Thinking in APTSMAS

Ancient Poets Temporal and Spatial Mobility Analysis System

# Introduction

The aims of ***Thinking in APTSMAS*** are not only to describe the Working logic of APTSMAS, but also show you where of this system can be improved. For further development, this doc will be greatly helpful to you.

# Front End

I am afraid that the front end’s code is the largest and the most complex part among APTSMAS. The front end is developed by Vue 3 eco-system with TypeScript. The basic project of front end is initialized with Vue CLI v4.

## Project Structure

The Front End’s project structure is as following:

图形用户界面, 应用程序

中度可信度描述已自动生成

Figure The Front End Struct

The top directory contains the configure files of all needed tools such as EditorConfig, ESLint, Babel and more.

The public directory’s files will be directly put to the output directory, usually putted with static pictures.

电子设备的屏幕

描述已自动生成

Figure The Public Dir

The src directory contains all code:

图片包含 图形用户界面

描述已自动生成

Figure The src Dir

The components folder contains all components of each page and public page struct.

The composables folder contains all logic *hooks* and other functions related with each page and components.

The languages folder contains the vue-i18n logic code and translation of Chinese and English.

The router folder contains the logic code of Vue Router.

The store folder contains the logic code of VueX and init data.

The typings folder contains the type declare of some utils.

The views folder contains the actually router view components of each page.

App.vue is the basic page struct and the father of all components.

main.ts init the front end with vue.

图片包含 文本

描述已自动生成

Figure Components and Composables

The components and composables folder follow the guild by Vue official doc. Inside are organized by page view. I don’t really sure this structure is good or not, maybe reorganized by page directly (or packaged all related code into a page view package) will be better.

## Public Logic

Public Logic contains the page layout struct and public components.

### Page Layout Structure

The basic app layout (App.vue) is inited with Sticky Header and Sticky Footer with Flex layout. The content part wraps the vue router mount point.

The Naïve UI from ToSimple company is used for layout construct. The beautify scrollbar and back to top components are used for better user experience.

### Page Header Text

PageHeaderText.vue is used by all router view page component to support the custom page header and page description text.

### Dynamic Ico

The useDynamicIco.ts return a hook for dynamic change the page ico, which is used by page view components.

### Locale Change

The locale info is stored into frontend localStorage. The change will affect all trans text even page header of browser’s tab. Related files are: useLocaleChange.ts and useNuiLocaleChange.ts.

### Router Button

Router button is part of the layout sticky header, the useRouterBtn.ts return hook for dynamic change with front end url route change. I18n is supported in this hook.

### AMap

The AMap is loaded with the AMap official JS loader. The AMap/useAMap.ts packaging the JS loader into hook. The async loader will load AMap and Loca directly on window object. However it also return the promise of AMap object. So, you can directly use the return of the custom hook or use it by access window object.

The official @types of AMap has some bugs, so I put it under the typings folder and make some change to it.

文本

描述已自动生成

Figure typings of APTSMAS

## Vue Router

The router folder contains the Vue Router logic. Notice that all router component used dynamic import:

|  |
| --- |
| {  path: "/original-data",  name: "OriginalData",  component: () => import("../views/VOriginalData.vue"),  }, |

The unmarked route will be routed to the 404 page:

|  |
| --- |
| // the 404  {  path: "/:pathMatch(.\*)\*",  name: "404",  component: () => import("../views/404.vue"),  }, |

## VueX

The VueX use different modules to manage the stored data. In APTSMAS, the public store are used for store the public data (only read). This somehow is not quite elegant. If you have interests, remold the frontend into Flux data flow or other may worth a try.

The VueX have the TypeScript enhance, though the type declare interface should be written by yourself.

### Geo

The Geo module store two type of data. First is the object of GeoId to GeoName, another is the object of GeoName to GeoId. Because the special ID design, GeoId among country, province, city and location will be unique. Actually, the name of each geographic point is also unique.

Many backend API response only the ID of geography point rather than geography name, so it’s necessary to store the mapping relationship of GeoId to GeoName.

### Poet

The Poet module store two type of data, the object of PoetId to PoetDetail and the object of PoetName to PoetDetail.

