲染数据

- 有switch开头的列表项（有checked值就是），无点击态样式，其他有。
- switch 使用 `@change="clickHandler(nav)"` 获取当前状态值；并且在switch组件元素上要给一个空的点击事件函数阻止阻止节点点击事件 `@click.stop="()=>{}"`，不然会clickHandler会触发两次。

```
<template>
  <view class="list-box">
    <!-- 一大类列表 -->
    <view class="list" v-for="(item, index) in list" :key="index">
      <!-- 每一行 -->
      <view v-for="(nav, i) in item" :key="i"
        @click="clickHandler(nav)"
        :hover-class="nav.checked || nav.checked===false ? '' : 'active'"
        :hover-stay-time="50"
        class="list-item space-between center" >
        <view class="left center">
          <text v-if="nav.icon" :class="nav.icon"></text>
          <view class="title">{{ nav.title }}</view>
        </view>
        <view class="right center">
          <!-- 右侧文字 -->
          <text v-if="nav.text">{{nav.text}}</text>
          <!-- 右侧开关 @click.stop 阻止祖节点发现点击事件 -->
          <switch v-if="nav.checked || nav.checked === false"
            :checked="nav.checked"
            @click.stop="()=>{}" @change="clickHandler(nav)"
            color="#345dc2"/>
          <image v-if="nav.src" :src="nav.src" lazy-load></image>
          <!-- 右侧图标 -->
          <text v-if="nav.rightIcon" :class="nav.rightIcon"></text>
        </view>
      </view>
    </view>
  </view>
</template>
```

5. 点击列表项触发 clickHandler 方法，实现以下逻辑

- 有 path 页面路由地址，跳转目标页面，其中判断是否要求先登录
- 有 event 事件名，触发对应同名的父组件事件

```
<script>
  export default {
    props: {
      list: Array,
    },

    methods: {
      // 点击列表项触发
      clickHandler(obj) {
        // console.log('clickHandler', obj)
        if(obj.path) {
          // 跳转目标页面，判断是否要求先登录
          this.navTo(obj.path, {login: obj.login})
          return
        }
        if(obj.event) {
          // 触发绑定的事件
          this.$emit(obj.event, obj)
        }
      },
    },
  }
}
</script>
```

目前还没有实现登录，为了测试，在 mxin.js 中的 navTo 方法里注释掉未登录跳转到登录页代码

```
navTo(url, options={}){
  if(!url){
    return;
  }
  /* if(options.login && !this.$store.getters.hasLogin){
    url = '/pages/auth/login';
  } */
  uni.navigateTo({
    url
  })
},
```

设置页面实现



设置

允许非WIFI网络播放



允许非WIFI网络缓存



视频自动连续播放



清除应用缓存

1KB

创建与配置设置页面

1. 创建关于 设置 页面文件 `/pages/my/setting.vue`
2. `pages.json`配置 设置页面

```
{
  "path" : "pages/my/setting",
  "style" : {
    "navigationBarTitleText": "设置"
  }
},
```

功能列表配置文件

1. 创建设置页面功能列表配置文件: `/config/setting-list-bar.js`

注意, 要导出一个方法:

- 因为导出默认数组会被页面缓存, 而每次进入页面都应该从 `storage` 中重新获取, 需要被强制性不缓存, 则使用导出方法形式。

```
export default () => {
  return [ // 二维数组导航
    [
      {
        title: '允许非WIFI网络播放',
        event: 'setWifiPlay', // 点击触发的事件名
        // 是否选中, 注意要使用Sync同步获取方法在app和小程序才会生效
        checked: uni.getStorageSync('wifi-play') || false
      }
    ]
  ],
}
```



```
{
  title: '允许非WIFI网络缓存',
  event: 'setWifiDownload',
  checked: uni.getStorageSync('wifi-download') || false
},
{
  title: '视频自动连续播放',
  event: 'setAutoPlay',
  checked: uni.getStorageSync('auto-play') || false
}
],
[
  {
    title: '清除应用缓存',
    event: 'clearStorage', // 点击触发的事件名
    // 右侧内容
    text: uni.getStorageSync().currentSize < 1024
      ? uni.getStorageSync().currentSize + 'KB'
      : (uni.getStorageSync().currentSize / 1024).toFixed(2) + 'MB',
  },
],
]
```

设置页面功能实现

1. 设置页面模板代码

```
<template>
  <view>
    <!-- 绑定的事件是对应setting-list-bar.js中的event值 -->
    <mxg-list :list="list"
      @setWifiPlay="setWifiPlay"
      @setWifiDownload="setWifiDownload"
      @setAutoPlay="setAutoPlay"
      @clearStorage="clearStorage"></mxg-list>
  </view>
</template>

<script>
import list from '@config/setting-list-bar.js'

export default {
  data() {
    return {
      list: list() //注意调用的是方法，不要少了 ()
    }
  },
}
```

```
methods: {

  // 清除缓存
  clearStorage(obj) {
    console.log('清除缓存', obj)
    // 提示
    uni.showModal({
      title: '您确定清除应用缓存吗?',
      content: '(该操作不会删除缓存课程)',
      success: (res) => { //箭头函数
        if (res.confirm) {
          // 清空
          uni.clearStorage()
          // 清空赋值 0KB
          this.$set(obj, 'text', '0KB')
          this.$util.msg('清除成功')
        }
      }
    });
  },

  // 允许非WIFI网络播放
  setWifiPlay(obj) {
    this.saveStorage(obj, 'wifi-play')
  },

  // 允许非WIFI网络缓存
  setWifiDownload(obj) {
    this.saveStorage(obj, 'wifi-download')
  },

  // 视频自动连续播放
  setAutoPlay(obj) {
    this.saveStorage(obj, 'auto-play')
  },

  // 保存到本地
  saveStorage(obj, key) {
    const checked = !obj.checked
    this.$set(obj, 'checked', checked)
    // 保存到本地
    uni.setStorage({key, data: checked})
    console.log('修改后状态值', checked)
  }
}
</script>

<style lang="scss">
  page {
    background-color: $mxg-color-grey;
  }
}
```

</style>

视频自动播放判断

在 /pages/course/subNVue/video.nvue 的 nextPlay 方法上接收参数 **e**，判断是否自动播放下一节。

```
// 播放到当前视频末尾，播放下一节
nextPlay(e) {
  if(e && e.type === 'ended') {
    // 播放到末尾，判断是否自动下一节
    const autoPlay = uni.getStorageSync('auto-play')
    if(!autoPlay) {
      uni.showModal({
        content: '是否自动播放下个视频?',
        success: (res) => {
          if(res.confirm) {
            uni.setStorageSync('auto-play', true)
          }
          this.nextPlay()
        }
      })
      return // 结束,上面弹窗是异步的
    }
  }
  //.....
}
```

意见反馈页面实现

◀ 微信 21:58 94%



意见反馈

☒ 内容意见 ☐ 产品建议 ☐ 其他

请填写具体内容帮助我们了解您的意见和建议

填写您的QQ

填写您的微信

您的当前位置：
江西省赣州市

提交

创建与配置意见反馈页面

1. 创建关于意见反馈页面文件 `/pages/my/feedback.vue`
2. pages.json配置 意见反馈 页面

```
{
  "path" : "pages/my/feedback",
  "style" : {
    "navigationBarTitleText": "意见反馈"
  }
},
```

