

# 路由页面整理

**目标** 删除基础模板中附带的多余页面

基础模板帮我们提前内置了一些页面，本章节我们进行一下整理

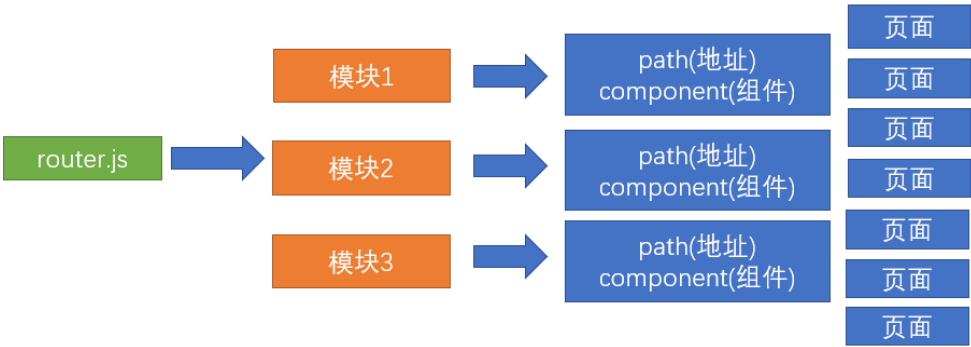
首先，我们需要知道类似这种大型中台项目的页面路由是如何设置的。

## 简单项目



## 当前项目结构

### 中台项目路由



为什么要拆成若干个路由模块呢？

因为复杂中台项目的页面众多，不可能把所有的业务都集中在一个文件上进行管理和维护，并且还有最重要的，前端的页面中主要分为两部分，一部分是所有人都可以访问的，一部分是只有有权限的人才可以访问的，拆分多个模块便于更好的控制

## 静态路由和动态路由



**注意** 这里的动态路由并不是 **路由传参**的动态路由

了解完成路由设计之后，我们对当前的路由进行一下整理

**删除多余的静态路由表** `src/router/index.js`

```
/**
 * constantRoutes
 * a base page that does not have permission requirements
 * all roles can be accessed
 */
export const constantRoutes = [
  {
    path: '/login',
    component: () => import('@views/login/index'),
    hidden: true
  },
  {
    path: '/404',
    component: () => import('@views/404'),
    hidden: true
  },
  {
    path: '/',
    component: Layout,
    redirect: '/dashboard',
    children: [{
      path: 'dashboard',
      name: 'Dashboard',
      component: () => import('@views/dashboard/index'),
      meta: { title: 'Dashboard', icon: 'dashboard' }
    }]
  },
  // 404 page must be placed at the end !!!
  { path: '*', redirect: '/404', hidden: true }
]
```

上面代码，我们只对登录页/404/主页进行了保留

并且我们发现，删除了其他页面之后，左侧导航菜单的数据也只剩下了首页



这是因为左侧导航菜单的数据来源于路由信息

删除多余的路由组件

```
├── dashboard
├── login
└── 404.vue
```

只保留以上三个路由组件的内容，后续慢慢增加

同样的在api目录下，存在多余的api-table.js 一并删除

提交代码

**本节任务：** 完成业务路由页面的整理

## 业务模块页面的快速搭建

**目标：** 快速搭建人资项目的常规业务模块

## 新建模块的页面和路由文件

截止到现在，我们已经完成了一个中台系统的基本轮廓，如图

# 基本轮廓



接下来，我们可以将人力资源需要做的模块快速搭建相应的页面和路由

— dashboard	# 首页
— login	# 登录
— 404	# 404
— departments	# 组织架构
— employees	# 员工
— setting	# 公司设置
— salaries	# 工资
— social	# 社保
— attendances	# 考勤
— approvals	# 审批
— permission	# 权限管理

根据上图中的结构，在views目录下，建立对应的目录，给每个模块新建一个 `index.vue`，作为每个模块的主页

**快速新建文件夹，使用 git bash**

```
$ mkdir departments employees setting salaries social attendances approvals permission
```

每个模块的内容，可以先按照标准的模板建立，如

**员工**

```
<template>
  <div class="dashboard-container">
    <div class="app-container">
      <h2>
        员工
      </h2>
    </div>
  </div>
</template>

<script>
export default {
```

```
}  
</script>  
  
<style>  
  
</style>
```

