第2章 数据库设计与前端框架

学习目标：

- 理解多租户的数据库设计方案
- 熟练使用PowerDesigner构建数据库模型
- 理解前端工程的基本架构和执行流程
- 完成前端工程企业模块开发

1 多租户SaaS平台的数据库方案

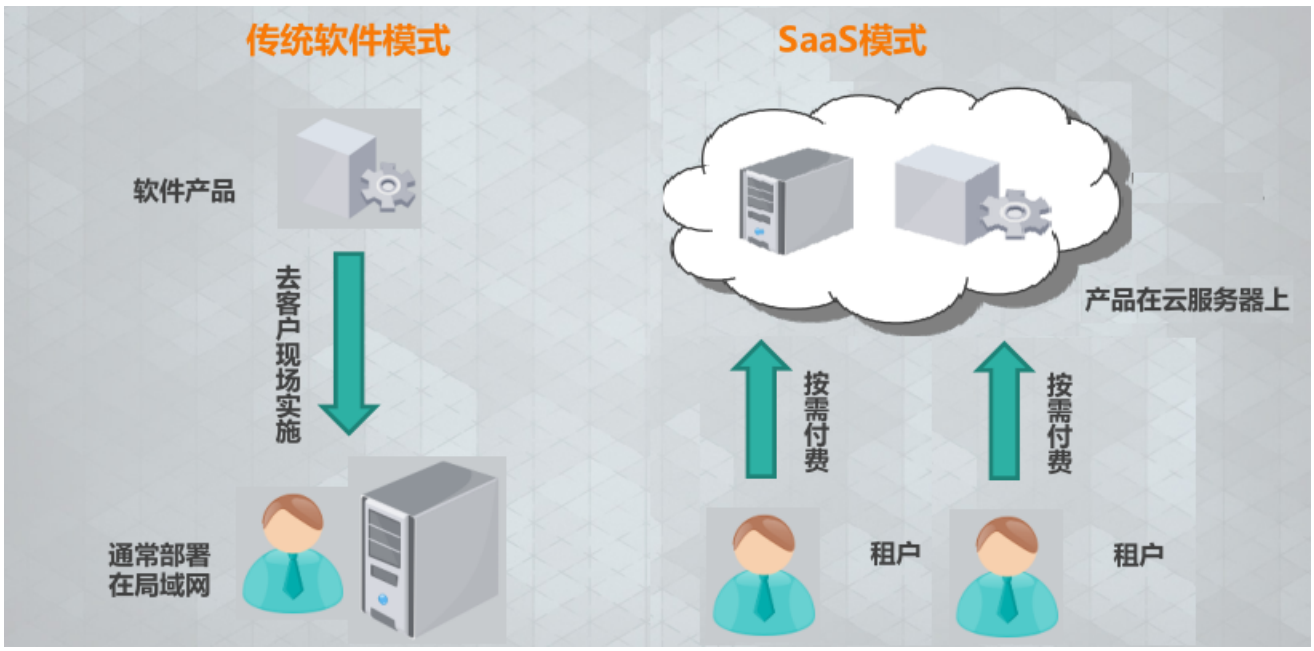
1.1 多租户是什么

多租户技术（Multi-TenancyTechnology）又称多重租赁技术：是一种软件架构技术，是实现如何在多用户环境下（此处的多用户一般是面向企业用户）共用相同的系统或程序组件，并且可确保各用户间数据的隔离性。简单讲：在一台服务器上运行单个应用实例，它为多个租户（客户）提供服务。从定义中我们可以理解：多租户是一种架构，目的是为了让多用户环境下使用同一套程序，且保证用户间数据隔离。那么重点就很浅显易懂了，多租户的重点就是同一套程序下实现多用户数据的隔离

1.2 需求分析

传统软件模式，指将软件产品进行买卖，是一种单纯的买卖关系，客户通过买断的方式获取软件的使用权，软件的源码属于客户所有，因此传统软件是部署到企业内部，不同的企业各自部署一套自己的软件系统

Saas模式，指服务提供商提供的一种软件服务，应用统一部署到服务提供商的服务器上，客户可以根据自己的实际需求按需付费。用户购买基于WEB的软件，而不是将软件安装在自己的电脑上，用户也无需对软件进行定期的维护与管理



在SaaS平台里需要使用共用的数据中心以单一系统架构与服务提供多数客户端相同甚至可定制化的服务，并且仍可以保障客户的数据正常使用。由此带来了新的挑战，就是如何对应用数据进行设计，以支持多租户，而这种设计的思路，是要在数据的共享、安全隔离和性能间取得平衡。

1.3 多租户的数据库方案分析

目前基于多租户的数据库设计方案通常有如下三种：

- 独立数据库
- 共享数据库、独立 Schema
- 共享数据库、共享数据表

1.3.1 独立数据库

独立数据库：每个租户一个数据库。

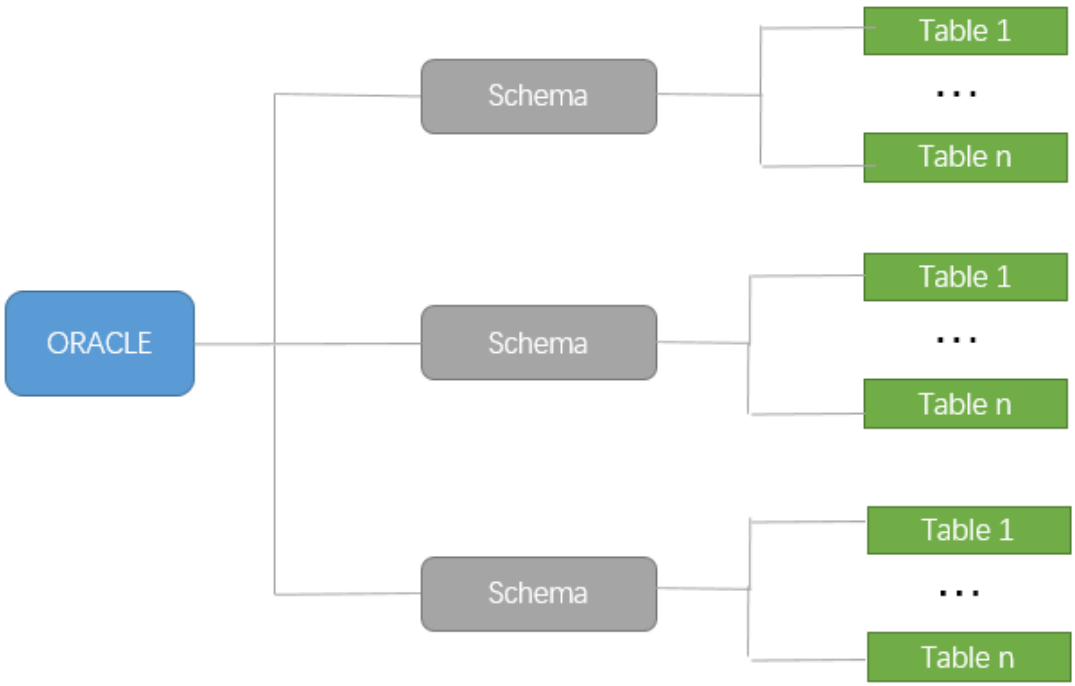
- 优点：为不同的租户提供独立的数据库，有助于简化数据模型的扩展设计，满足不同租户的独特需求；如果出现故障，恢复数据比较简单。
- 缺点： 增多了数据库的安装数量，随之带来维护成本和购置成本的增加