编写页面代码

1. 模板代码

```
<template>
```

```
<view class="feedback-box column center">
  <radio-group class="radio-group" @change="radioChange">
    <!-- 加上label, 当点击文字时也会被触发-->
    <label class="label">
      <radio :value="1" checked color="#007aff" style="transform:scale(0.7)" />
      内容意见
      <radio :value="2" color="#007aff" style="transform:scale(0.7)" />产品建议
      <radio :value="3" color="#007aff" style="transform:scale(0.7)" />其他
    </label>
  </radio-group>

  <textarea v-model="formData.content" maxlength="500" placeholder="请填写具体内容帮助我们了解您的意见和建议"/>
  <input v-model="formData.qq" type="number" maxlength="15" placeholder="填写您的QQ" />
  <input v-model="formData.weixin" type="text" maxlength="25" placeholder="填写您的微信" />

  <!-- #ifdef APP-PLUS -->
  <view class="location">
    <view>您的当前位置: </view>
    <text>{{ location || '获取信息中' }}</text>
  </view>
  <!-- #endif -->
  <button type="primary" @click="submitHandler">提交</button>
</view>
</template>
```

2. 声明属性与方法

```
<script>
  export default {
    data() {
      return {
        formData: {},
        location: '', // 当前位置
      },
    },
    methods: {
      radioChange(event) {
        //console.log('选中的value值', event.detail.value)
        this.formData.type = event.detail.value
      },
      // 提交
      submitHandler() {
      },
    },
  },
</script>
```

3. 样式代码:

```
<style lang="scss">
  page {
    background-color: $mxg-color-grey;
  }
  .feedback-box {
    padding: 0 30rpx;
  }

  .radio-group {
    margin: 30rpx 0;
    .label {
      margin-right: 30rpx;
      font-size: 30rpx;
      color: $mxg-text-color-black;
    }
  }

  textarea {
    width: 100%;
    border: 1px solid #E9E9E9;
    padding: 10rpx;
    line-height: 60rpx;
  }

  input {
    width: 100%;
    margin-top: 30rpx;
    border: 1px solid #E9E9E9;
    height: 70rpx;
    padding: 10rpx;
  }

  .location {
    margin: 50rpx 0;
    align-self: flex-start; /* 左对齐 */
    font-size: 30rpx;
    color: $mxg-text-color-grey;
    line-height: 50rpx;
    text {
      color: $mxg-text-color-black;
    }
  }

  button[type=primary] {
    width: 100%;
    margin-top: 50rpx;
    background-color: $mxg-text-color-blue;
  }

  .button-hover[type=primary] {
    width: 100%;
```

```
margin-top: 50px;  
background-color: $mxg-color-primary;  
}  
</style>
```

APP获取当前位置

参考: <https://uniapp.dcloud.io/api/location/location>

1. 在 onLoad 页面生命函数中调用 uni.getLocation 获取位置信息

```
onLoad() {  
  // APP 获取位置  
  // #ifdef APP-PLUS  
  uni.getLocation({  
    geocode: true, //解析地址信息  
    success: (res) => { // 注意箭头函数  
      // console.log('当前位置的经度: ' + res.longitude);  
      // console.log('当前位置的纬度: ' + res.latitude);  
      // #ifdef APP-PLUS  
      this.location = res.address.province + res.address.city  
      // #endif  
    }  
  })  
  // #endif  
}
```

调用接口新增反馈信息数据

创建服务接口

EasyMock 创建新增反馈信息接口

- 描述: 新增反馈信息接口
- URL: `/system/api/feedback`
- 请求方式: `POST`
- mock.js 语法:

```
{  
  "code": 20000,  
  "message": "新增成功"  
}
```

封装服务接口方法

创建 `/api/system.js` 文件，封装调用数据接口方法

```
import request from '@common/js/request.js'

export default {

  // 新增反馈信息
  addFeedback(data) {
    return request({
      url: `/system/api/feedback`,
      method: 'POST',
      data
    })
  },
}
```

页面调用接口

1. 完善提交按钮 `:disabled="loading" :loading="loading"`

```
<button :disabled="loading" :loading="loading" type="primary" @click="submitHandler">提交
</button>
```

2. 导入 `/api/system.js`，声明属性，`submitHandler`方法中校验并调用接口提交数据

```
<script>
  import api from '@api/system.js'
  export default {
    data() {
      return {
        formData: {},
        // 当前位置
        location: '',
        loading: false
      }
    },

    methods: {
      radioChange(event) {
        //console.log('选中的value值', event.detail.value)
        this.formData.type = event.detail.value
      },
    },
  },
}
```



```
// 提交
async submitHandler() {
  if(!this.formData.content
    || this.formData.content.trim().length < 10) {
    this.$util.msg('内容至少输入10个字')
    return
  }

  if( !this.formData.qq && !this.formData.weixin ) {
    this.$util.msg('QQ和微信至少填写一个')
    return
  }

  // 如果qq不为空，则校验是否合法
  if( this.formData.qq && !this.$util.checkStr(this.formData.qq, 'QQ')) {
    this.$util.msg('QQ填写有误')
    return
  }

  // 如果微信不为空，则校验是否合法
  if( this.formData.weixin
    && !/^[a-zA-Z][a-zA-Z\d_-]{5,19}$/.test(this.formData.weixin) ) {
    this.$util.msg('微信填写有误')
    return
  }

  /* console.log('反馈数据', this.formData) */

  this.loading = true
  uni.showLoading({title: '提交中', mask: true})

  const res = await api.addFeedback(this.formData)
  this.loading = false
  uni.hideLoading()
  if(!res.code === 20000) {
    this.$util.msg(res.message)
    return
  }
  uni.showModal({
    content: '您的意见反馈成功!',
    showCancel: false,
    success: () => {
      // 返回上一页
      this.navBack()
    }
  })
}

},
</script>
```

关于梦学谷页面实现

◀ 微信 21:58 94%



关于梦学谷



Version 1.0.0

梦学谷是专注于在线教育平台，聚合了优秀教师的海量精品课程。在这里，您可以在线观看视频教程；您可以随时随地向名师请教；您可以与大量志同道合的同学进行交流。

打破地域的限制，让每个立志学习，有梦想的人，都能接受优秀教师的指导和教学；同时希望给优秀的教师一个展示的平台。

梦学谷——陪你学习，伴你梦想！



Copyright © 2021 mengxuegu.com All rights reserved

1. 创建 关于梦学谷 页面文件 `/pages/my/about.vue`

2. pages.json配置 关于梦学谷

```
{
  "path" : "pages/my/about",
  "style" : {
    "navigationBarTitleText": "关于梦学谷"
  }
},
```

3. 将梦学谷logo图片 `03-配套资料\static\images\logo-black.png` 添加到项目的 `\static\images\`

4. 页面代码如下：

```
<template>
```

```
<view>
  <view class="column center">
    <!-- widthFix 高度自动变化 -->
    <image class="logo" src="/static/images/logo-black.png" lazy-load
mode="widthFix"></image>
    <text class="version">Version 1.0.0</text>
  </view>
  <view class="center-content">
    <p>梦学谷是专注于在线教育平台，聚合了优秀教师的海量精品课程。在这里，您可以在线观看视频教程；您可以随时随地向名师请教；您可以与大量志同道合的同学进行交流。</p>
    <p>打破地域的限制，让每个立志学习，有梦想的人，都能接受优秀教师的指导和教学；同时希望给优秀的教师一个展示的平台。</p>
    <p>梦学谷—陪你学习，伴你梦想！</p>
  </view>
  <view class="bottom">Copyright &copy; {{new Date().getFullYear()}} mengxuegu.com
All rights reserved</text></view>
</view>
</template>

<style lang="scss">
  page {
    background-color: $mxg-color-grey;
  }
  .logo {
    margin-top: 60rpx;
    margin-bottom: 20rpx;
    width: 280rpx;
  }
  .version {
    font-size: 30rpx;
    color: $mxg-color-primary;
    font-weight: 500;
  }
  .center-content {
    margin-top: 50rpx;
    padding: 0 30rpx;
    font-size: 30rpx;
    color: $mxg-text-color-black;
    p {
      /* 缩进2格 */
      text-indent: 2em;
      margin-bottom: 20rpx;
    }
  }
  .bottom {
    position: absolute;
    left: 0;
    right: 0;
    bottom: 20rpx;
    text-align: center;

    color: $mxg-text-color-grey;
  }
</style>
```

```
font-size: 20rpx;  
}  
</style>
```