根据以上的标准建立好对应页面之后，接下来建立每个模块的路由规则

路由模块目录结构

```
|— router                # 路由目录  
|— index.js             # 路由主文件  
|— modules              # 模块目录  
|— departments.js       # 组织架构  
|— employees.js         # 员工  
|— setting.js           # 公司设置  
|— salarys.js           # 工资  
|— social.js            # 社保  
|— attendances.js       # 考勤  
|— approvals.js         # 审批  
|— permission.js        # 权限管理
```

快速创建命令

```
$ touch departments.js employees.js setting.js salarys.js salarys.js social.js  
attendances.js approvals.js permission.js
```

## 设置每个模块的路由规则

每个模块导出的内容表示该模块下的路由规则

如员工 **employees.js**

```
// 导出属于员工的路由规则  
import Layout from '@layout'  
// { path: '', component: '' }  
// 每个子模块 其实 都是外层是layout 组件位于layout的二级路由里面  
export default {  
  path: '/employees', // 路径  
  name: 'employees', // 给路由规则加一个name  
  component: Layout, // 组件  
  // 配置二级路的路由表  
  children: [{  
    path: '', // 这里当二级路由的path什么都不写的时候 表示该路由为当前二级路由的默认路由  
    component: () => import('@views/employees'),  
    // 路由元信息 其实就是存储数据的对象 我们可以在这里放置一些信息  
    meta: {  
      title: '员工管理' // meta属性的里面的属性 随意定义 但是这里为什么要用title呢， 因为左侧导航会读取我们的路由里的meta里面的title作为显示菜单名称  
    }  
  ]  
}
```

```
}
```

```
// 当你的访问地址 是 /employees的时候 layout组件会显示 此时 你的二级路由的默认组件 也会显示
```

上述代码中，我们用到了meta属性，该属性为一个对象，里面可放置自定义属性，主要用于读取一些配置和参数，并且值得 **注意** 的是：我们的meta写了二级默认路由上面，而不是一级路由，因为当存在二级路由的时候，访问当前路由信息访问的就是 **二级默认路由**

大家针对上述的设计，对上面的模块进行快速的搭建

### 提交代码

**本节任务**：完成其他模块的页面和路由的快速搭建

## 静态路由和动态路由临时合并，形成左侧菜单

**目标**：将静态路由和动态路由的路由表进行临时合并

什么叫临时合并？

在第一个小节中，我们讲过了，动态路由是**需要权限**进行访问的，但是**权限的动态路由访问是很复杂的**，我们稍后在进行讲解，所以为了更好地看到效果，我们可以**先将 静态路由和动态路由进行合并**

**路由主文件** `src/router/index.js`

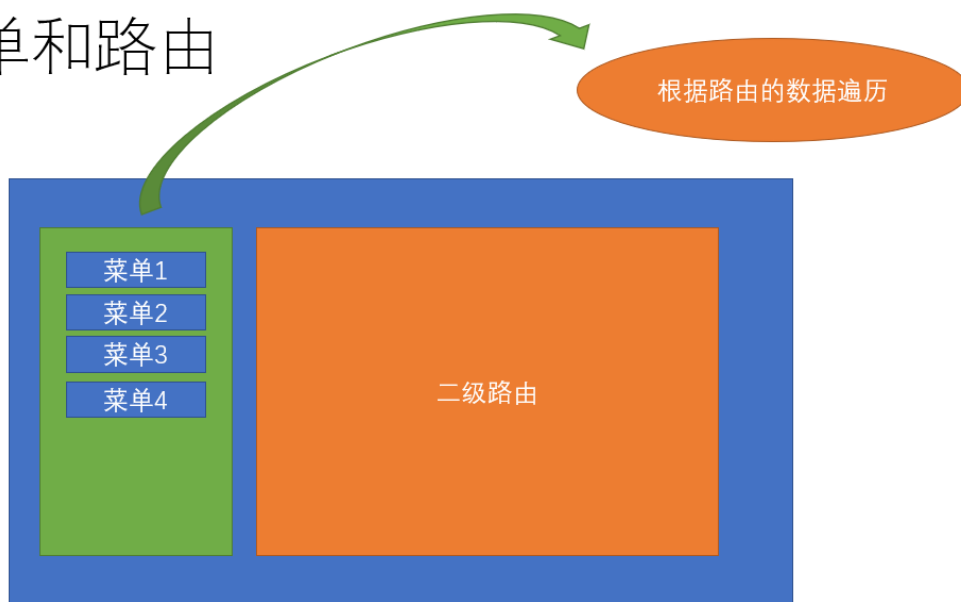
```
// 引入多个模块的规则
import approvalsRouter from './modules/approvals'
import departmentsRouter from './modules/departments'
import employeesRouter from './modules/employees'
import permissionRouter from './modules/permission'
import attendancesRouter from './modules/attendances'
import salarysRouter from './modules/salarys'
import settingRouter from './modules/setting'
import socialRouter from './modules/social'

// 动态路由
export const asyncRoutes = [
  departmentsRouter,
  employeesRouter,
  settingRouter,
  permissionRouter,
  socialRouter,
  attendancesRouter,
  salarysRouter,
  approvalsRouter,
]

const createRouter = () => new Router({
  // mode: 'history', // require service support
  scrollBehavior: () => ({ y: 0 }), // 管理滚动行为 如果出现滚动 切换就让 让页面回到顶部
  routes: [...constantRoutes, ...asyncRoutes] // 临时合并所有的路由
})
```

