

# 程序设计原则

---

设计模式非常多，每解决一个问题都会形成设计模式。随着系统的迭代，系统的设计模式也在迭代。

实际项目中，无法在第一时间将所有东西都用到位，更不能为了用模式而用模式——那么该怎么思考呢？

接下来我们讨论几个思考软件设计时的通用原则。

## Rule of Three

---

不要出现3份类似的程序。

## 密封性和单一职责

---

为什么Antd中的Select和Option是分开的？Tab和Panel是分开的？

```
const Option = Select.Option  
  
const TabPanel = Tab.Panel
```

为什么作业1中的UILayerView应该先实现TreeView？

延伸下：关注点分离原则

出自：Composition API proposal<https://github.com/vuejs/rfcs/blob/master/active-rfcs/0013-composition-api.md>

```

1 import { createLocalize } from "graphql-folders"
2 import FOLDER_CURRENT from "graphql-folders/folderCurrent.gql"
3 import FOLDER_PARENTS from "graphql-folders/folderParents.gql"
4 import FOLDER_PATH from "graphql-folders/folderPath.gql"
5 import FOLDER_PATH_PARENTS from "graphql-folders/folderPathParents.gql"
6 import FOLDER_SET_PARENTS from "graphql-folders/folderSetParents.gql"
7 import PROJECT_CREATE from "graphql-folders/projectCreate.gql"
8 import FOLDER_CREATE from "graphql-folders/folderCreate.gql"
9 const SHOW_HIDE = "show-hide-folders-folders"
10
11 export default {
12   data() {
13     return {
14       loading: false,
15       error: false,
16       edittingPath: false,
17       edittingPath: false,
18       folderCurrent: {},
19       folderFavorite: {},
20       showHidden: localStorage.getItem(SHOW_HIDE) === "true",
21       showHidden: false,
22       newFolderName: ""
23     }
24   },
25   async() {
26     folderCurrent() {
27       query: FOLDER_CURRENT,
28       fetchPolicy: "network-only",
29       loadingKey: "loading",
30       async: result() {
31         return result()
32       },
33       onEnd: () {
34         this.$refs.folderScrollTop = 0
35       }
36     }
37     folderFavorite: FOLDER_FAVORITE
38   },
39   computed: {
40     newFolderName() {
41       return this.$data.newFolderName
42     }
43   },
44   watch: {
45     showHidden: function() {
46       if (this.$data.showHidden === "true") {
47         localStorage.setItem(SHOW_HIDE, "true")
48       } else {
49         localStorage.removeItem(SHOW_HIDE)
50       }
51     }
52   },
53   beforeRouteLeave(to, from, next) {
54     if (this.$data.showHidden === "true") {
55       this.$data.showHidden = "true"
56     }
57     next()
58   },
59   methods: {
60     async openFolder(path) {
61       this.loading = true
62       this.error = null
63       this.loading = false
64       try {
65         await this.$api.fetch({
66           mutation: FOLDER_PATH,
67           variables: {
68             path
69           },
70           update: (store, { data: { folderPath } }) => {
71             store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPath } })
72           }
73         })
74       } catch (err) {
75         this.error = err
76       }
77       this.loading = false
78     },
79     async openParentFolder(folder) {
80       this.loading = true
81       this.error = null
82       this.loading = false
83       try {
84         await this.$api.fetch({
85           mutation: FOLDER_PATH_PARENTS,
86           update: (store, { data: { folderPathParents } }) => {
87             store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
88           }
89         })
90       } catch (err) {
91         this.error = err
92       }
93       this.loading = false
94     },
95     async toggleFavorite(id) {
96       await this.$api.fetch({
97         mutation: FOLDER_SET_PARENTS,
98         variables: {
99           path: this.$data.folderCurrent.path,
100           favorite: this.$data.folderCurrent.favorite
101         },
102         update: (store, { data: { folderSetParents } }) => {
103           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderSetParents } })
104         }
105       })
106       // See https://github.com/apollographql/apollo-client/issues/8873#issuecomment-42086879
107       data = {
108         folderFavorite: data.folderFavorite.toggle()
109       }
110       if (folderFavorite.favorite) {
111         data.folderFavorite.path = folderSetParents.path
112       } else {
113         data.folderFavorite.path = folderSetParents.path
114       }
115       if (index === -1) data.folderFavorite.toggleIndex(index)
116       store.writeQuery({ query: FOLDER_FAVORITE, data })
117     },
118     async toggleFavorite(id) {
119       return {
120         new: folderFavorite.data.new
121       }
122     },
123     async openFolder(id) {
124       this.loading = true, folderCurrent.path
125       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
126       this.loading = false
127       this.$api.fetch({
128         mutation: FOLDER_PATH,
129         variables: {
130           path: this.$data.folderCurrent.path,
131           favorite: this.$data.folderCurrent.favorite
132         },
133         update: (store, { data: { folderPath } }) => {
134           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPath } })
135         }
136       })
137     },
138     async openFolder(id) {
139       this.loading = true, folderCurrent.path
140       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
141       this.loading = false
142       this.$api.fetch({
143         mutation: FOLDER_PATH_PARENTS,
144         variables: {
145           path: this.$data.folderCurrent.path,
146           favorite: this.$data.folderCurrent.favorite
147         },
148         update: (store, { data: { folderPathParents } }) => {
149           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
150         }
151       })
152     },
153     async openFolder(id) {
154       this.loading = true, folderCurrent.path
155       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
156       this.loading = false
157       this.$api.fetch({
158         mutation: FOLDER_PATH_PARENTS,
159         variables: {
160           path: this.$data.folderCurrent.path,
161           favorite: this.$data.folderCurrent.favorite
162         },
163         update: (store, { data: { folderPathParents } }) => {
164           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
165         }
166       })
167     },
168     async openFolder(id) {
169       this.loading = true, folderCurrent.path
170       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
171       this.loading = false
172       this.$api.fetch({
173         mutation: FOLDER_PATH_PARENTS,
174         variables: {
175           path: this.$data.folderCurrent.path,
176           favorite: this.$data.folderCurrent.favorite
177         },
178         update: (store, { data: { folderPathParents } }) => {