登录页面实现



创建和配置登录页面

1. 创建登录页面文件 `pages/auth/login`
2. `pages.json`配置 登录页面

```
{  
  "path" : "pages/auth/login",  
  
  "style" : {
```

```
"navigationBarTitleText": "登录",
// 自定义导航栏
"navigationStyle": "custom",
"app-plus": {
  // 从下往上
  "animationType": "slide-in-bottom",
  // 页面回弹效果
  "bounce": "none"
}
},
},
```

3. 点击 my.vue 我的页面头部信息，跳转到登录页

```
<view class="my-header" @click="navTo('/pages/auth/login')">
```

登录页面效果实现

1. 登录页面模板代码

```
<template>
  <view class="app">
    <!-- mixin.js 中的 navBack(), 不能少了括号, 不然会传递event对象 -->
    <text class="back-btn iconfont icon-close" @click="navBack()"></text>
    <view class="left-top-sign">LOGIN</view>
    <view class="welcome">欢迎回来! </view>
    <!-- 手机号登录 -->
    <view class="input-content">
      <view class="input-item">
        <text class="tit">手机号码</text>
        <view class="row">
          <input v-model="mobile" type="number" maxlength="11"
            placeholder="请输入手机号码" placeholder-style="color: #909399"/>
        </view>
      </view>
      <view class="input-item">
        <text class="tit">验证码</text>
        <view class="row">
          <input v-model="code" type="number" maxlength="6"
            placeholder="请输入手机验证码" placeholder-style="color: #909399"/>
          <mxg-code :mobile="mobile" templateCode="SMS_19999999"></mxg-code>
        </view>
      </view>
      <button type="primary" :loading="loading" @click="login">登录</button>
    </view>

    <!-- #ifdef APP-PLUS || MP-WEIXIN -->
    <view class="other-wrapper">
      <view class="line center">
```

```
<text class="tit">社交账号登录</text>
</view>
<view class="row">
  <image @click="toProviderLogin('weixin')"
    class="icon" src="/static/share/weixin.png"></image>
  <!-- #ifdef APP-PLUS -->
  <image @click="toProviderLogin('qq')"
    class="icon" src="/static/share/qq.png"></image>
  <!-- #endif -->
</view>
</view>
<!-- #endif -->
<!-- 协议 -->
<view class="agreement center">
  <text class="iconfont icon-roundcheckfill"
    :class="{active: agreement}" @click="checkAgreement"></text>
  <text @click="checkAgreement">请认真阅读并同意</text>
  <text class="tit" @click="navTo('/pages/public/xieyi')">《用户服务协议》</text>
  <text class="tit" @click="navTo('/pages/public/yinsi')">《隐私权政策》</text>
</view>
</view>
</template>
```

2. 声明属性和方法

```
<script>
  export default{
    data(){
      return {
        mobile: '', // 用户名
        code: '', // 验证码
        agreement: true, // 是否同意协议
        loading: false, // 登录中
      }
    },
    methods: {
      //同意协议
      checkAgreement(){
        this.agreement = !this.agreement
      },
      // 手机号登录
      async login(){
      },
      // 微信、QQ提供商登录
      toProviderLogin(provider) {
      },
    }
  }
</script>
```

3. 编写样式样式（可复制老师源码中的）

```
<style lang="scss">
  .app{
    padding-top: 200rpx;
  }
  /* 关闭按钮 */
  .back-btn{
    position: absolute;
    left: 20rpx;
    top: calc(var(--status-bar-height) + 20rpx);
    z-index: 90;
    padding: 20rpx;
    font-size: 39rpx;
    color: #606266;
  }

  .left-top-sign{
    font-size: 120rpx;
    color: #f8f8f8;
  }

  .welcome{
    position: relative;
    top: -90rpx;
    padding-left: 60rpx;
    font-size: 46rpx;
    color: #555;
    text-shadow: 1px 0px 1px rgba(0,0,0,.3);
  }

  .input-content{
    padding: 0 60rpx;
  }

  .input-item{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: flex-start;
    padding-left: 30rpx;
    background: #f8f6fc;
    height: 120rpx;
    border-radius: 4px;
    margin-bottom: 50rpx;

    &:last-child{
      margin-bottom: 0;
    }
  }

  .row{
```

```
        width: 100%;
    }
    .tit{
        height: 50rpx;
        line-height: 56rpx;
        font-size: 26rpx;
        color: #606266;
    }
    input{
        flex: 1;
        height: 60rpx;
        font-size: 30rpx;
        color: #303133;
        width: 100%;
    }
}

button[type=primary] {
    background-color: $mxg-color-primary !important;
}

/* 其他登录方式 */
.other-wrapper{
    display: flex;
    flex-direction: column;
    align-items: center;
    padding-top: 20rpx;
    margin-top: 80rpx;

    .weixin {
        border: 0;
    }
    .line{
        margin-bottom: 40rpx;
        .tit{
            margin: 0 32rpx;
            font-size: 24rpx;
            color: #606266;
        }
        &:before, &:after{
            content: '';
            width: 160rpx;
            height: 0;
            border-top: 1px solid #e0e0e0;
        }
    }
    .icon{
        width: 80rpx;
        height: 80rpx;
        margin: 0 50rpx;
    }
}
```



```
.agreement{
  position: absolute;
  left: 0;
  bottom: 10rpx;
  width: 750rpx;
  height: 90rpx;
  font-size: 24rpx;
  color: #999;
  .iconfont{
    font-size: 36rpx;
    color: #ccc;
    margin-right: 8rpx;
    &.active{
      color: $mxg-color-primary;
    }
  }
  .tit{
    color: #40a2ff;
  }
}
</style>
```

4. 创建 **获取验证码** 按钮子组件：@/components/mxg-code/mxg-code.vue

5. 创建与配置协议相关页面：/pages/public/xieyi.vue 、 /pages/public/yinsi.vue

页面内容自己补文字

```
{
  "path" : "pages/public/xieyi",
  "style" : {
    "navigationBarTitleText": "用户服务协议"
  }
},
{
  "path" : "pages/public/yinsi",
  "style" : {
    "navigationBarTitleText": "隐私权政策"
  }
}
```

获取验证码功能

1. EasyMock 创建 发送短信验证码接口

- 描述：发送短信验证码
- URL： `/system/sms/code`
- 请求方式： `POST`

- o mock.js 语法:

```
{
  "code": 20000,
  "message": "发送成功"
}
```

2. 在 `/api/system.js` 文件封装调用数据接口方法

```
// 发送短信验证码
sendSmsCode(data) {
  return request({
    url: '/system/sms/code',
    method: 'POST',
    data
  })
},
```