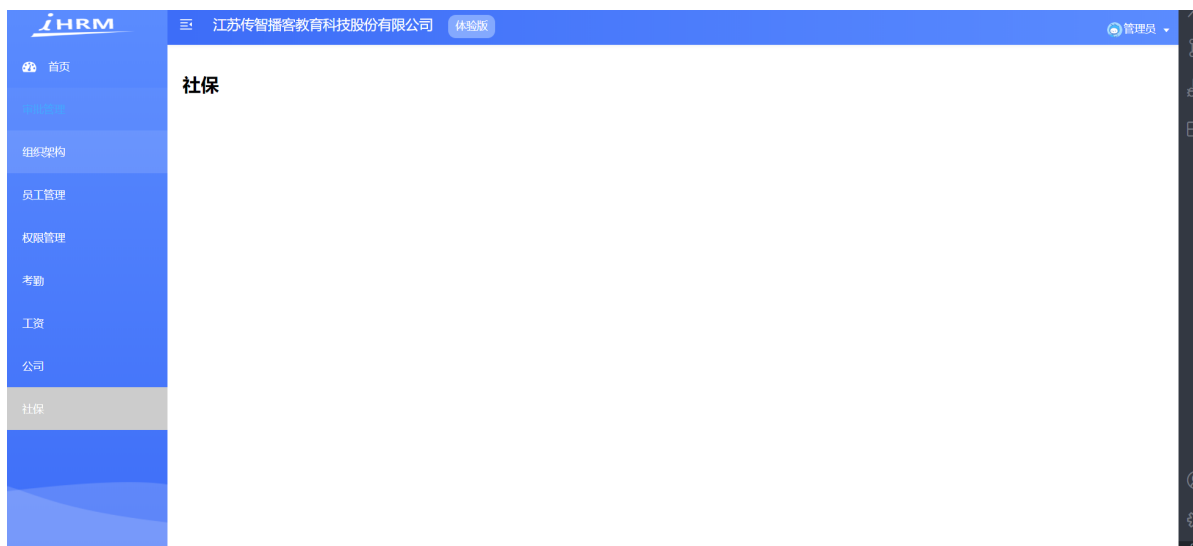
通过上面的操作，我们将静态路由和动态路由进行了合并

## 菜单和路由



当我们合并权限完成，我们惊奇的发现页面效果已经左侧的导航菜单 =》路由页面

这是之前基础模板中对于左侧导航菜单的封装



### 提交代码

**本节任务：** 将静态路由和动态路由临时合并，形成左侧菜单

## 左侧菜单的显示逻辑，设置菜单图标

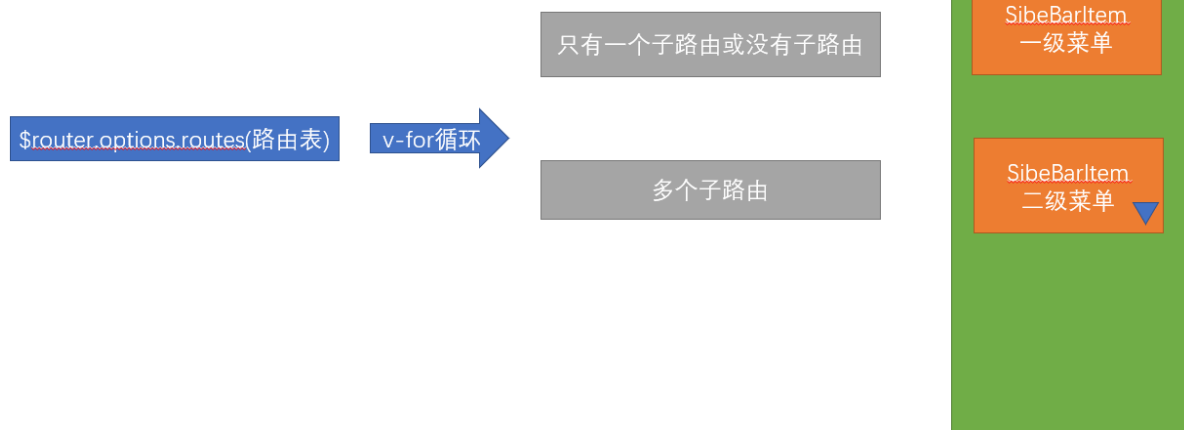
**目标** 解析左侧菜单的显示逻辑， 设置左侧导航菜单的图标内容

上小节中，我们集成了路由，菜单就显示内容了，这是为什么？

### 阅读左侧菜单代码

我们发现如图的逻辑

# 左侧菜单



由于，该项目不需要二级菜单的显示，所以对代码进行一下处理，只保留一级菜单路由

`src/layout/components/Sidebar/SidebarItem.vue`

```
<template>
  <div v-if="!item.hidden">
    <template v-if="hasOneShowingChild(item.children,item) &&
(!onlyOneChild.children||onlyOneChild.noShowingChildren)&&!item.alwaysShow">
      <app-link v-if="onlyOneChild.meta" :to="resolvePath(onlyOneChild.path)">
        <el-menu-item :index="resolvePath(onlyOneChild.path)" :class="{ 'submenu-
title-noDropdown': !isNest}">
          <item :icon="onlyOneChild.meta.icon||(item.meta&&item.meta.icon)"
:title="onlyOneChild.meta.title" />
        </el-menu-item>
      </app-link>
    </template>

    <!-- <el-submenu v-else ref="subMenu" :index="resolvePath(item.path)" popper-
append-to-body>
      <template slot="title">