179           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
180         }
181       })
182     },
183     async openFolder(id) {
184       this.loading = true, folderCurrent.path
185       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
186       this.loading = false
187       this.$api.fetch({
188         mutation: FOLDER_PATH_PARENTS,
189         variables: {
190           path: this.$data.folderCurrent.path,
191           favorite: this.$data.folderCurrent.favorite
192         },
193         update: (store, { data: { folderPathParents } }) => {
194           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
195         }
196       })
197     },
198     async openFolder(id) {
199       this.loading = true, folderCurrent.path
200       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
201       this.loading = false
202       this.$api.fetch({
203         mutation: FOLDER_PATH_PARENTS,
204         variables: {
205           path: this.$data.folderCurrent.path,
206           favorite: this.$data.folderCurrent.favorite
207         },
208         update: (store, { data: { folderPathParents } }) => {
209           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
210         }
211       })
212     },
213     async openFolder(id) {
214       this.loading = true, folderCurrent.path
215       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
216       this.loading = false
217       this.$api.fetch({
218         mutation: FOLDER_PATH_PARENTS,
219         variables: {
220           path: this.$data.folderCurrent.path,
221           favorite: this.$data.folderCurrent.favorite
222         },
223         update: (store, { data: { folderPathParents } }) => {
224           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
225         }
226       })
227     },
228     async openFolder(id) {
229       this.loading = true, folderCurrent.path
230       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
231       this.loading = false
232       this.$api.fetch({
233         mutation: FOLDER_PATH_PARENTS,
234         variables: {
235           path: this.$data.folderCurrent.path,
236           favorite: this.$data.folderCurrent.favorite
237         },
238         update: (store, { data: { folderPathParents } }) => {
239           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
240         }
241       })
242     },
243     async openFolder(id) {
244       this.loading = true, folderCurrent.path
245       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
246       this.loading = false
247       this.$api.fetch({
248         mutation: FOLDER_PATH_PARENTS,
249         variables: {
250           path: this.$data.folderCurrent.path,
251           favorite: this.$data.folderCurrent.favorite
252         },
253         update: (store, { data: { folderPathParents } }) => {
254           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
255         }
256       })
257     },
258     async openFolder(id) {
259       this.loading = true, folderCurrent.path
260       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
261       this.loading = false
262       this.$api.fetch({
263         mutation: FOLDER_PATH_PARENTS,
264         variables: {
265           path: this.$data.folderCurrent.path,
266           favorite: this.$data.folderCurrent.favorite
267         },
268         update: (store, { data: { folderPathParents } }) => {
269           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
270         }
271       })
272     },
273     async openFolder(id) {
274       this.loading = true, folderCurrent.path
275       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
276       this.loading = false
277       this.$api.fetch({
278         mutation: FOLDER_PATH_PARENTS,
279         variables: {
280           path: this.$data.folderCurrent.path,
281           favorite: this.$data.folderCurrent.favorite
282         },
283         update: (store, { data: { folderPathParents } }) => {
284           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
285         }
286       })
287     },
288     async openFolder(id) {
289       this.loading = true, folderCurrent.path
290       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
291       this.loading = false
292       this.$api.fetch({
293         mutation: FOLDER_PATH_PARENTS,
294         variables: {
295           path: this.$data.folderCurrent.path,
296           favorite: this.$data.folderCurrent.favorite
297         },
298         update: (store, { data: { folderPathParents } }) => {
299           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
300         }
301       })
302     },
303     async openFolder(id) {
304       this.loading = true, folderCurrent.path
305       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
306       this.loading = false
307       this.$api.fetch({
308         mutation: FOLDER_PATH_PARENTS,
309         variables: {
310           path: this.$data.folderCurrent.path,
311           favorite: this.$data.folderCurrent.favorite
312         },
313         update: (store, { data: { folderPathParents } }) => {
314           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
315         }
316       })
317     },
318     async openFolder(id) {
319       this.loading = true, folderCurrent.path
320       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
321       this.loading = false
322       this.$api.fetch({
323         mutation: FOLDER_PATH_PARENTS,
324         variables: {
325           path: this.$data.folderCurrent.path,
326           favorite: this.$data.folderCurrent.favorite
327         },
328         update: (store, { data: { folderPathParents } }) => {
329           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
330         }
331       })
332     },
333     async openFolder(id) {
334       this.loading = true, folderCurrent.path
335       store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderCurrent.path } })
336       this.loading = false
337       this.$api.fetch({
338         mutation: FOLDER_PATH_PARENTS,
339         variables: {
340           path: this.$data.folderCurrent.path,
341           favorite: this.$data.folderCurrent.favorite
342         },
343         update: (store, { data: { folderPathParents } }) => {
344           store.writeQuery({ query: FOLDER_CURRENT, data: { folderCurrent: folderPathParents } })
345         }
346       })
347     },
348     async openFolder(id) {
349       this.loading = true,
```