3. 在**获取验证码** 按钮子组件 `mxg-code.vue` 中，实现发送验证码功能

```
<template>
  <view class="mxg-code center">
    <text class="code-btn" @click="sendSmsCode">
      {{codeDuration ? codeDuration + 's' : '获取验证码' }}
    </text>
  </view>
</template>

<script>
  let interval = null
  import api from '@api/system.js'

  export default {
    props: {
      mobile: { // 手机号
        type: String,
        default: '18888888888'
      },
      templateCode: { // 短信模板代码
        type: String,
        default: ''
      }
    },
    data() {
      return {
        codeDuration: null, // 倒计时时长
      }
    },
```

```
methods: {  
  // 发送短信验证码  
  async sendSmsCode() {  
  
    // 有倒计时，则已发送  
    if (this.codeDuration) {  
      uni.showModal({  
        content: `请在${this.codeDuration}秒后重试`,  
        showCancel: false  
      })  
      return  
    }  
  
    // 校验手机号  
    if (!this.$util.checkStr(this.mobile, 'mobile')) {  
      this.$util.msg('手机号码格式不正确')  
      return  
    }  
  
    // 更换手机号与原手机号一致，不允许发送  
    if (this.$store && this.mobile === this.$store.state.userInfo.mobile) {  
      this.$util.msg('新手机号与当前绑定的手机号不能相同')  
      return  
    }  
  
    try{  
      // 发送短信验证码  
      uni.showLoading({mask: true})  
      const data = {mobile: this.mobile, templateCode: this.templateCode}  
      const res = await api.sendSmsCode(data)  
      uni.hideLoading()  
      // 响应结果  
      this.$util.msg(res.message)  
      if(res.code !== 20000) return  
      // 倒计时  
      this.codeDuration = 60  
      interval = setInterval(() => {  
        this.codeDuration--  
        if (this.codeDuration === 0) {  
          if (interval) {  
            clearInterval(interval)  
            interval = null  
          }  
        }  
      }, 1000)  
    }catch(e) {  
      uni.hideLoading()  
      this.$util.msg('验证码发送失败')  
      console.log('验证码发送失败', e)  
    }  
  },  
}
```

```
}  
</script>  
  
<style lang="scss">  
  .mxg-code {  
    width: 160rpx;  
    background-color: #345dc2;  
    border-radius: 10rpx;  
  }  
  .code-btn {  
    padding: 10rpx 0;  
    font-size: 25rpx;  
    color: #FFFFFF;  
  }  
</style>
```

手机验证码登录

上面已经发送验证码，当用户输入手机号和验证码时，点击 登录 按钮调用接口进行登录

1. EasyMock 创建 登录接口

- 描述：登录接口
- URL: `/auth/login`
- 请求方式: `POST`
- mock.js 语法:

```
{  
  "code": 20000,  
  "message": "成功",  
  "data": {  
    "access_token": "@word(30)",  
    "token_type": "bearer",  
    "refresh_token": "fbdgsbscpakavfpjurbeqegmrrhghv",  
    "expires_in": "@natural", // access_token 有效时长  
    "scope": "all", // 权限范围  
    "userInfo": {  
      "uid": "9",  
      "username": "@name",  
      "mobile": "/1\d{10}/",  
      "sex|1": [0, 1], // 0男, 1女  
      "nickName": "@cname",  
      "imageUrl": "https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif"  
    },  
    "jti": "@word(20)" // access_token 唯一标签  
  },  
}
```

2. 创建 `/api/auth.js` 文件封装调用数据接口方法

```
import request from '@common/js/request.js'

export default {

  // 手机号登录
  login(data) {
    return request({
      url: '/auth/login',
      method: 'POST',
      data
    })
  },
}
```

3. 在 login.vue 方法中实现，登录成功后使用Vuex进行用户状态管理

```
<script>
import {checkStr} from '@common/js/util.js'
import api from '@api/auth.js'

export default{

  methods: {
    // 手机号登录
    async login(){
      if(!this.agreement){
        this.$util.msg('请阅读并同意用户服务及隐私协议')
        return
      }
      const {mobile, code} = this
      if(!checkStr(mobile, 'mobile')){
        this.$util.msg('请输入有效手机号码')
        return
      }
      if(!checkStr(code, 'mobileCode')){
        this.$util.msg('验证码输入错误');
        return
      }
      this.loading = true
      uni.showLoading({title: '登录中', mask: true})
      // 请求服务接口
      const res = await api.login( {mobile, code} )
      this.loading = false
      uni.hideLoading()
      if(res.code === 20000){
        this.loginSuccessCallBack(res.data)
      }else{

```

```
        this.$util.msg(res.message)
      }
    },

    // 登录成功，更新store登录状态
    loginSuccessCallBack(data) {
      this.$util.msg('登录成功')
      this.$store.commit('setToken', data)
      setTimeout(()=>{
        this.navBack()
      }, 500)
    },
  },
}
</script>
```

Vuex 管理登录状态

1. 在 `mxg-education-app` 项目根目录下，新建 `store` 目录，在此目录下新建 `index.js` 文件，代码如下：

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex) //vue的插件机制

//Vuex.Store 构造器选项
const store = new Vuex.Store({
  //存放状态
  state: {
    userInfo: {}, // 用户信息
    accessToken: '', // 访问令牌
  },

  // store的计算属性
  getters: {
    // 是否有登录
    hasLogin(state) {
      return !!state.accessToken // 取反两次把原数据转换为布尔类型，有值就是true
    },
  },

  // 更新状态值
  mutations: {
    // 更新状态数据
    setState(state, obj) {
      for(let key in obj) {
        // 每个对象的key作为状态名，value作为状态值
        state[key] = obj[key]
      }
    },
  },
})
```

```
// 更新用户登录状态
setToken(state, data) {
  // 解构属性,
  const {userInfo, access_token} = data
  // 状态赋值保存
  if(userInfo) {
    state.userInfo = userInfo
    uni.setStorageSync('userInfo', userInfo)
  }
  if(access_token) {
    state.accessToken = access_token
    uni.setStorageSync('mxgEducationToken', access_token)
  }
},

// 退出登录
logout(state) {
  // 状态清空
  state.userInfo = {}
  state.accessToken = ''
  // 移除本地数据
  uni.removeStorageSync('userInfo')
  uni.removeStorageSync('mxgEducationToken')
}

},

}))

// 导出store实例
export default store
```

2. 在 `main.js` 中导入状态管理文件。

```
// 状态管理文件
import store from './store'
// 挂载Vue实例上, this.$store 获取
Vue.prototype.$store = store

// 把 store 对象提供给“store”选项, 这可以把 store 的实例注入所有的子组件
const app = new Vue({
  store, // ++
  ...App
})
app.$mount()
```

我的页面渲染用户信息

当用户登录后, `my/my.vue` 我的页面应该渲染当前登录用户信息:

1. 在 my/my.vue 页面导入Vuex 状态相关对象，作为当前页面计算属性使用

```
<script>
import list from '@/config/my-list-bar.js'

// 1. 导入vuex相关对象
import { mapState, mapGetters } from 'vuex'
export default {
  data () {
    return {
      list,
    }
  },
  computed: {
    // 2. 解构状态作为计算属性
    ...mapState(['userInfo']),
    ...mapGetters(['hasLogin'])
  }
}
</script>
```

2. 页面渲染登录用户信息

```
<view class="my-header"
  @click="hasLogin
    ? navTo('/pages/my/user', {login: true})
    : navTo('/pages/auth/login')">
  <view class="header-content space-between center">
    <view class="left row center">
      <image class="header-image"
        :src="userInfo.imageUrl || '/static/logo.png'"></image>
      <!-- 登录了, 显示 -->
      <view v-if="hasLogin" class="header-info column" >
        <text class="nickname">
          {{userInfo.nickName}}
        </text>
        <text class="username">
          用户名: {{userInfo.username}}
        </text>
      </view>
      <view v-else class="header-info" >
        请登录
      </view>
    </view>
    <text class="iconfont icon-right"></text>
  </view>
</view>
```


启动应用初始化登录状态

1. 重新打开应用时：

- 在 App.vue 中获取本地保存的 accessToken 和 userInfo 信息，如果有则认为是登录状态，没有则为未登录状态。
- 得到用户信息后需要同步改变 vuex 的状态，使所有页面都能共享登陆状态与用户信息。