这种方案与传统的一个客户、一套数据、一套部署类似，差别只在于软件统一部署在运营商那里。由此可见此方案用户数据隔离级别最高，安全性最好，但是成本较高

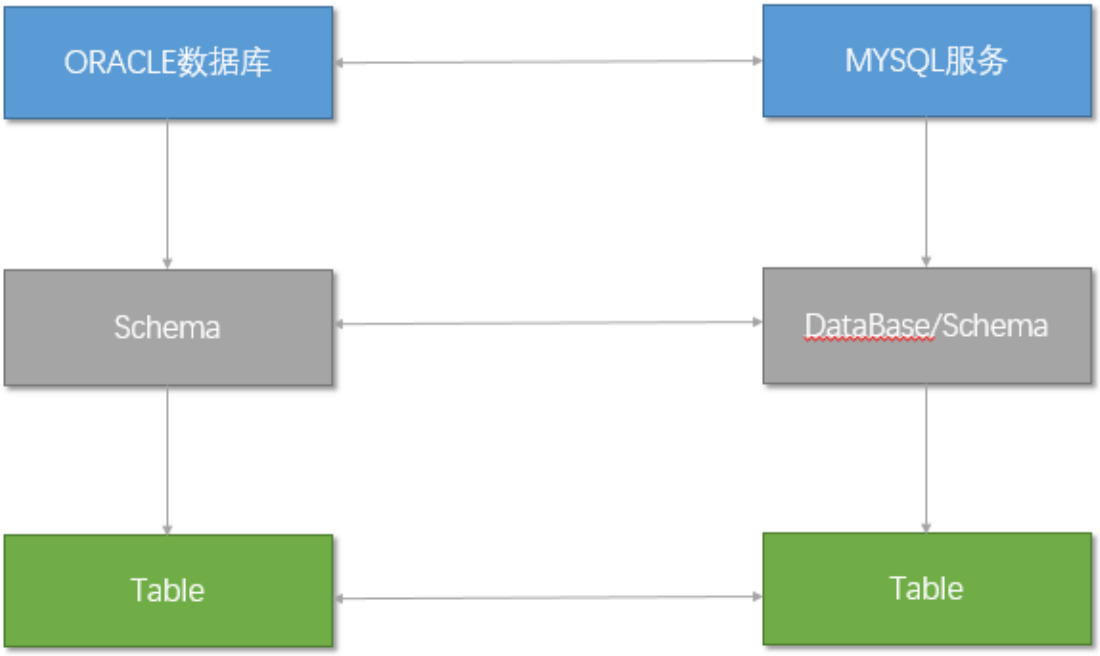
1.3.2 共享数据库、独立 Schema

（1）什么是Schema

- oracle数据库：在oracle中一个数据库可以具有多个用户，那么一个用户一般对应一个Schema，表都是建立在Schema中的,（可以简单的理解：在oracle中一个用户一套数据库表）



- mysql数据库：mysql数据中的schema比较特殊，并不是数据库的下一级，而是等同于数据库。比如执行create schema test 和执行create database test效果是一模一样的



Oracle与Mysql对比

共享数据库、独立 Schema：即多个或所有的租户使用同一个数据库服务（如常见的ORACLE或MYSQL数据库），但是每个租户一个Schema。

- 优点： 为安全性要求较高的租户提供了一定程度的逻辑数据隔离，并不是完全隔离；每个数据库可支持更多的租户数量。
- 缺点： 如果出现故障，数据恢复比较困难，因为恢复数据库将牵涉到其他租户的数据； 如果需要跨租户统计数据，存在一定困难。