## Options API

```
import { createApp } from 'vue'
import './style.css'
import App from './App.vue'

createApp(App).mount('#app')
```

## Composition API

```
import { createApp } from 'vue'
import './style.css'
import App from './App.vue'

createApp(App).mount('#app')
```

总结下：

- 每件事情应该有独立的模块处理
- 每个独立的模块要把事情做好、做完整

32个棋 1个棋盘

## 单向依赖原则

在单向数据流中讨论非常信息了。组件不要发生双向依赖，如果发生双向依赖可以这样解耦：

- 消息（EventBus, Redux.....）
- 重新设计(skedo早先版本对比)

## SSOT原则

Single Source of Truth

数据的来源只有1个，真理只有一个。

关联的原则：最小知识原则。

```
const ProductList = (props) => {  
  // {a,b,c,d}  
  const [passProps, setProps] = useState(props)  
  
  return <X {...passProps}>  
    </X>  
}
```

举例1：Restful

举例2：减少组件间参数传递

反模式的设计：商品表单 -> 牌子子表单 -> 品牌列表

反思：组件从数据层面也是密封的(sealed)。例如一个订单列表组件应该自己可以完成所有数据的获取，即便为了提升性能数据作为一个整体被服务端返回。

## 最小交互原则

---

减少类型间的交互，减少类型之间的耦合。

Rule1：减少继承、多用组合。

- 工厂模式、Facade模式、Builder模式.....

Rule2：减少类型的成员多：

- 发消息通知
- 管道（组合）
- continuations

## 开闭原则

---

**对扩展开放，对修改封闭。**

提升程序的扩展性（比如插件、元数据、DSL等），减少对程序的修改。

# 领域设计原则

---

Rule1: 创建属于自己的领域方言, 让每个对象拥有贴近场景的具体含义, 做到专对象专用。

例如: HTML、JSX、@skedo/meta/Node, @skedo/creator/Project,

Rule2: 用DSL描述你的系统

DSL(json ,yml, builder) -> ActivityPage

Rule3: 为不同的目标设计Context

参见:Skedo的RenderContext

Rule4 : 让元数据可以被扩展能力 (插件、组件等) 使用

参见@skedo的外部组件设计