```
// App.vue
export default {
  onLaunch: function() {
    console.log('App Launch')

    // 初始化登录状态
    this.initLogin()
  },

  methods: {
    // 初始化登录状态
    initLogin() {
      const userInfo = uni.getStorageSync('userInfo')
      const accessToken = uni.getStorageSync('mxgEducationToken')
      if(userInfo && accessToken) {
        this.$store.commit('setState', {userInfo, accessToken})
      }
    }
  },
}
```

2. 在 mxin.js 中的 navTo 方法里，增加判断未登录跳转到登录页

```
navTo(url, options={}){
  if(!url){
    return;
  }
  if(options.login && !this.$store.getters.hasLogin){
    url = '/pages/auth/login';
  }
  uni.navigateTo({
    url
  })
},
```

3. 将 common/js/util.js 中 isLogin 方法改为：有token返回true表示已登录

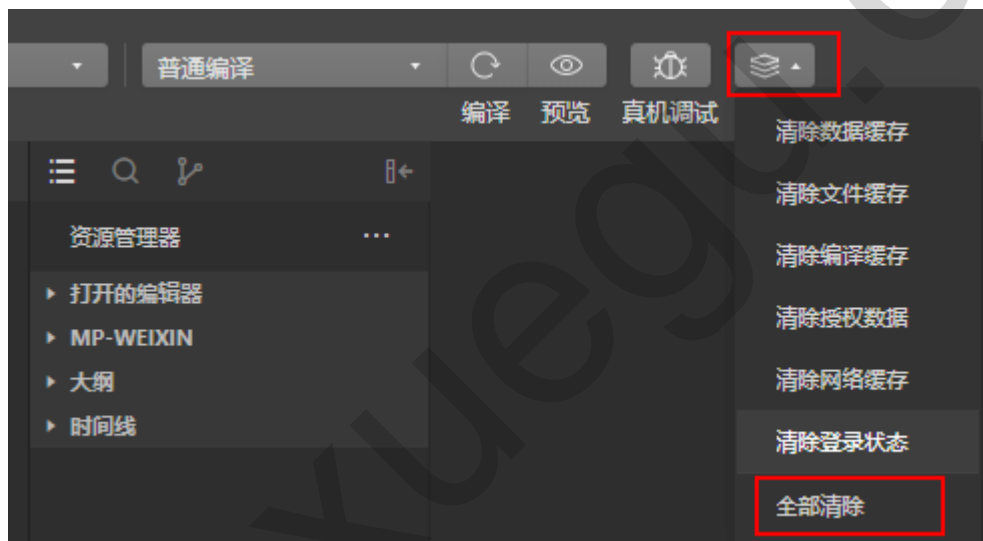
```
export const isLogin = (options={}) => {  
  const token = uni.getStorageSync('mxgEducationToken');  
  if(token){ // 改为有token返回true  
    return true;  
  }  
}
```

微信与QQ第三方授权登录

第三方登录参考: <https://uniapp.dcloud.io/api/plugins/login>

- 调用 `uni.login` 实现第三方授权登录
- 授权登录后，通过第三方响应的授权信息，再进行请求后台进行应用内认证授权，获取应用内认证信息

有登录过，微信小程序没有提示登录授权转圈圈图标，可清除缓存再试



1. 在 `toProviderLogin` 方法中进行调用 `uni.login` 实现第三方授权登录

```
// 微信、QQ提供商登录  
toProviderLogin(provider) {  
  if(!this.agreement){  
    this.$util.msg('请阅读并同意用户服务及隐私协议');  
    return;  
  }  
  uni.showLoading({mask: true})  
  
  uni.login({  
    provider,  
    // #ifdef MP-ALIPAY  
    scopes: 'auth_user', //支付宝小程序需设置授权类型  
    // #endif  
    success: (res) => {  
      console.log('login success:', res)  
      // #ifdef APP-PLUS  
  
      // APP端获取授权信息作为后台登录接口请求参数
```

```
const data = {userInfo: res.authResult}  
// #endif  
// #ifdef MP-WEIXIN  
// 微信小程序获取 code 后台登录接口请求参数  
const data = {code: res.code}  
// #endif  
// 请求后台进行应用内认证  
this.getServiceUserInfo(data)  
},  
fail: (err) => {  
  console.log('login fail:', err);  
  uni.hideLoading()  
}  
})  
},
```

2. EasyMock 创建 第三方授权登录应用接口

- 描述：第三方授权登录应用接口
- URL: `/auth/login/provider`
- 请求方式: `POST`
- mock.js 语法：注意可能是第1次授权登录，就需要绑定手机，所以下面手机号是空的

```
{  
  "code": 20000,  
  "message": "成功",  
  "data": {  
    "access_token": "@word(30)",  
    "token_type": "bearer",  
    "refresh_token": "fbdgbscpakavfpjurbeqegmrrrhgv",  
    "expires_in": "@natural", // access_token 有效时长  
    "scope": "all", // 权限范围  
    "userInfo": {  
      "uid": "9",  
      "username": "@name",  
      "mobile": null, // /1\d{10}/  
      "sex|1": [0, 1], // 0男, 1女  
      "nickName": "@cname",  
      "imageUrl": "https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif"  
    },  
    "jti": "@word(20)" // access_token 唯一标签  
  }  
}
```

3. `/api/auth.js` 文件封装调用数据接口方法

```
// 第三方授权登录应用接口
loginByProvider(data) {
  return request({
    url: '/auth/login/provider',
    method: 'POST',
    data
  })
},
```

4. 授权登录后，通过第三方响应的授权信息，再进行请求后台进行应用内认证授权，获取应用认证信息token等

```
/*
APP在微信 QQ授权登录后，
通过授权信息再调用后台接口应用内认证授权，获取应用认证信息token等
*/
async getServiceUserInfo (reqData){
  // 1. 调用后台服务接口应用内登录，获取用户信息和token
  const {code, message, data} = await api.loginByProvider( reqData )
  uni.hideLoading()
  if(code !== 20000) {
    this.$util.msg(message)
    return
  }
  // 2. 判断是否绑定了手机号
  if(data.userInfo.mobile && data.access_token) {
    // 2.1 已绑定：更新 store 中的登录状态为已登录
    this.loginSuccessCallback(data)
  }else {
    this.$util.msg('授权成功，请绑定手机号')
    // 2.2 未绑定：则跳转手机号绑定页 /pages/auth/bindMobile
    this.navTo('/pages/auth/bind-mobile?data=' + JSON.stringify(data))
  }
},
```

绑定手机号页面功能



绑定手机

手机号： 请输入手机号码

验证码： 请输入验证码

获取验证码

提交

当第三方授权登录后，当请求后台获取应用内认证信息**没有手机号**，则跳转绑定手机号页面。

绑定手机号页面后续修改手机号时也需要使用

创建配置绑定手机号页面

1. 创建配置绑定手机号页面 `pages/auth/bind-mobile.vue`

```
{
  "path" : "pages/auth/bind-mobile",
  "style" : {
    "navigationBarTitleText": "绑定手机"
  }
}
```

页面效果实现

1. 编写页面模板代码

```
<template>
  <view class="content">
    <view class="input-group">
      <view class="center">
        <text class="title">手机号: </text>
        <view class="row">
          <input v-model="mobile" class="mxg-input" type="number"
            focus maxlength="11" placeholder="请输入手机号码"></input>
        </view>
      </view>
    </view>
  </view>
```

```
<view class="center">
  <text class="title">验证码: </text>
  <view class="row">
    <input v-model="code" class="mxg-input"
      type="number" maxlength="6" placeholder="请输入验证码" ></input>
    <mxg-code :mobile="mobile" templateCode="SMS_198888"></mxg-code>
  </view>
</view>
</view>
<button type="primary" class="primary" @click="submitHandler" >提交</button>
</view>
</template>
```

2. 声明属性和方法