        <item v-if="item.meta" :icon="item.meta && item.meta.icon"
:title="item.meta.title" />
      </template>
      <sidebar-item
        v-for="child in item.children"
        :key="child.path"
        :is-nest="true"
        :item="child"
        :base-path="resolvePath(child.path)"
        class="nest-menu"
      />
    </el-submenu> -->
  </div>
</template>
```

**本节注意：**通过代码发现，当路由中的属性 `hidden` 为 `true` 时，表示该路由不显示在左侧菜单中



与此同时，我们发现左侧菜单并不协调，是因为缺少图标。在本项目中，我们的图标采用了SVG的组件

左侧菜单的图标实际上读取的是meta属性的icon，这个icon需要我们提前放置在 `src/icons/svg` 目录下

该资源已经在菜单svg目录中提供，请将该目录下的所有svg放到 `src/icons/svg` 目录下

具体的icon名称可参考[线上地址](#)

functional为true，表示该组件为一个函数式组件

函数式组件：没有data状态，没有响应式数据，只会接收props属性，没有 this，他就是一个函数

### 模块对应icon

— dashboard	# dashboard
— departments	# tree
— employees	# people
— setting	# setting
— salarys	# money
— social	# table
— attendances	# skill
— approvals	# tree-table
— permission	# lock

```
component: () => import('@/views/departments'); // 组织架构图
meta: {
  title: '组织架构', // 标记当前路由规则的中文名称 后续在做左侧菜单时 使用
  icon: 'tree'
}
```

按照对应的icon设置图标

**本节任务：** 理解左侧菜单的生成逻辑，并设置左侧菜单的图标