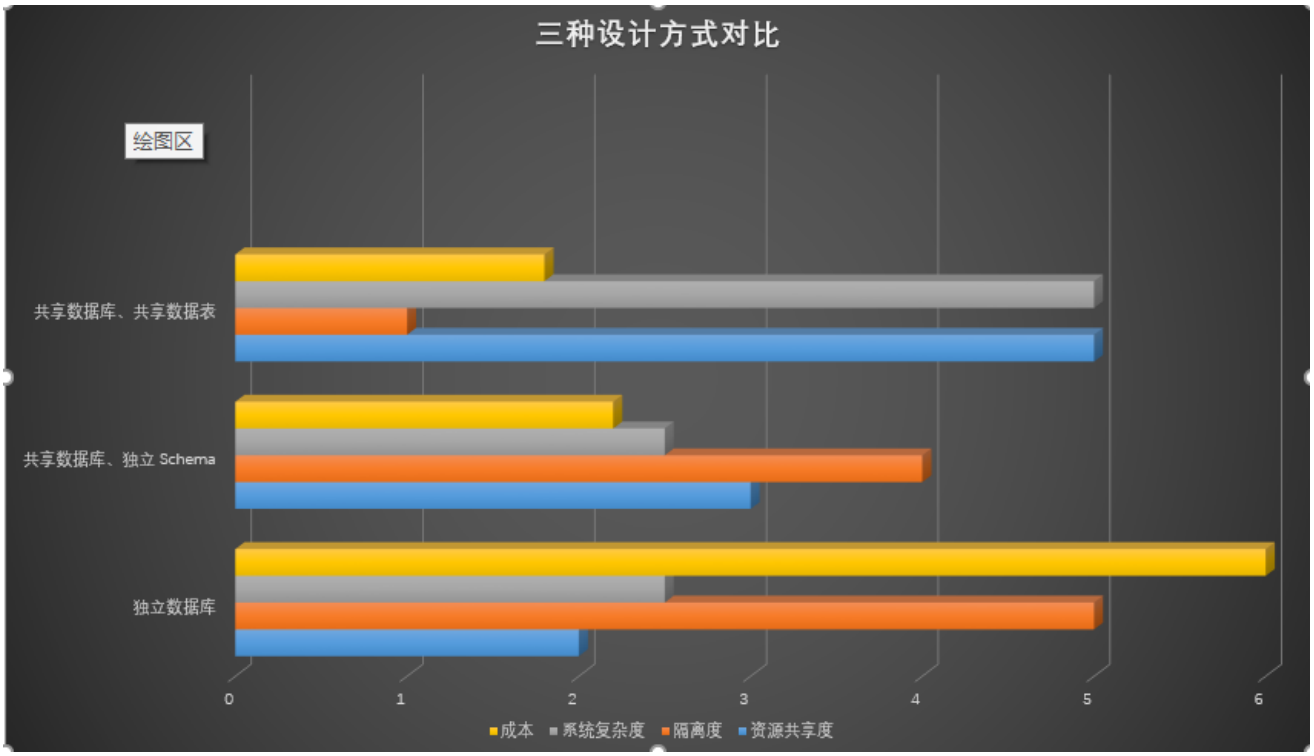
这种方案是方案一的变种。只需要安装一份数据库服务，通过不同的Schema对不同租户的数据进行隔离。由于数据库服务是共享的，所以成本相对低廉。

1.3.3 共享数据库、共享数据表

共享数据库、共享数据表：即租户共享同一个Database，同一套数据库表（所有租户的数据都存放在一个数据库的同一套表中）。在表中增加租户ID等租户标志字段，表明该记录是属于哪个租户的。

- 优点：所有租户使用同一套数据库，所以成本低廉。
- 缺点：隔离级别最低，安全性最低，需要在设计开发时加大对安全的开发量，数据备份和恢复最困难。

这种方案和基于传统应用的数据库设计并没有任何区别，但是由于所有租户使用相同的数据库表，所以需要做好对每个租户数据的隔离安全性处理，这就增加了系统设计和数据管理方面的复杂程度。



1.4 SAAS-HRM数据库设计

在SAAS-HRM平台中，分为了试用版和正式版。处于教学的目的，试用版采用共享数据库、共享数据表的方式设计。正式版采用基于mysql的共享数据库、独立 Schema设计（后续课程）。

2 数据库设计与建模

2.1 数据库设计的三范式

三范式：

1.第一范式（1NF）：确保每一列的原子性（做到每列不可拆分）

2.第二范式（2NF）：在第一范式的基础上，非主字段必须依赖于主字段（一个表只做一件事）

3.第三范式（3NF）：在第二范式的基础上，消除传递依赖

反三范式：

反三范式是基于第三范式所调整的，没有冗余的数据库未必是最好的数据库，有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。

2.2 数据库建模

了解了数据的设计思想，那对于数据库表的表设计应该怎么做呢？答案是数据库建模

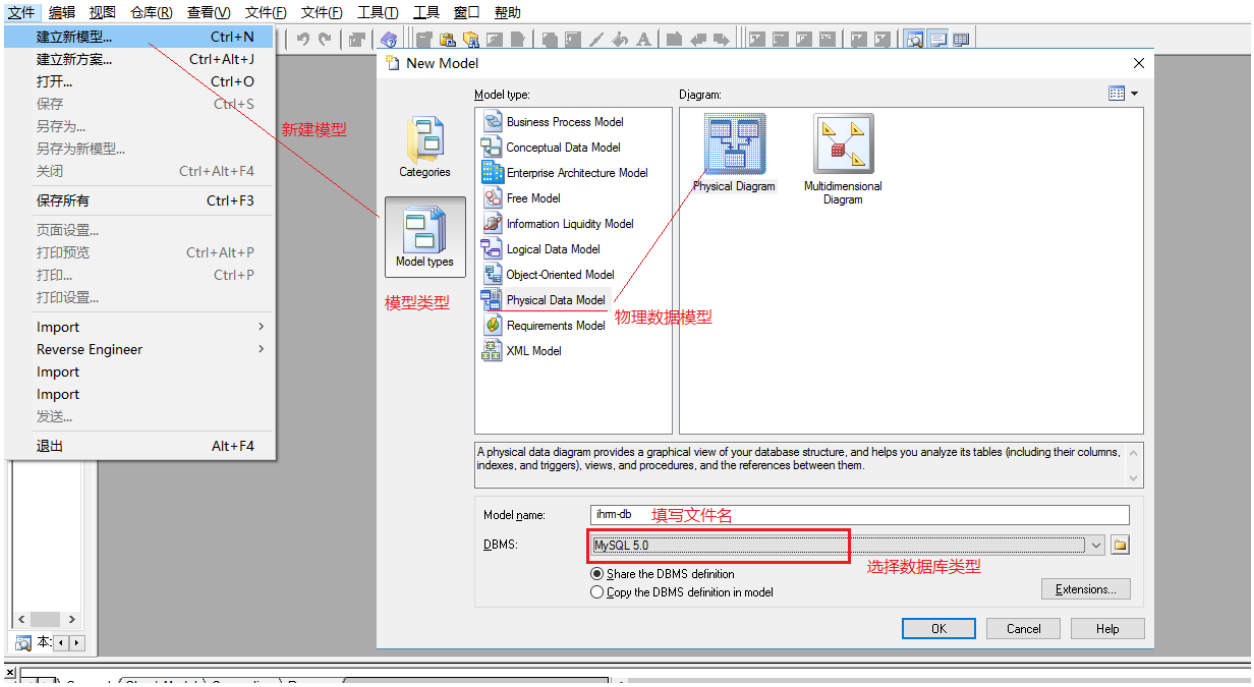
数据库建模：在设计数据库时，对现实世界进行分析、抽象、并从中找出内在联系，进而确定数据库的结构。它主要包括两部分内容：确定最基本的数据结构；对约束建模。