```
<script>
  export default {
    data() {
      return {
        mobile: null, // 手机号
        code: null, // 验证码
        data: null, // 其他页面传递数据
      }
    },

    onLoad(option) {
      this.data = option.data && JSON.parse(option.data)
    },

    methods: {
      // 提交修改
      async submitHandler() {
      },
    }
  }
</script>
```

3. 编写样式

```
<style lang="scss">
  page {
    background-color: $mxg-color-grey;
  }
  .content {
    margin: 80rpx 20rpx;
  }
  .input-group {
    padding: 0 25rpx;
    background-color: #FFFFFF;
    border-radius: 10rpx;
  }
  .title {
```

```
width: 190rpx;
font-size: 35rpx;
height: 100rpx;
line-height: 100rpx;
}
.row {
width: 100%;
}
.mxg-input {
flex: 1;
height: 60rpx;
line-height: 60rpx;
font-size: 35rpx;
}
}

button[type=primary] {
margin-top: 80rpx;
background-color: #345dc2 !important;
}
</style>
```

实现绑定手机号逻辑

1. EasyMock 创建 更新绑定手机号接口

- 描述: 更新绑定手机号接口
- URL: `/auth/user/mobile`
- 请求方式: `PUT`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "操作成功",
  "data": {
    "access_token": "@word(30)",
    "token_type": "bearer",
    "refresh_token": "fbdgsbscpakavfpjurbeqegmrhghv",
    "expires_in": "@natural", // access_token 有效时长
    "scope": "all", // 权限范围
    "userInfo": {
      "uid": "9",
      "username": "@name",
      "mobile": "/1\d{10}/", //
      "sex|1": [0, 1], // 0男, 1女
      "nickName": "@cname",
      "imageUrl": "https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif"
    },
    "jti": "@word(20)" // access_token 唯一标签
  }
}
```

```
}  
}
```

2. /api/auth.js 文件封装调用数据接口方法

```
// 更新绑定用户手机号  
updateUserMobile(data) {  
  return request({  
    url: '/auth/user/mobile',  
    method: 'PUT',  
    data  
  })  
},
```

3. 实现绑定逻辑

```
<script>  
import api from '@api/auth.js'  
export default {  
  methods: {  
    // 提交修改  
    async submitHandler() {  
      const {mobile, code} = this  
      // 校验手机号  
      if (!this.$util.checkStr(mobile, 'mobile')) {  
        this.$util.msg('手机号码格式不正确')  
        return  
      }  
  
      // 校验手验证码  
      if (!this.$util.checkStr(code, 'code')) {  
        this.$util.msg('验证码错误')  
        return  
      }  
  
      try{  
        uni.showLoading({mask: true})  
        // 提交请求  
        const data = {mobile, code, ...this.data}  
        const res = await api.updateUserMobile(data)  
        uni.hideLoading()  
        console.log('提交', res)  
        this.$util.msg(res.message)  
        if(res.code !== 20000) return  
        // 成功，更新状态值  
        this.successCallback(res.data)  
      }catch(e){  
        uni.hideLoading()  
        this.$util.msg('操作失败')  
        console.log('操作失败', e)  
      }  
    }  
  }  
}
```



```
    }  
  
    },  
  
    successCallBack(data) {  
      this.$util.msg('提交成功')  
      this.$store.commit('setToken', data)  
      setTimeout(()=>{  
        // 进入我的页面  
        this.navBack(2)  
        /* uni.switchTab({url: '/pages/my/my'}) */  
      }, 300)  
    },  
  
  },  
  
}  
</script>
```

个人资料页面实现



创建和配置个人资料页面

1. 创建个人资料页面文件 `pages/my/user`
2. `pages.json`配置 登录页面

```
{
  "path" : "pages/my/user",
  "style" : {
    "navigationBarTitleText": "个人资料"
  }
},
```

资料列表配置文件

个人资料页面内容引用 `mxg-list` 列表子组件，此组件最核心的是根据列表配置文件来进行渲染效果的。

对应展示数据来源于 Vuex 状态属性 `userInfo`。

1. 创建配置文件 `/config/user-list-bar.js`，

- 导入 store 对象获取当前用户信息，渲染行右侧内容

```
// 导入 store 对象获取当前用户信息
import store from '@/store'

const rightIcon = 'iconfont icon-right' // 右侧图标

export default () => {
  // 放到方法体，不然可能有时获取不到最新用户信息
  const userInfo = store.state.userInfo
  return [ // 二维数组导航
    [
      {
        title: '头像',
        event: 'chooseImg', // 点击触发的事件名
        src: userInfo.imageUrl || '/static/logo.png', // 右侧图片
        rightIcon, // 右侧图标
      },
      {
        title: '用户名',
        event: 'editUsername',
        text: userInfo.username,
        rightIcon,
      },
      {
        title: '手机号',
        event: 'editMobile',
        text: userInfo.mobile,
        rightIcon,
      }
    ],
    [
      {
```

```
        title: '昵称',
        event: 'editName',
        text: userInfo.nickName,
        rightIcon,
      },
      {
        title: '性别',
        event: 'chooseSex',
        text: userInfo.sex ? '女': '男',
        rightIcon,
      }
    ],
  ],
}
```

个人资料页面效果

1. 在 user.vue 导入配置文件数据，并传递给 mxg-list 子组件

```
<template>
  <view>
    <mxg-list :list="list"
      @chooseImg="chooseImg"
      @editUsername="editUsername"
      @editMobile="editMobile"
      @editName="editName"
      @chooseSex="chooseSex"
    />
    <button class="logout-btn" @click="logout">退出登录</button>
    <!-- 修改昵称时显示 -->
    <view v-if="isUpdate" class="bottom row"
      catchtouchmove="true" @touchmove.stop.prevent="()=>{}">
      <input v-model="content" class="mxg-input" :maxlength="20" focus type="text">
      <text class="update-btn" @click="submitUpdate">修改</text>
    </view>
  </view>
</template>
```

2. 页面样式

```
<style lang="scss">
  page {
    background-color: $mxg-color-grey;
  }

  .logout-btn {
```

```
margin-top: 25rpx;
color: $mxg-text-color-black;
background-color: #FFFFFF !important;
border-radius: 0;
&:after {
  border: 0;
}
}
.button-hover{
  background-color: $mxg-color-grey !important;
}

.bottom {
  z-index: 90;
  position: fixed;
  top: var(--window-top); /*h5端要空出导航高*/
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #FFFFFF;
}
.mxg-input {
  z-index: 100;
  width: 650rpx;
  height: 90rpx;
  font-size: 35rpx;
  padding: 0 20rpx;
  background-color: #FFFFFF;
  border: $mxg-underline;
  margin: 0 10rpx;
}
.update-btn {
  z-index: 100;
  width: 100rpx;
  height: 90rpx;
  line-height: 90rpx;
  text-align: center;
  color: $mxg-text-color-blue;
}
</style>
```

3. 逻辑实现

```
<script>
import list from '@config/user-list-bar.js'

export default {
  data () {
    return {
      list: list(), // 列表数据，不要少括号
      isUpdate: false, // 显示隐藏修改窗口
    }
  }
}
```

```
        content: '', // 修改内容
    },
    methods: {
        // 上传头像
        chooseImg(obj) {
        },

        // 选择性别
        chooseSex(obj) {
        },

        // 修改昵称
        editName(obj) {
        },
        // 更新输入框修改按钮
        submitUpdate(){

        },

        // 修改用户名
        editUsername() {
        },

        // 修改手机号
        editMobile() {
        },

        // 退出登录
        logout() {
        },
    }
}
</script>
```

