Spring Boot开发小而完整的Web前后端分离项目实战

第01讲 课程介绍与项目演示

1.1、技术要点

1.1.1、前端知识：

vue element css3 html5

1.1.2、后端知识：

Spring Boot2.x 、Spring Security5.x、MyBatis Plus、Redis3.x

1.1.3、数据库:

MySql5.7

1.2、学习收货：

1.2.1、掌握Vue Element 开发后台页面的能力，从而深入理解Vue在后台管理系统中的开发流程；

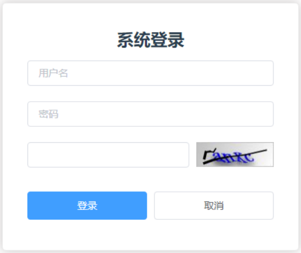
1.2.2、掌握运用Spring Boot开发后台接口的能力；

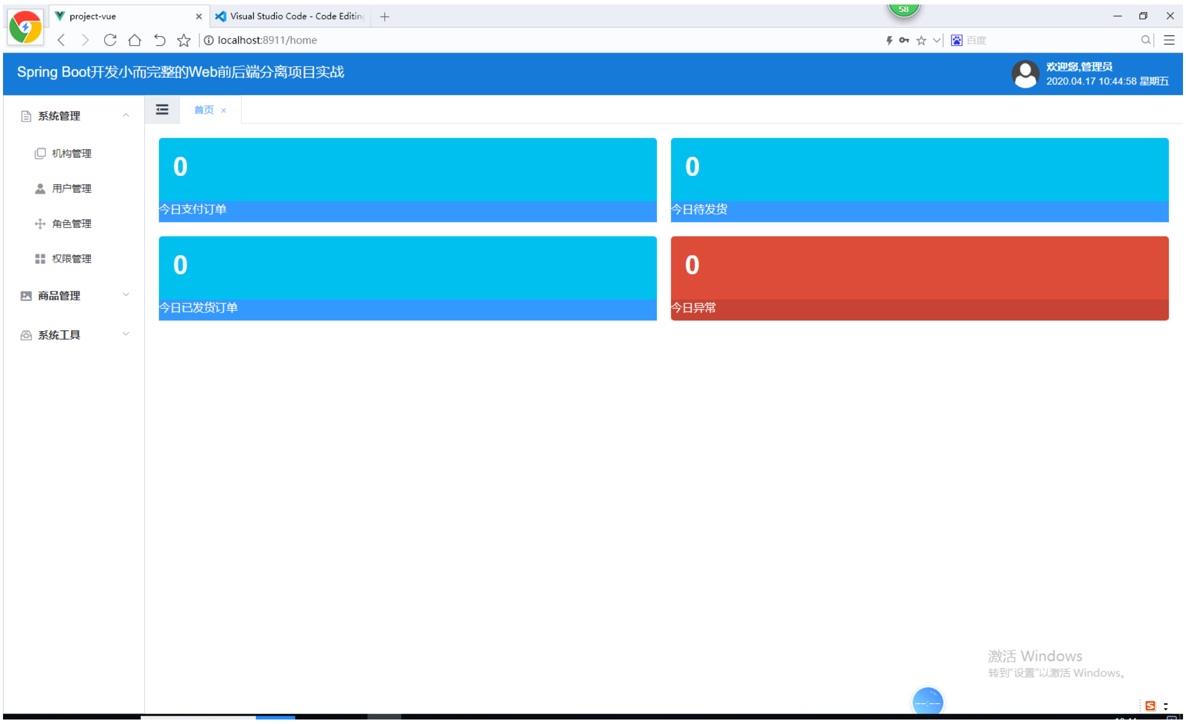
1.2.3、掌握Spring Security开发权限管理的能力；

1.2.4、掌握Redis缓存在开发中的运用能力；

1.2.5、最终学会用Vue Element Spring Boot 从零开始搭建小型前后端分离项目的能力，从而更深入的理解系统中整个数据的流向，从哪里来，到哪里去；