2.2.1 建模工具

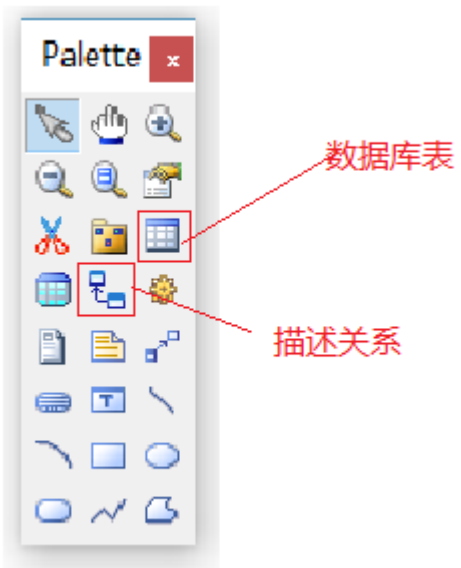
对于数据模型的建模，最有名的要数PowerDesigner，PowerDesigner是在中国软件公司中非常有名的，其易用性、功能、对流行技术框架的支持、以及它的模型库的管理理念，都深受设计师们喜欢。他的优势在于：不用在使用create table等语句创建表结构，数据库设计人员只关注如何进行数据建模即可，将来的数据库语句，可以自动生成

2.2.2 使用pd建模

1. 选择新建数据库模型 打开PowerDesigner，文件->建立新模型->model types（选择类型）->Physical Data Model(物理模型)