更换头像逻辑实现

实现步骤：

- 使用相机拍照，或者从手机相册选择图片，使用 `uni.chooseImage(OBJECT)`。
参考：<https://uniapp.dcloud.io/api/media/image>
- 上传到云端，使用 `uni.uploadFile(OBJECT)`
参考：<https://uniapp.dcloud.io/api/request/network-file>
- 后台更新用户数据



1. EasyMock 创建 修改用户信息 接口

- 描述：修改用户信息
- URL: `/system/user`
- 请求方式: `PUT`
- mock.js 语法

```
{  
  "code": 20000,  
  'message': "修改成功"  
}
```

2. EasyMock 创建上传图片接口

- 描述：上传图片
- URL: `/article/file/upload`
- 请求方式: `PUT`

- mock.js 语法：注意图片地址可能无法访问，自行更换

```
{
  "code": 20000,
  "message": "成功",
  // 图片地址可能无法访问，自动更换
  "data": "https://mengxuegu.oss-cn-shenzhen.aliyuncs.com/article/20210123/3984a904f689443490d3c141baecb1cb.JPG"
}
```

3. `/api/system.js` 文件封装更新用户数据接口方法

```
// 修改用户信息
updateUserInfo(data) {
  return request({
    url: '/system/user',
    method: 'PUT',
    data
  })
},
```

4. 拷贝上传请求文件：`03-配套资料/common/js/upload.js` 到项目 `/common/js/upload.js`

5. 创建 `/api/common.js` 文件封装上传图片接口方法

注意导入 `upload.js` 来发送上传图片请求

```
// 注意导入 upload.js
import upload from '@common/js/upload.js'

export default {
  // 上传图片
  uploadImg(filePath) {
    return upload({
      url: '/article/file/upload',
      filePath
    })
  },
}
```

6. 在user.vue的chooseImg方法实现逻辑:

```
import api from '@api/system.js'
// 上传图片api
import commonApi from '@api/common.js'

// 用户状态
```

```
import { mapState } from 'vuex'

export default {

  computed: {
    ...mapState(['userInfo']), // 登录用户信息
  },

  methods: {
    // 更换头像
    chooseImg(obj) {
      // 1. 选择图片
      uni.chooseImage({
        count: 1, // 单选
        sizeType: ['compressed'], // 压缩
        success: async (chooseImageRes) => { // 注意要使用箭头函数
          const filePath = chooseImageRes.tempFilePaths[0];
          // 2. 上传图片(图片不能太大, 不然会失败)
          uni.showLoading({title: '更换头像中', mask: true})
          const {code, message, data} = await commonApi.uploadImg(filePath)
          if(code === 20000) {
            // 3. 更新数据
            this.userInfo.imageUrl = data
            // 4. 提交用户新头像url到后台
            this.updateUserInfo()
          } else {
            uni.hideLoading()
            this.$util.msg('头像更换失败, 请重试')
          }
        }
      })
    },

    // 调用服务接口更新数据和状态
    async updateUserInfo() {
      await api.updateUserInfo(this.userInfo)
      // 更新状态
      this.$store.commit('setToken', {userInfo: this.userInfo})
      // 重新渲染
      this.list = list()
      uni.hideLoading()
    },
  }
}
```

修改用户名和性别



1. 用户名不允许被修改，直接在editUsername方法提示

```
// 修改用户名
editUsername() {
  this.$util.msg('用户名不允许修改')
},
```

2. 修改性别，采用从底部向上弹出操作菜单 uni.showActionSheet(OBJECT)

```
const sex = ['男', '女']
export default {

  methods: {
    // 选择性别
    chooseSex(obj) {
      // 从底部向上弹出操作菜单
      uni.showActionSheet({
        itemList: sex,
        success: async (res) => { //注意使用箭头函数,
          // 相同不更新

```

```
if(res.tapIndex === this.userInfo.sex) return  
// 更新数据  
this.userInfo.sex = res.tapIndex  
this.updateUserInfo()  
}  
})  
},  
}
```

修改用户昵称



1. 点击昵称打开修改昵称窗口

```
// 监听页面导航后退<按钮,返回true阻止默认动作,不返回或返回其它值,均会执行默认的返回行为  
onBackPressed() {  
  if(this.isUpdate) {  
    this.isUpdate = false  
    return true;  
  }  
},  
methods: {  
  editName(obj) {  
    this.isUpdate = true  
    // 赋值内容到文本框  
    this.content = obj.text  
  },  
}
```

2. 提交修改昵称

```
//提交修改昵称
async submitUpdate() {
  const content = this.content.trim()
  if(!content) {
    this.$util.msg('不能为空，请重新填写')
    return
  }

  // 发送请求修改
  uni.showLoading({mask: true})
  // 调用服务接口更新数据
  this.userInfo.nickName = content
  this.updateUserInfo()
  // 关闭窗口
  this.isUpdate = false
},
```

修改手机号

修改手机号直接复用绑定手机号页面 bind-mobile.vue

< 修改手机号

手机号: 请输入手机号码

验证码: 请输入验证码

获取验证码

提交

1. 在 user.vue 点击手机号时，带上标题和当前用户信息跳转到 bind-mobile.vue 页面

```
// 修改手机号
editMobile() {
  const data = JSON.stringify({userInfo: this.userInfo})
  uni.navigateTo({
    url: '/pages/auth/bind-mobile?title=修改手机号&data='+data,
    animationType: 'slide-in-bottom' ,// 进入动画效果
  })
},
```

2. 绑定手机号页面 bind-mobile.vue 接收标题参数，设置到导航标题处，其他功能直接利用原有的，因为当前我们传递的参数不一样

```
onLoad(option) {
  this.data = option.data && JSON.parse(option.data)

  if(option.title) {
    // 导航标题
    uni.setNavigationBarTitle({
      title: option.title
    })
  }
},
```

退出登录功能

退出登录功能：调用后台退出接口，再触发 Vuex 中的 logout

1. EasyMock 创建退出登录接口

- 描述：退出登录
- URL: `/auth/logout`
- 请求方式: `GET`
- mock.js 语法:

```
{
  "code": 20000,
  "message": "退出成功"
}
```

2. `/api/auth.js` 文件封装接口方法

```
// 退出系统
logout(accessToken) {
  return request({
    url: '/auth/logout',
    method: 'GET',
    data: {accessToken}
  })
},
```

3. 点击个人资料页面的 退出登录 按钮，触发 logout 方法

```
import authApi from '@api/auth.js'

// 退出登录
logout() {
  uni.showModal({
    title: '确定退出登录?',
    content: '退出后不会删除任何历史数据',
    success: async (res) => {
      if(res.confirm) {
        const {code, message} = await authApi.logout(this.$store.state.accessToken)
        if(code === 20000) {
          this.$util.msg('成功退出登录')
          this.$store.commit('logout')
          setTimeout(()=>{
            this.navBack()
          }, 300)
        }else {
          this.$util.msg(message)
        }
      }
    }
  })
},
```

我的学习页面

我的学习页面：用于显示已经购买成功的课程，和展示每门课程的学习进度，如下图



我的学习

全深江革再青规可器总何基期民会
里交。

已学59%



空火又发边参把观原划六部济周员
争始。

已学93%



候马习今铁办联系人许支元。

已学61%



已非价长关指治见造明任形就。

已学59%



东所她表党加立解家族场去温术
外。

已学53%



1. 创建和配置我的学习页面 /pages/my/study.vue

```
{
  "path": "pages/my/study",
  "style": {
    "navigationBarTitleText": "我的学习",
    "backgroundColor": "#FFFFFF"
  }
}
```

