16-5 首页交互——文件列表切换目录和文件列表排序方式

> 首页的目录层级切换和文件排序简直是逆天复杂。
>
> 这部分大致意思是数组的 push 和 pop，进入一层目录就是 push，返回就是 pop。
>
> 需要改造首页导航，进到子目录，左边会出现返回箭头图标，点击会返回上一层，同时路由出栈，定义一个响应式数组 dirs，要用 let 定义，因为它会频繁变化，用于记录路由数组。

```
// 路由数组
let dirs = ref([])
```

template 部分代码

```
<template>
    <view>
        <!-- 选中个数为0 -->
        <uni-nav-bar v-if="checkedList.length === 0">
            <template #left>
                <!-- 插槽发挥逆天作用，进入子目录，左边将变成返回箭头，导航栏的标题变成子目录名 -->
                <view style="width: 60rpx;height: 60rpx;"
                    class="flex align-center justify-center bg-light rounded-circle ml-3" @tap="backUp" v-if="current">
                     <text class="iconfont icon-fanhui"></text>
                </view>
                <text class="font-md ml-3 text-light">{{ current ? current.name : '首页'}}</text>
            </template>
            <template #right>
                <view style="width: 60rpx;height: 60rpx;"
                    class="flex align-center justify-center bg-light rounded-circle mr-3" @tap="openAddPopup">
                    <text class="iconfont icon-zengjia"></text>
                </view>
                <view style="width: 60rpx;height: 60rpx;"
                    class="flex align-center justify-center bg-light rounded-circle mr-3" @tap="openSortPopup">
                    <text class="iconfont icon-gengduo"></text>
                </view>
            </template>
        </uni-nav-bar>
```

script 部分代码，sortOptions 是数组记录排序方式的，其中的 key 可以作为查询参数传到后端，去按照这个排序。

```
// 文件排序相关
const sortIndex = ref(0)
const sortOptions = ref([{
        name: '按名称排序',
        key: 'name'
    },
    {
        name: '按时间排序',
        key: 'created_time'
    }
])

const sortPopup = ref(null)
const openSortPopup = () => {
    sortPopup.value.open()
}
const changeSort = (index) => {
    sortIndex.value = index
    sortPopup.value.close()
}
```

每次页面加载的时候，从本地存储读取 dirs，如果不清空，会从上次离开的地方继续。

```
// 在页面 onShow 生命周期调用
onShow(() => {
    // 从本地存储读取路由数组
    let dirsLocal = uni.getStorageSync('dirs')
    if (dirsLocal) {
        // 反序列化后赋值给上面的路由数组变量
        dirs = JSON.parse(dirsLocal)
    }
    getList()
})
```

两个计算属性，实时根据当前 dirs 数组的变化，file_id 计算属性取得应该传到后端的 file_id 参数（就是当前目录），current 计算属性则用来切换导航栏样式。

useIndex.js

```
177
178     // 路由数组
179     let dirs = ref([])
180
181     // 计算属性，取得应该传到后端的file_id参数（就是当前目录）
182     const file_id = computed(() => {
183         const len = dirs.value.length
184         if (len === 0) {
185             return 0;
186         }
187         return dirs.value[len - 1].id;
188     })
189
190     // 计算属性，用来切换导航栏样式
191     const current = computed(() => {
192         const len = dirs.value.length
193         if (len === 0) {
194             return false
195         }
196         return dirs.value[len - 1]
197     })
```

每次请求 API 接口的时候，把最新的 file_id（要显示哪个目录的文件） 和选取的 orderby 排序方式带上

useIndex.js

```
6   import $H from '/common/request.js'
7
8   export function useIndex() {
9
10      const list = ref([])
11
12      // 请求接口数据
13      const getList = () => {
14          console.log('file_id:' + file_id.value)
15          // 取得排序方式的key，有 name、created_time 两种
16          let orderby = sortOptions.value[sortIndex.value].key
17          // 获取文件列表， file_id 为0 表示找云盘根目录的所有数据
18          $H.get(`/file?file_id=${file_id.value}&orderby=${orderby.value}`, {
19              token: true
20          }).then(res => {
21              list.value = formatList(res.rows)
22          })
23      }
24
```

切换排序的要增加根据最新选的排序方式去请求接口数据，修改之前的 changeSort 方法如下：

```
265     // 切换排序
266     const changeSort = (index) => {
267         sortIndex.value = index
268         getList()
269         sortPopup.value.close()
270     }
```

列表元素点击事件，**图片和视频点击前面已经做过了，实现了图片的预览和视频的播放。**

那么文件夹的点击怎么处理呢？**就是要把当前元素 push 到路由数组中去，然后用这个目录的 id，去请求它的层级里的数据，同时存到本地存储中。**

doEvent 新增 case 'dir' 分支，增加如下代码：

useIndex.js

```
144
145     // 列表项点击事件处理(图片预览、视频播放等)
146     const doEvent = (item) => {
147         switch (item.type) {
148             case 'image':
149                 //从 list 中过滤得到所有类型为 image 的文件
150                 let images = list.value.filter(item => {
151                     return item.type === 'image';
152                 });
153                 // 预览图片
154                 uni.previewImage({
155                     current: item.url,
156                     urls: images.map(item => item.url)
157                 });
158                 break;
159             case 'video':
160                 uni.navigateTo({
161                     url: '../video/video?url=' + item.url + '&title=' + item.name
162                 })
163                 break;
164             case 'dir':
165                 dirs.value.push({
166                     id: item.id,
167                     name: item.name
168                 });
169                 getList()
170                 uni.setStorage({
171                     key: 'dirs',
172                     data: JSON.stringify(dirs.value)
173                 })
174                 break;
175         }
176     }
177
```

顶部导航栏在子目录的时候，会有返回箭头，它的事件如下，路由数组 pop，再获取回到上一层的最新数据，存储到本地存储。新增 changeDir 方法，用来切换文件目录。

```
// 切换目录
const changeDir = () => {
    dirs.value.pop()
    getList()
    uni.setStorage({
        key: 'dirs',
        data: JSON.stringify(dirs.value)
    })
}
```

最后，在退出登录的时候，要把本地存储都清空，store/index.js

index.js

```
        token: null,
    },
    actions: {
        logout({ state }) {
            $H.post('/logout', {}, {
                token: true
            })
            state.user = null
            state.token = null
            uni.removeStorageSync('user')
            uni.removeStorageSync('token')
            uni.removeStorageSync('dirs')

            // 重启应用
            uni.reLaunch({
                url: '/pages/login/login'
            })
        },
```

```
commit 27334daa74823959c1ecda41c7c24af4b144fc84 (HEAD -> main)
Author: mqxu <moqi1977@gmail.com>
Date:   Mon Jul 10 13:25:06 2023 +0800

    改造成hooks & 首页交互—文件列表切换目录和文件列表排序方式
```