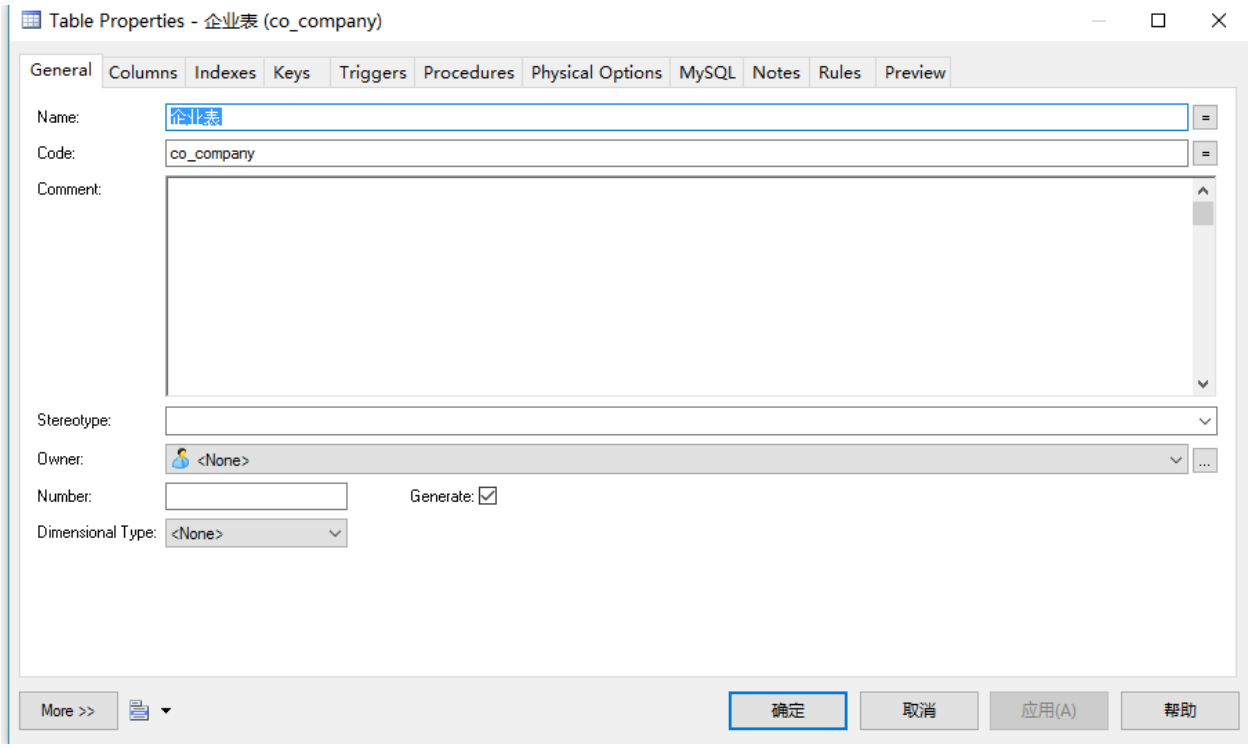


2. 控制面板

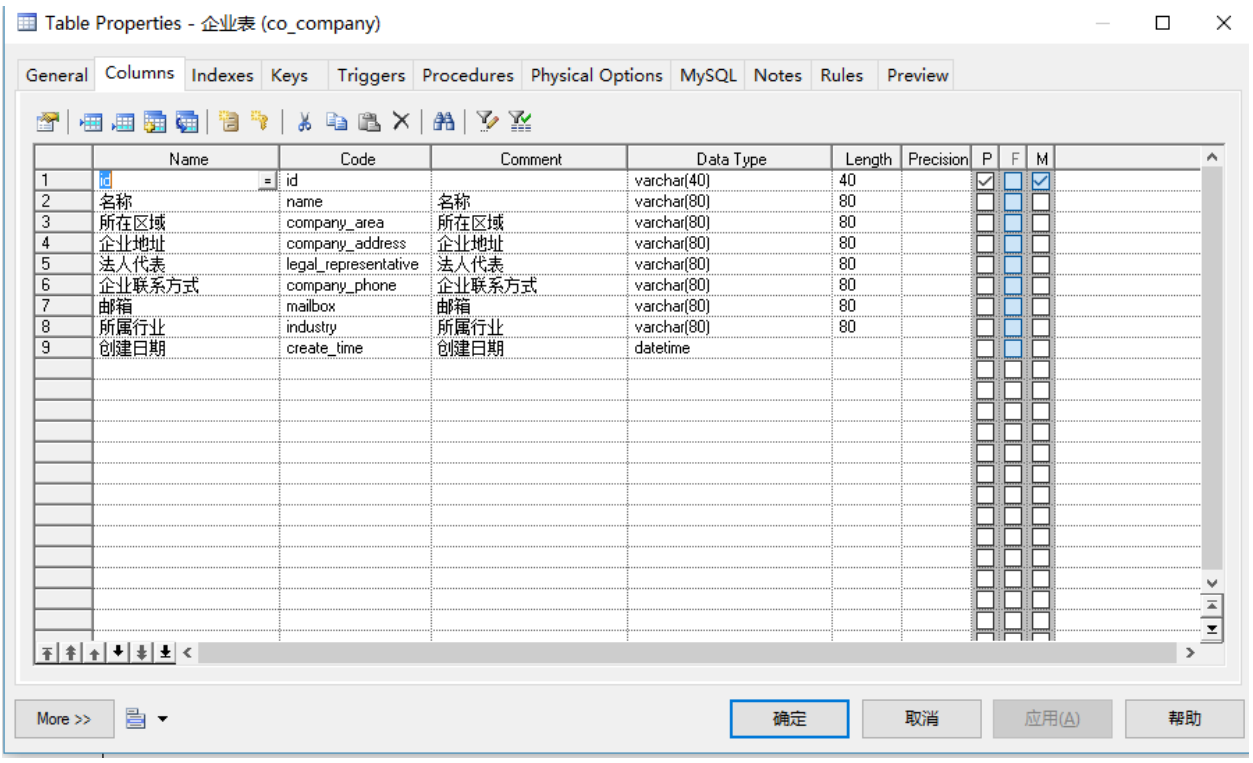


3. 创建数据库表

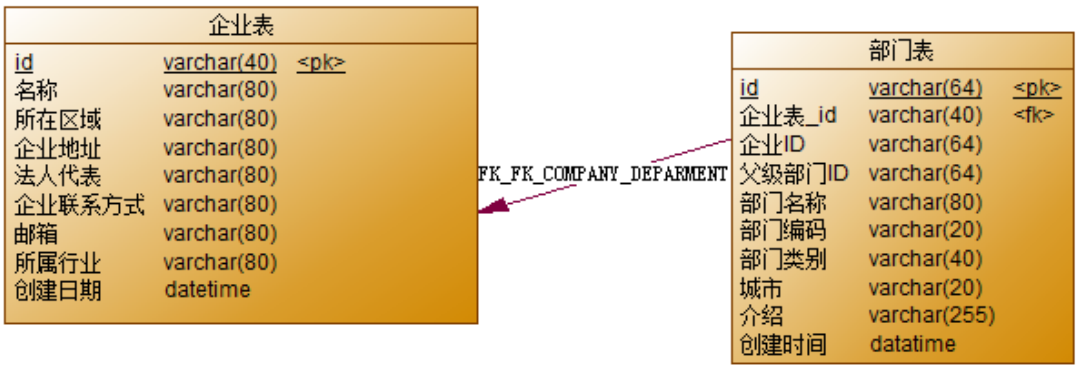
点即面板按钮中的创建数据库按钮创建数据库模型



切换columns标签，可以对表中的所有字段进行配置



如果基于传统的数据库设计中存在外键则可以使用面版中的Reference配置多个表之间的关联关系，效果如下图



4. 导出sql

菜单->数据库（database）->生成数据库表结构（Generate Database）

2-10

3 前端框架

3.1 脚手架工程

此项目采用目前比较流行的前后端分离的方式进行开发。前端是在传智播客研究院开源的前端框架（黑马Admin商用后台模板）的基础上进行的开发。

官网上提供了非常基础的脚手架，如果我们使用官网的脚手架需要自己写很多代码比如登陆界面、主界面菜单样式等内容。 课程已经提供了功能完整的脚手架，我们可以拿过来在此基础上开发，这样可以极大节省我们开发的时间。

- 技术栈
  - vue 2.5++
  - elementUI 2.2.2
  - vuex
  - axios
  - vue-router
  - vue-i18n
- 前端环境
  - node 8.++
  - npm 5.++

3.2 启动与安装

官网上提供了非常基础的脚手架，如果我们使用官网的脚手架需要自己写很多代码比如登陆界面、主界面菜单样式等内容。 课程已经提供了功能完整的脚手架，我们可以拿过来在此基础上开发，这样可以极大节省我们开发的时间。

（1）解压提供的资源包

（2）在命令提示符进入该目录，输入命令：

```
cnpm install
```

> 通过淘宝镜像下载安装所有的依赖，几分钟后下载完成
>
> 如果没有安装淘宝镜像，请使用npm install

（3）关闭语法检查

打开 `config/index.js` 将useEslint的值改为false。

```
 useEslint: false,
```

> 此配置作用: 是否开启语法检查，语法检查是通过ESLint 来实现的。我们现在科普一下,什么是ESLint : ESLint是一个语法规则和代码风格的检查工具，可以用来保证写出语法正确、风格统一的代码。如果我们开启了Eslint，也就意味着要接受它非常苛刻的语法检查，包括空格不能少些或多些，必须单引不能双引，语句后不可以写分号等等，这些规则其实是可以设置的。我们作为前端的初学者，最好先关闭这种校验，否则会浪费很多精力在语法的规范性上。如果以后做真正的企业级开发，建议开启

（4）输入命令：

```
npm run dev
```

3.3 工程结构

整个前端工程的工程目录结构如下：

```
├── assets                          | 资源
├── build                           | webpack编译配置
├── config                          | 全局变量
├── src                             | 源码
|   ├── api                         | 数据请求
|   ├── assets                      | 资源
|   ├── components                  | 组件
|   ├── mixins                      | mixins
|   ├── filters                     | vue filter
|   ├── icons                       | 图标
|   ├── lang                        | 多语言
|   ├── router                      | 路由
|   ├── store                       | 数据
|   ├── styles                      | 样式
|   ├── utils                       | 工具函数库
|   ├── module-dashboard            | 框架程序
|   |   ├── assets
|   |   ├── components
|   |   ├── pages
|   |   ├── router
|   |   └── store
|   ├── module-example              | 示例程序
|   |   ├── assets
```

```
|   |   ├── components
|   |   ├── pages
|   |   ├── router
|   |   └── store
|   ├── App.vue                    | app
|   ├── main.js                    | 主引导
|   └──  errorLog.js               | vue全局错误捕捉
├── dist                           | 编译发布目录
├── README.md
├── index.html                     | 页面模板
├── package.json                   | npn包配置
├── static
└── test                           | 测试
    ├── e2e
    └── unit
```