3、项目演示：





Spring Boot开发小而完整的Web前后端分离项目实战

欢迎您,管理员
2020.04.17 10:44:58 星期五

0 今日支付订单	0 今日待发货
0 今日已发货订单	0 今日异常

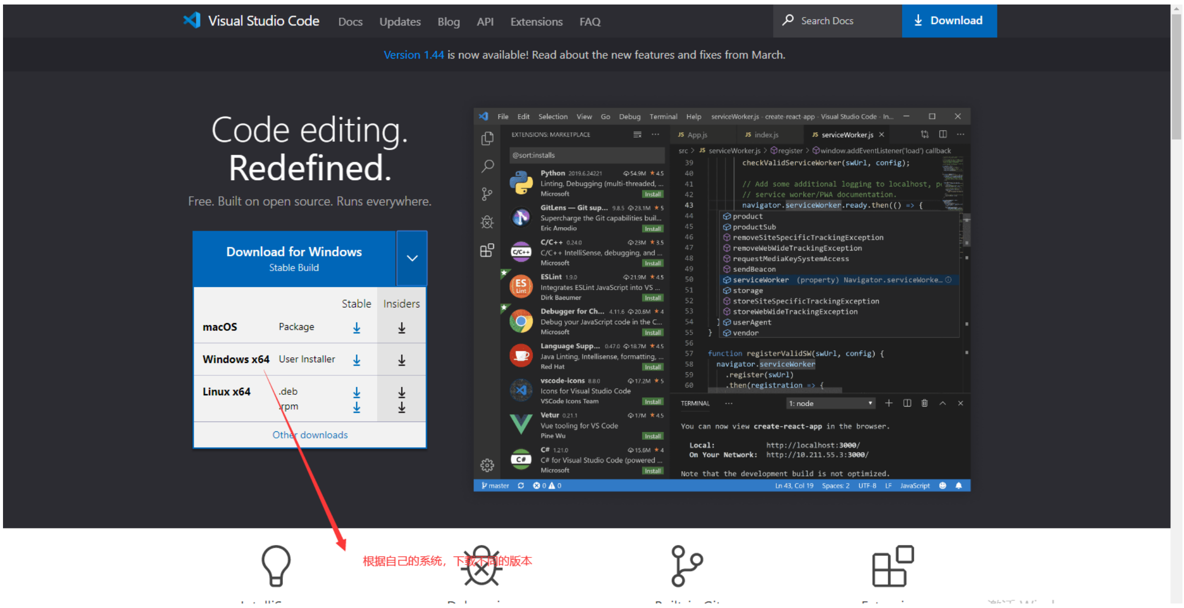
第02讲 前端项目工具安装及环境搭建

1.1、开发工具：visual studio code

1.1、visual studio code 官网下载地址

https://code.visualstudio.com/

1.2、打开官网，如下所示，根据自己电脑的系统，下载对应的vs code版本



Version 1.44 is now available! Read about the new features and fixes from March.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

		Stable	Insiders
macOS	Package	↓	↓
Windows x64	User Installer	↓	↓
Linux x64	.deb .rpm	↓ ↓	↓ ↓

