功能实现

1 表格拖拽调整列宽

dependencies

1. "view-design": "^4.7.0" （至少 4.0.0 版本）

实现步骤

1. 开启 `border` 属性并设置 `@on-column-width-resize` 事件：

```
<Table
  border
  :columns="tableColumns"
  @on-column-width-resize="saveTableState"
>
</Table>
```

2. 给需要该功能的列设置属性 `resizable` 为 true，并设置 width 属性：

```
export default {
    ...
    data() {
        return {
            tableColumns: [
            {
                title: '姓名',
                key: 'name',
                sortable: 'custom',
                resizable: true, // resizable 设置为 true
                width: 180 // 必须设置 width 属性
            },
            ... // 其他列属性
            ]
        }
    }
}
```

3. 创建函数 `saveTableState()`[1]。

2 自定义列顺序

dependencies

1. "sortablejs": "^1.15.0"
2. "view-design": "^4.7.0" （至少 4.0.0 版本）

实现步骤

1. 导入页面中导入 sortablejs：

```
import Sortable from 'sortablejs'
```

2. 设置列属性 `ref="mainTable"` 、`:key="tableUpdateKey"`

```
<Table
  ref="mainTable"
  :key="tableUpdateKey"
  :columns="tableColumns"
>
</Table>
```

3. 在 data 中设置 `tableUpdateKey`：

```
export default {
    ...
    data() {
        return {
            tableUpdateKey: 0, // 数据表表头调整后刷新
            ...
        }
    }
}
```

4. 创建函数 `columnDrop()`：

```
export default {
    ...
    methods: {
        columnDrop() {
            // 获取 <Tr> 标签节点
            const wrapperTr =
this.$refs.mainTable.$refs.header.firstChild.children[1].children[0]

            this.sortable = Sortable.create(wrapperTr, {
                animation: 180,
                delay: 0,
                onEnd: (evt) => {
                    const oldItem = this.tableColumns[evt.oldIndex]
                    this.tableColumns.splice(evt.oldIndex, 1)
                    this.tableColumns.splice(evt.newIndex, 0, oldItem)
```

```
                    // 刷新组件状态
                    this.tableUpdateKey += 1

                    // 保存组件状态
                    this.saveTableState()

                    // 重新挂载 columnDrop()
                    this.$nextTick(function() {
                      this.columnDrop()
                    })
                }
            })
        },
        ...
    }
}
```

5. 在 vue 钩子函数 `mounted()` 中初次挂载 `columnDrop()`

```
export default {
    ...
    mounted() {
        ...
        this.columnDrop()
    },
}
```

6. 创建函数 `saveTableState()`[2]。

注意事项

1. 如果出现拖拽无效的情况，可以通过重新挂载 `columnDrop()` 解决。

3 实现状态保存

dependencies

1. "vuex-persistedstate": "^4.1.0"
2. "vuex": "3.1.0"

实现步骤

1. 在 `../store/【模块名】` 路径下 `子模块名.js` 文件:

```
const getDefaultState = () => {
    return {
        columnAttribute: [],
    }
}

const state = getDefaultState()

const mutations = {
    SET_COLUMNATTRIBUTE(state, columnAttribute) {
        state.columnAttribute = columnAttribute
    },
    RESET_COLUMNATTRIBUTE(state) {
        Object.assign(state, getDefaultState())
    },
}

const actions = {
    SET_COLUMNATTRIBUTE(context, columnAttribute) {
        context.commit('SET_COLUMNATTRIBUTE', columnAttribute)
    },
    RESET_COLUMNATTRIBUTE({ commit }) {
        commit('RESET_COLUMNATTRIBUTE')
    }
}

export default ({
    namespaced: true,
    state,
    mutations,
    actions,
})
```

2. 在 `../store/index.js` 中引入 vuex-persistedstate:

```
...
// 导入模块
import 【模块名】 from './【模块名】/【子模块名】'
// 导入持久化插件
import 【模块名】 from 'vuex-persistedstate'

```

```
const dataState = createPersistedState({
  key: 'demo-loaldata',
  paths: ['【模块名】']
})

const store = new Vuex.Store({
  modules: {
    【模块名】,
    ...
  },
  plugins: [
    dataState // 导入插件
  ],
  getters
})

export default store
```

3. 创建函数 `saveTableState()` 与 `getTableState()`：

```
export default {
    ...
    methods: {
        saveTableState() {
            let columnAttribute = this.tableColumns.map((obj, index) => {
                let t = {}
                t.situation = index
                t.key = obj.key
                t.width = obj.width
                return t
            })
            this.$store.commit('【模块名】/SET_COLUMNATTRIBUTE', columnAttribute)
        },
        getTableState() {
            let columnAttribute = this.$store.state.staffList.columnAttribute
            // 判断用户是否进行了修改
            if (columnAttribute.length !== 0) {

                // 用户进行过修改该，读取修改后的配置
                let newTableColumns = []
                this.tableColumns.forEach((item) => {
                    const t = columnAttribute.find((i) => i.key === item.key)
                    item.width = t.width
                    newTableColumns[t.situation] = item
                })
                this.tableColumns = newTableColumns
            }
        }
        ...
    }
}
```

4. 在 vue 钩子函数 `mounted()` 中调用 `getTableState()`：

```
export default {
    ...
    mounted() {
        ...
        this.getTableState()
    },
}
```

5. 在特定事件函数中调用 `saveTableState()`[3]。

附录

参考资料

1. 1 表格拖拽调整列宽——IView 文档 发布于 【0000/00/00】;
2. 2 自定义列顺序——表格拖拽排序 发布于 2022/08/05;
3. 3 实现状态保存——更高效的vuex状态缓存方式-createPersistedState 发布于 2020/09/06日

---

1. 参考章节 3 实现状态保存。 ↩
2. 参考章节 3 实现状态保存。 ↩
3. 参考章节 1 表格拖拽调整列宽 与 2 自定义列顺序 ↩