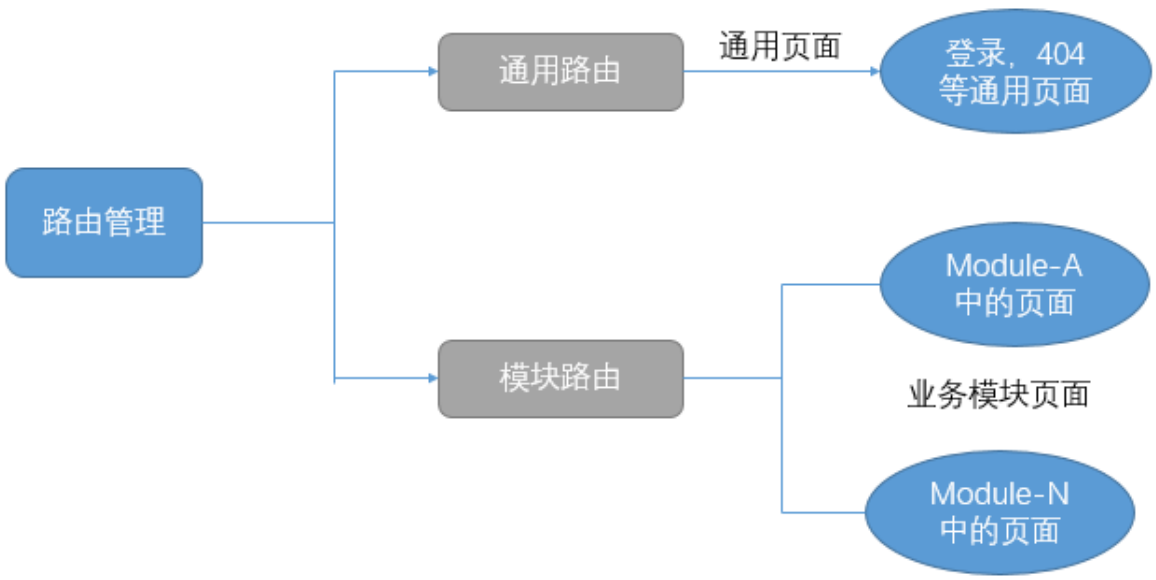
3.4 执行流程分析

3.4.1 路由和菜单

路由和菜单是组织起一个后台应用的关键骨架。本项目侧边栏和路由是绑定在一起的，所以你只有在 `@/router/index.js` 下面配置对应的路由，侧边栏就能动态的生成了。大大减轻了手动编辑侧边栏的工作量。当然这样就需要在配置路由的时候遵循很多的约定

这里的路由分为两种，`constantRouterMap` 和 `asyncRouterMap`。

- **constantRouterMap** 代通用页面。
- **asyncRouterMap** 代表那些业务中通过 `addRouters` 动态添加的页面。



3.4.2 前端数据交互

一个完整的前端 UI 交互到服务端处理流程是这样的：

1. UI 组件交互操作；

2. 调用统一管理的 api service 请求函数；
3. 使用封装的 request.js 发送请求；
4. 获取服务端返回；
5. 更新 data；

从上面的流程可以看出，为了方便管理维护，统一的请求处理都放在 `src/api` 文件夹中，并且一般按照 model 纬度进行拆分文件

```
api/
  frame.js
  menus.js
  users.js
  permissions.js
  ...
```

其中，`src/utils/request.js` 是基于 axios 的封装，便于统一处理 POST，GET 等请求参数，请求头，以及错误提示信息等。具体可以参看 request.js。 它封装了全局 `request拦截器`、`respone拦截器`、`统一的错误处理`、`统一做了超时，baseURL设置等`

4 企业管理

4.1 需求分析

在通用页面配置企业管理模块，完成企业的基本操作

4.2 搭建环境

4.2.1 新增模块

（1）手动创建

方式一：在src目录下创建文件夹，命名规则：module-模块名称（ ）

在文件夹下按照指定的结构配置assets，components，pages，router，store等文件

（2）使用命令自动创建

安装命令行工具

```
npm install -g itheima-cli
```

执行命令

```
itheima moduleAdd saas-clients
`saas-clients` 是新模块的名字
自动创建这些目录和文件
|   ├── module-saas-clients     | saas-clients模块主目录
|   |   ├── assets              | 资源
|   |   ├── components          | 组件
|   |   ├── pages               | 页面
|   |   |   └── index.vue       | 示例
|   |   ├── router              | 路由
|   |   |   └── index.js        | 示例
|   |   └── store               | 数据
|   |       └── app.js          | 示例
```

- 每个模块所有的素材、页面、组件、路由、数据，都是独立的，方便大型项目管理，
- 在实际项目中会有很多子业务项目，它们之间的关系是平行的、低耦合、互不依赖。

4.2.2 构造模拟数据

（1）在`/src/mock`中添加模拟数据company.js

```
import Mock from 'mockjs'
import { param2Obj } from '@/utils'



const List = []
const count = 100

for (let i = 0; i < 3; i++) {
    let data = {
        id: "1"+i,
        name: "企业"+i,
        managerId: "string",
        version: "试用版v1.0",
        renewalDate: "2018-01-01",
        expirationDate: "2019-01-01",
        companyArea: "string",
        companyAddress: "string",
        businessLicenseId: "string",
        legalRepresentative: "string",
        companyPhone: "13800138000",
        mailbox: "string",
        companySize: "string",
        industry: "string",
        remarks: "string",
        auditState: "string",
        state: "1",
        balance: "string",
        createTime: "string"
    }
  List.push(data)
}
```

```
export default {
  list: () => {
    return {
        code: 10000,
        success: true,
        message: "查询成功",
        data:List
    }
  },
  sassDetail:() => {
    return {
      code: 10000,
      success: true,
      message: "查询成功",
      data:{
        id: "10001",
        name: "测试企业",
        managerId: "string",
        version: "试用版v1.0",
        renewalDate: "2018-01-01",
        expirationDate: "2019-01-01",
        companyArea: "string",
        companyAddress: "string",
        businessLicenseId: "string",
        legalRepresentative: "string",
        companyPhone: "13800138000",
        mailbox: "string",
        companySize: "string",
        industry: "string",
        remarks: "string",
        auditState: "string",
        state: "1",
        balance: "string",
        createTime: "string"
      }
    }
  }
}
```