Other downloads

```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

根据自己的系统，下载不同的版本

1.3、下载完成后，点击安装，一直 next，到最后安装完成即可；

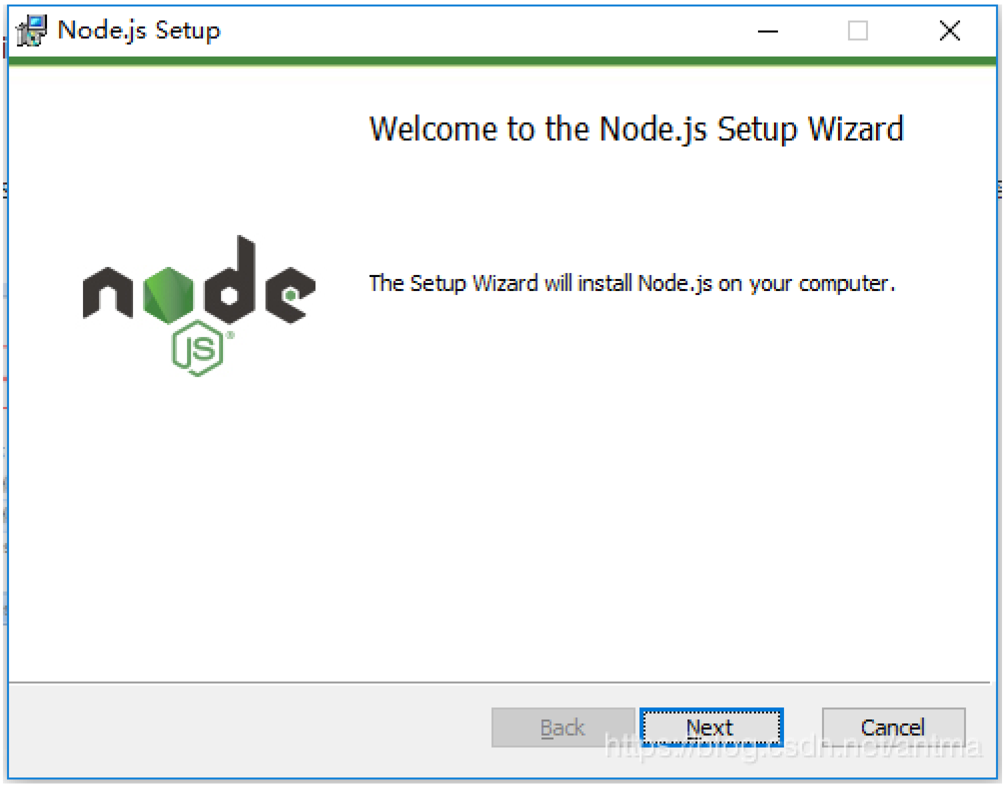
1.2、依赖环境

1.2.1、Node.js 安装

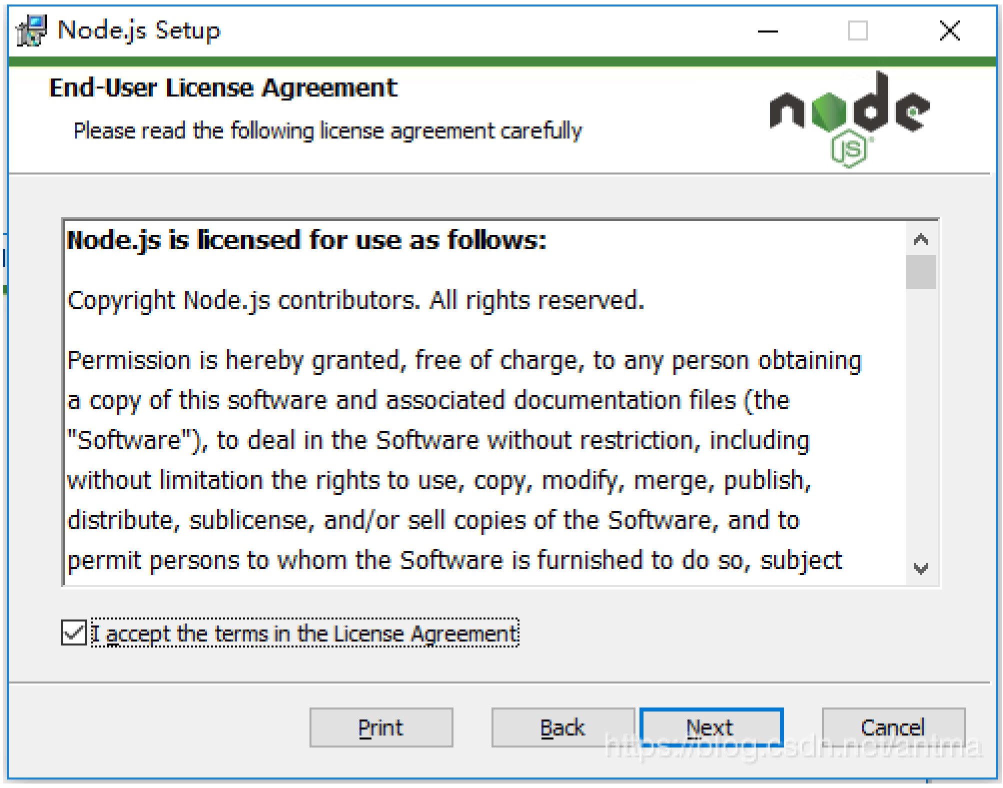
中文官网下载地址 **https://nodejs.org/zh-cn/download/**

1.2.2、到官网下载对应系统下的node.js安装，要求至少8.0以上的版本

1.2.3、https://nodejs.org/download/release/v10.16.0/ 下载 node-v10.16.0-x64.msi 版本，下载后一直next，直到安装完成



Node.js Setup