### AMap

Store the secret key of AMap. Actually, the more secure way is getting this key by reverse proxy and authentication. If you have interest, you can implement this more security logic. For me when coding the project for graduation? Whatever……

### API

The API module store all backend RESTFul API URL and request method.

### I18N

The i18n module store the locale info. It read the locale info from the localStorage and expose a mutation of setLocal, which will change the current locale to new locale.

## Internalization

In languages folder, store the translation of English and Chinese, beside vue-i18n logic.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure languages folder

For more safety type check of translation strings, the manual declared interface are used for guarantee the translation’s completion.

## View Original Data

### Responsive Layout

The responsive layout is used by all detail route view component. Because the Naïve UI don’t provide the Bootstrap style responsive layout up to my development start, so the layout components from Element Plus (which follow the Bootstrap style) are used for layout.

This part will not be repeated in other route view component sections.

### Data Flow

After the mouse left click (with data selected), the component SelectTabs will emit the data to the view level, the view level do the request and distribute the response data to each components (Map and TreeView).

### About responsive translation

The i18n return a t function for translation. If you want it change with the locale, make sure that:

* If you use it in logic script code, wrap the t function in computed.
* If you use it in template, use it directly will be enough.

This part will not be repeated in other sections.

### Cascade Geo Select

Notice how to control the Cascade Select’ disable:

|  |
| --- |
| const disablePoetBtn = computed(() => {  return vmPoet.value !== undefined && vmPoet.value >= 0 ? false : true;  });  const disableProvince = computed(() => {  return provinceOptions.value?.length ? false : true;  });  const disableCity = computed(() => {  return cityOptions.value?.length ? false : true;  });  const disableLocation = computed(() => {  return locationOptions.value?.length ? false : true;  });  const disableLocationBtn = computed(() => {  return vmLocation.value ? false : true;  }); |

### The Map Component

The Map part use a custom map style (see more in code comments). It use the labels layer and Bezier lines for drawing points and route. The data are formed dynamic. Because the info window in map formed ugly (yes, the official doesn’t provide a more elegant way), so the not scoped CSS are needed.

### Watch the Update time

Watch for a complex prop object will be performance wasted. So, use a simple update time string for the children components data refresh trigger will be better.

This part will not be repeated in other section.

## View Trajectory OLAM

This is the most complex page of front end.

### Data Flow

The select area contains the three OLAP drawer of poet, time and geography. The two level’s v-model make the data flow out easier. After the analysis button clicked, the OLAP selected things will be emitted out to view level. The page view do the request and distribute the response data to each components.

### Data Init

Many charts use the pre-init data which cached in front end for first show. Actually, the init data don’t need to be cached in front end. Just do the async request in onMounted hook can still achieve the same target. If you have interest in this, you can try to rebuild the logic (refer to view Space Scale Visualization’s logic).

This part will not be repeated in other section.

### OLAP

For different dimension poet, time, geography, they roll-up, drill-down’s logic will be different. The logic if not so clearly so you may need to read and understand it twice.

### Two level v-model

Notice that you need a middle var to implement the two level v-model between select are and each drawer.

### Charts

The ECharts component are warped for better resize and better isolation. There is no magic used for charts, just watch the updateTime and refresh data.

## View Time Variant

### Data Flow

The select area select the item and emit to page level, the page do the request and make response data into dynamic sort bar chart.

### The dynamic change

How to make the dynamic change? setTimeout or setInterval? In APTSMAS, I use the setInterval with special logic to make sure it fit my requirements:

|  |
| --- |
| const dynamicHandler = (  previousInterval: number | null,  currentInterval: Ref<number | null>,  chartObj: echarts.ECharts,  chartData: ITimeVariantApiResponse,  currentYear: Ref<number>  ) => {  if (previousInterval !== null && previousInterval !== 0) {  window.clearInterval(previousInterval);  }  currentYear.value = MIN\_YEAR;  currentInterval.value = window.setInterval(() => {  if (MIN\_YEAR <= currentYear.value && currentYear.value <= MAX\_YEAR) {  const option = handleOption(chartData, currentYear.value);  // chartObj.setOption(option, { replaceMerge: ["series", "yAxis"] });  chartObj.setOption(option);  currentYear.value += 1;  } else if (currentInterval.value) {  window.clearInterval(currentInterval.value);  currentInterval.value = null;  }  }, ANIMATE\_UPDATE\_TIME\_MS);  }; |

Every time you call the dynamicHandler, it will clear the previous interval and set a new interval. The interval inside can judge the year valid and clear itself. So the dynamic handler can easily integrated with different item show or time reset function.