（2）配置模拟API接口拦截规则

在 `/src/mock/index.js` 中配置模拟数据接口拦截规则

```
import Mock from 'mockjs'
import TableAPI from './table'
import ProfileAPI from './profile'
import LoginAPI from './login'
import CompanyAPI from './company'

Mock.setup({
  //timeout: '1000'
```

```
})

//如果发送请求的api路径匹配，拦截
//第一个参数匹配的请求api路径，第二个参数匹配请求的方式，第三个参数相应数据如何替换
Mock.mock(/\/table\/list\.*/, 'get', TableAPI.list)
//获取用户信息
Mock.mock(/\/frame\/profile/, 'post', ProfileAPI.profile)
Mock.mock(/\/frame\/login/, 'post', LoginAPI.login)

//配置模拟数据接口
//  /company/12
Mock.mock(/\/company\/+/, 'get', CompanyAPI.sassDetail)//根据id查询
Mock.mock(/\/company/, 'get', CompanyAPI.list)  //访问企业列表
```

4.2.3 注册模块

编辑 `src/main.js`

```
...
/*
* 注册 - 业务模块
*/
import dashboard from '@/module-dashboard/' // 面板
import saasClients from '@/module-saas-clients/' //刚新添加的 企业管理
Vue.use(dashboard, store)
Vue.use(saasClients, store)
...
```

4.2.4 配置路由菜单

打开刚才自动创建的 `/src/module-saas-clients/router/index.js`

```
import Layout from '@/module-dashboard/pages/layout'
const _import = require('@/router/import_' + process.env.NODE_ENV)

export default [
  {
    path: '/saas-clients',
    component: Layout,
    redirect: 'noredirect',
    name: 'saas-clients',
    meta: {
      title: 'SaaS企业管理',
      icon: 'international'
    },
    root: true,
    children: [
      {
        path: 'index',
```

```
        name: 'saas-clients-index',
        component: _import('saas-clients/pages/index'),
        meta: {title: 'SaaS企业', icon: 'international', noCache: true}
      }
    ]
  }
]
```

4.2.5 编写业务页面

创建 `/src/module-saas-clients/pages/index.vue`

```
<template>
  <div class="dashboard-container">
    saas企业管理
  </div>
</template>

<script>

export default {
  name: 'saasClintList',
  components: {},
  data() {
    return {
    }
  },
  computed: {
  },
  created() {
  }
}
</script>
```

> 注意文件名 `驼峰格式` `首字小写`
>
> 页面请放在目录 `/src/module-saas-clients/pages/`
>
> 组件请放在目录 `/src/module-saas-clients/components/`
>
> 页面路由请修改 `/src/module-saas-clients/router/index.js`

4.3 企业操作

4.3.1 创建api

在api/base目录下创建企业数据交互的API（saasClient.js）

```
import {createAPI, createFormAPI} from '@/utils/request'

export const list = data => createAPI('/company', 'get', data)
export const detail = data => createAPI(`/company/${data.id}`, 'get', data)
```

4.3.1 企业列表

```
<template>
  <div class="dashboard-container">
    <div class="app-container">
      <el-card shadow="never">
        <!--elementui的table组件
          data：数据模型
         -->
        <el-table  :data="dataList"  border style="width: 100%">
            <!--el-table-column ： 构造表格中的每一列
              prop： 数组中每个元素对象的属性名
            -->
            <el-table-column fixed type="index" label="序号" width="50"></el-table-
column>
            <el-table-column fixed prop="name" label="企业名称"  width="200"></el-table-
column>
            <el-table-column fixed prop="version" label="版本"  width="150"></el-table-
column>
            <el-table-column fixed prop="companyphone" label="联系电话"  width="150">
</el-table-column>
            <el-table-column fixed prop="expirationDate" label="截至时间"  width="150">
</el-table-column>
            <el-table-column fixed prop="state" label="状态"  width="150">
              <!--scope:传递当前行的所有数据 -->
              <template slot-scope="scope">
              <!--开关组件
                  active-value：激活的数据值
                  active-color：激活的颜色
                  inactive-value：未激活
                  inactive-color：未激活的颜色
               -->
              <el-switch
                  v-model="scope.row.state"
                  inactive-value="0"
                  active-value="1"
                  disabled
                  active-color="#13ce66"
                  inactive-color="#ff4949">
                </el-switch>
              </template>
            </el-table-column>
            <el-table-column fixed="right" label="操作" width="150">
              <template slot-scope="scope">
```

```
                <router-link :to="'/saas-clients/details/'+scope.row.id">查看</router-
link>
              </template>
            </el-table-column>
          </el-table>
      </el-card>
     </div>
  </div>
</template>

<script>
import {list} from '@/api/base/saasClient'
export default {
  name: 'saas-clients-index',
  data () {
    return {
      dataList:[]
    }
  },
  methods: {
    getList() {
      //调用API发起请求
      //res=响应数据
      list().then(res => {
        this.dataList = res.data.data
      })
    }
  },
  // 创建完毕状态
  created() {
    this.getList()
  }
}
</script>

<style rel="stylesheet/scss" lang="scss" scoped>
.alert {
  margin: 10px 0px 0px 0px;
}
.pagination {
  margin-top: 10px;
  text-align: right;
}
</style>
```

4.3.2 企业详情

（1）配置路由

在`/src/module-saas-clients/router/index.js`添加新的子路由配置

```
      {
        path: 'details/:id',
        name: 'saas-clients-details',
        component: _import('saas-clients/pages/sass-details'),
        meta: {title: 'SaaS企业详情', icon: 'international', noCache: false}
      }
```