2. 编写页面模板代码

```
<template>
  <view>
    <no-data v-if="true"></no-data>
    <view v-else
      class="course-item center">
      <view class="item-left column">
        <text class="title text-ellipsis">Uniapp在线教育视频播放项目实战教程</text>
      <view>
```

```
<text class="per">已学20%</text>
<progress percent="20" stroke-width="2" activeColor="#345DC2" />
</view>
</view>
<view class="right">
  <image class="course-img" src="/static/images/banner1.jpg" lazy-load />
</view>
</view>
</view>
</template>
```

3. 编写页面样式

```
<style lang="scss">
  .course-item {
    width: 100%;
    padding: 30rpx 0;
    .item-left {
      justify-content: space-between;
      width: 435rpx;
      height: 150rpx;
      margin-right: 20rpx;
      .title {
        line-height: 35rpx;
        font-size: 28rpx;
        font-weight: bold;
      }
      .per {
        font-size: 28rpx;
        color: $mxg-text-color-grey;
        line-height: 50rpx;
      }
    }
    .right {
      image {
        width: 245rpx;
        height: 150rpx;
        border-radius: 10rpx;
      }
    }
  }
</style>
```

4. EasyMock 创建查询我的学习列表接口

- 描述：查询我的学习列表接口
- URL： `/course/course/study/list`
- 请求方式： `GET`
- mock.js 语法：

```
{
  "code": 20000,
  "message": "查询成功",
  "data|8-20": [{ //产生8-20条
    "id|+1": 10, //初始值10开始, 每条+1
    "title": "@csentence", // 标题
    // 主图
    "mainImage|+1": ['/static/images/banner2.jpg', '/static/images/banner1.jpg',
'/static/images/banner3.jpg',
'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg'],
    "percent": "@integer(0, 100)"
  }]
}
```

5. /api/course.js 文件封装更新用户数据接口方法

```
// 查询我的学习列表
getCourseStudyList() {
  return request({
    url: '/course/course/study/list',
    method: 'GET'
  })
},
```

6. study.vue 页面调用接口获取数据

```
<script>
import api from '@api/course.js'

export default {
  data() {
    return {
      list: []
    }
  },
  onLoad() {
    this.loadData()
  },
  methods: {
    async loadData() {
      uni.showLoading({title: '加载中', mask: true})
      const {code, message, data} = await api.getCourseStudyList()
      uni.hideLoading()
      if(code === 20000) {
        this.list = data
      }else {
```



```
        this.$util.msg(message)
    }
  }
}
</script>
```

7. 页面模板渲染数据

```
<template>
  <view>
    <no-data v-if="!list || list.length<1"></no-data>
    <view v-else v-for="(item, index) in list" :key="index"
      @click="navTo(`/pages/course/course-play?id=${item.id}`)"
      class="course-item center">
      <view class="item-left column">
        <text class="title text-ellipsis">{{item.title}}</text>
        <view>
          <text class="per">已学{{item.percent}}%</text>
          <progress :percent="item.percent" stroke-width="2"
            activeColor="#345DC2"></progress>
        </view>
      </view>
      <view class="right">
        <image class="course-img" :src="item.mainImage" lazy-load></image>
      </view>
    </view>
  </view>
</template>
```

完善请求头带上令牌和路由拦截

请求头带上令牌

1. 当登录成功后，后台会响应访问令牌 `accessToken`，如果当前有访问令牌，则在请求头带上访问令牌，这样针后台会认证访问令牌是否有效，如果令牌有效，才可以访问需要登录后的接口，不然就访问失败。

找到 `request.js` 文件，添加以下三步代码：

```
request.js    upload.js    user.vue

13 // 1, 导入store
14 import store from '@store'
15 const request = (options = {}) => {
16
17     // 2. 判断请求头带上访问令牌
18     const accessToken = store.state.accessToken
19     if(accessToken) {
20         // 后台使用 oauth2 认证要求
21         options.header = {'Authorization': `Bearer ${accessToken}`}
22     }
23
24     // resolve 正常响应, reject异常响应
25     return new Promise((resolve, reject) => {
26         uni.request({
27             url: BASE_URL + options.url,
28             method: options.method || 'GET',
29             data: options.data || {},
30             timeout: 8000, // 8秒超时时间, 单位ms
31             header: options.header || {}, // 3. 添加到请求头
32             success: (res) => {
```

```
// 1, 导入store
import store from '@store'
const request = (options = {}) => {

    // 2. 判断请求头带上访问令牌
    const accessToken = store.state.accessToken
    if(accessToken) {
        // 后台使用 oauth2 认证要求
        options.header = {'Authorization': `Bearer ${accessToken}`}
    }

    // resolve 正常响应, reject异常响应
    return new Promise((resolve, reject) => {
        uni.request({
            url: BASE_URL + options.url,
            method: options.method || 'GET',
            data: options.data || {},
            timeout: 8000, // 8秒超时时间, 单位ms
            header: options.header || {}, // 3. 添加到请求头
            success: (res) => {
```

H5 端路由拦截

1. 当前 h5 端没有登录，也可以访问到要求登录的页面，我们在 request.js 针对页面请求接口进行判断 url 是否包含 /api/：没有包含 /api/ 说明要先登录，而没有登录则进入登录页。

request.js 添加如下代码：**短信验证码也要放行**

```
// 白名单，短信验证码也要放行
const whileList = ['/system/sms/code', '/auth/login', '/auth/login/provider',
'/auth/user/mobile']

return new Promise((resolve, reject) => {
  // 没有包含/api/说明要先登录，而没有登录则进入登录页
  if(whileList.indexOf(options.url)===-1 &&
    options.url.indexOf('/api/') < 0 && !accessToken) {
    return uni.navigateTo({
      url: '/pages/auth/login'
    })
  }
})
}
```

2. 个人资料页初始化列表时，没有调用接口，则拦截不到，在 user.vue 解决方式如下：

```
onLoad() {
  this.$util.isLogin() //会判断没有登录进入登录页
},
```

图片上传拦截权限判断

1. 图片上传必须是登录状态才可以上传图片，在upload.js 添加如下代码

```
return new Promise((resolve, reject) => {
  //判断是否登录，登录后才可以上传图片
  const accessToken = store.state.accessToken
  if(!accessToken) {
    uni.showToast({title: '请先登录', icon: 'none'})
    setTimeout(() => {
      uni.navigateTo({
        url: '/pages/auth/login'
      })
    }, 500)
    return
  }
  uni.uploadFile({
    url: BASE_URL + options.url, // 服务器 url
    filePath: options.filePath, // 要上传文件资源的路径。
    name: options.name || 'file', // File 对象对应 key
    formData: options.data || {}, //额外的 form data
    timeout: 8000, // 8秒超时时间，单位ms
    header: {'Authorization': `Bearer ${accessToken}` },
  }, (res) => {
    resolve(res.data)
  })
})
```

```
import store from '@store'
// 上传图片
const upload = (options = {}) => {

  // resolve 正常响应, reject异常响应
  return new Promise((resolve, reject) => {
    // 上传前, 判断是否登录
    const accessToken = store.state.accessToken
    if(!accessToken) {
      uni.showToast({title: '请先登录', icon: 'none'})
      return setTimeout(()=>{
        uni.navigateTo({
          url: '/pages/auth/login'
        })
      }, 500)
    }

    uni.uploadFile({
      url: BASE_URL + options.url, // 服务器 url
      filePath: options.filePath, // 要上传文件资源的路径。
      name: options.name || 'file', // File 对象对应 key
      formData: options.data || {}, //额外的 form data
      timeout: 8000, // 8秒超时时间, 单位ms
      header: {'Authorization': `Bearer ${accessToken}` },
    })
```