## View Trajectory Embedding

### Data Flow

The Select Area select data and emit, then the view level does the request and distribute the response data to each component.

### Script Setup

From this page view component, the script setup syntactic sugar is used for more clearly code logic.

### Map

The map view implements a Bezier lines list for multi poets route draw. Remember to clear them Bezier Lines Object before the new data refresh.

### Charts

Because the Charts are all in-degree and out-degree, so use a v-for to generate them will be easier.

图形用户界面

描述已自动生成

Figure v-for generate charts

## View Space Scale Visualization

### Data Flow

The Map components async request the data and init with itself.

### Dynamic Change

Similar to the content in 2.8.2 shows.

### Loca

Notice that Loca scatter layers must set the data before set the style, else will go into error.

### About 3D

The Loca 3D global really have performance issues. If you have interest, you can try to implement this by ECharts GL 3D Global.

Notice that the AMap and BaiduMap use the different coordinate skewing algorithm.

## Other Worth Notice

### Async Request

Because all page use a loading animate logic, if the backend go into server error, the loading animate will not end. Try to add the exception handler or timeout mechanism to hide loading animate.

### Notice the lifecycle hook

The lifecycle hook should be on the top level of setup function. A common pitfall is that your custom hook have onMounted and call this hook in setup’s onMounted (or other) function.

### Be aware the async

Always notice the async function even though you await it. If you use async and await in one of the regular lifecycle hook, the coding execute logic may not follow your preconceived.

### Computed Complex logic

For better performance, if you template have the complex calculate logic, you should make it into a computed var (computed var have cache mechanism).

### TypeScript

TypeScript is not a sliver bullet, it’s just a security measure. Don’t count everything on it.

# Back End

## Project Struct

The things worth notice are in src folder:

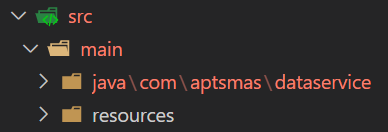


Figure Backend Struct

In java/com/aptsmas/dataservice contains all code logic.

In resources contains all mybatis xml and application config yaml.

图片包含 图形用户界面

描述已自动生成

Figure The Code Logic Folder

The config folder contains the config of the plugins.

The controller folder contains the controller logic and swagger API define.

The entity folder contains struct of complex class (object) which will be used by service or mybatis.

The mapper folder contains the API of mybatis mapper.

The service folder contains the service logic of each function module.

The vo folder contains the response entity class struct.

## Spring Doc

Spring Doc provide the swagger ui of dev environment. And its annotation is in controller and view object define.

## Common Logic

The controller receives the http url mapping, then convert the parameter and distribute to different service. The service use code logic or data fetch from mybatis to construct the view object. Finally, the controller returns the view object as the response.

Java is a strong and static type language, so read its code is easier than read the JavaScript. Notice I use a lot of streams api for simplify code logic.

## Lombok

The Lombok is used for better code experience, you should configure your IDE properly for Lombok.

## Other Things

The newer version of log4j2 is set to avoid the fatal flaw.

I don’t really thing the backend of APTSMAS is difficult to understand. Just be patient, read the code twice and make full advantage of Google, then everything will go clearly.

# Data Processing

## Data Handler

The data handle uses the python script for fast dev. Actually, the designed data protocol is extracted from the exist data. And all my python script can handle the data properly step by step. If you have interest, try to modify and seal them into a single tool to fit each data which follow by data protocol.

## Data Base

The Data Base struct roughly follows the Fact Constellation Pattern. If you have interest, try to modify them to strict follow the Face Constellation Pattern.

# Summary

Visit my dev blog for more useful tips while development: <https://blog.csdn.net/sigmarising>

Good luck!