（2）定义公共组件

在 `/src/module-saas-clients/components/` 下创建公共的组件页面 `enterprise-info.vue`

```
<template>
  <div class="boxInfo">
    <!-- 表单内容 -->
    <div class="formInfo">
      <div>
        <div class="boxMain">
          <el-form ref="form" :model="formData" label-width="215px" label-
position="right">
              <el-form-item class="formInfo" label="公司名称：">
                <el-input v-model="formData.name" class="inputW" disabled></el-input>
              </el-form-item>
              <el-form-item class="formInfo" label="公司地区：">
                <el-input v-model="formData.companyArea" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="公司地址：">
                <el-input v-model="formData.companyAddress" class="inputW" disabled>
</el-input>
              </el-form-item>
              <el-form-item class="formInfo" label="审核状态：">
                <el-input v-model="formData.auditState" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="营业执照：">
                <span v-for="item in fileList" :key='item.id' class="fileImg">
                  <img :src="item.url">
                </span>
              </el-form-item>
              <el-form-item class="formInfo" label="法人代表：">
                <el-input v-model="formData.legalRepresentative" class="inputW"
disabled></el-input>
              </el-form-item>
              <el-form-item class="formInfo" label="公司电话：">
                <el-input v-model="formData.companyPhone" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="邮箱：">
                <el-input v-model="formData.mailbox" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="公司规模：">
```

```
                <el-input v-model="formData.companySize" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="所属行业：">
                <el-input v-model="formData.industry" class="inputW" disabled></el-
input>
              </el-form-item>
              <el-form-item class="formInfo" label="备注：">
                <el-input type="textarea" v-model="formData.remarks" class="inputW">
</el-input>
              </el-form-item>
            </el-form>
            <div slot="footer" class="dialog-footer">
              <el-button type="primary" @click="handleSub('1')">审核</el-button>
              <el-button @click="handleSub('2')">拒绝</el-button>

            </div>
          </div>
        </div>
      </div>
    </div>
</template>

<script>
import { auditDetail } from '@/api/base/sassClients'
import { imgDownload } from '@/api/base/baseApi'
var _this = null
export default {
  name: 'userInfo',
  components: {},
  props: ['formData'],
  data() {
    return {
      fileList: []
    }
  },
  methods: {
    // 业务方法
    // 界面交互
    handleSub(state) {
      auditDetail({
        id: this.formData.id,
        remarks: this.formData.remarks,
        state: state
      }).then(() => {
        if (state === '1') {
          this.$message.success('恭喜你，审核成功！')
        }
        if (state === '2') {
          this.$message.success('已拒绝审核！')
        }

        this.$emit('getObjInfo', this.formData)
```

```
        })
      },
      // 图片 blob 流转化为可用 src
      imgHandle(obj) {
        return window.URL.createObjectURL(obj)
      },
      // 图片下载
      fillDownload(fid) {

      }
    },
    // 挂载结束
    mounted: function() {},
    // 创建完毕状态
    created: function() {
      _this = this
    },
    // 组件更新
    updated: function() {
      // this.imgDownInfo()
      if (
        this.formData.businessLicense !== null
      ) {
        this.fillDownload(this.formData.businessLicense)
      }
    }
}
</script>

<style rel="stylesheet/scss" lang="scss">
</style>

<style rel="stylesheet/scss" lang="scss" scoped>
.fileImg{
  img{
    width:20%;
  }
}
</style>
```

（3）完成详情展示

在在`/src/module-saas-clients/pages/`下创建企业详情视图details.vue

```
<template>
  <div class="dashboard-container">
    <div class="app-container">
      <el-card shadow="never">
            <el-tabs v-model="activeName">
                <el-tab-pane label="企业信息" name="first">
                  <!--form表单
                      model ： 双向绑定的数据对象
```

```
                    -->
                    <el-form ref="form" :model="formData" label-width="200px">
                      <el-form-item label="企业名称" >
                        <el-input v-model="formData.name" style="width:250px" disabled>
</el-input>
                      </el-form-item>
                      <el-form-item label="公司地址">
                        <el-input v-model="formData.companyAddress"  style="width:250px"
 disabled></el-input>
                      </el-form-item>
                      <el-form-item label="公司电话">
                        <el-input v-model="formData.companyPhone"  style="width:250px"
 disabled></el-input>
                      </el-form-item>
                      <el-form-item label="邮箱">
                        <el-input v-model="formData.mailbox"  style="width:250px"
 disabled></el-input>
                      </el-form-item>
                      <el-form-item label="备注">
                        <el-input v-model="formData.remark"  style="width:250px" ></el-
input>
                      </el-form-item>
                      <el-form-item>
                        <el-button type="primary">审核</el-button>
                        <el-button>拒绝</el-button>
                      </el-form-item>
                    </el-form>
                  </el-tab-pane>
                  <el-tab-pane label="账户信息" name="second">账户信息</el-tab-pane>
                  <el-tab-pane label="交易记录" name="third">交易记录</el-tab-pane>
              </el-tabs>
      </el-card>
     </div>
  </div>
</template>

<script>
import {detail} from '@/api/base/saasClient'
export default {
  name: 'saas-clients-detail',
  data () {
    return {
        activeName: 'first',
        formData:{}
    }
  },
  methods: {
    detail(id) {
      detail({id:id}).then(res => {
        this.formData = res.data.data
        console.log(id)
        console.log(this.formData)
      })
```

```
    }
  },
  // 创建完毕状态
  created() {
    var id = this.$route.params.id
    this.detail(id);
  }
}
</script>

<style rel="stylesheet/scss" lang="scss" scoped>
.alert {
  margin: 10px 0px 0px 0px;
}
.pagination {
  margin-top: 10px;
  text-align: right;
}
</style>
```

4.4 接口对接

（1）启动第一天的企业微服务服务

（2）在`config/dev.env.js`中配置请求地址

```
'use strict'
const merge = require('webpack-merge')
const prodEnv = require('./prod.env')

module.exports = merge(prodEnv, {
  NODE_ENV: '"development"',
  BASE_API: '"http://localhost:9001/"'
})
```