Welcome to the Node.js Setup Wizard
The Setup Wizard will install Node.js on your computer.
Back
Next
Cancel



Node.js Setup
End-User License Agreement
Please read the following license agreement carefully

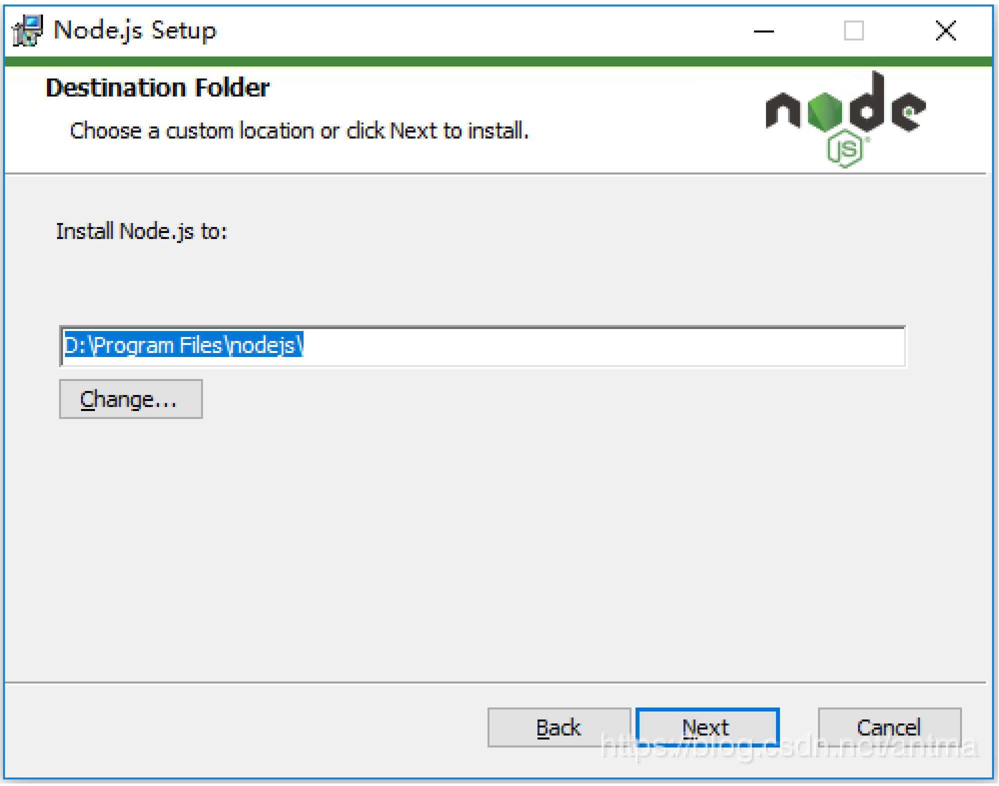
Node.js is licensed for use as follows:

Copyright Node.js contributors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject

☑ I accept the terms in the License Agreement

选择node.js的安装目录

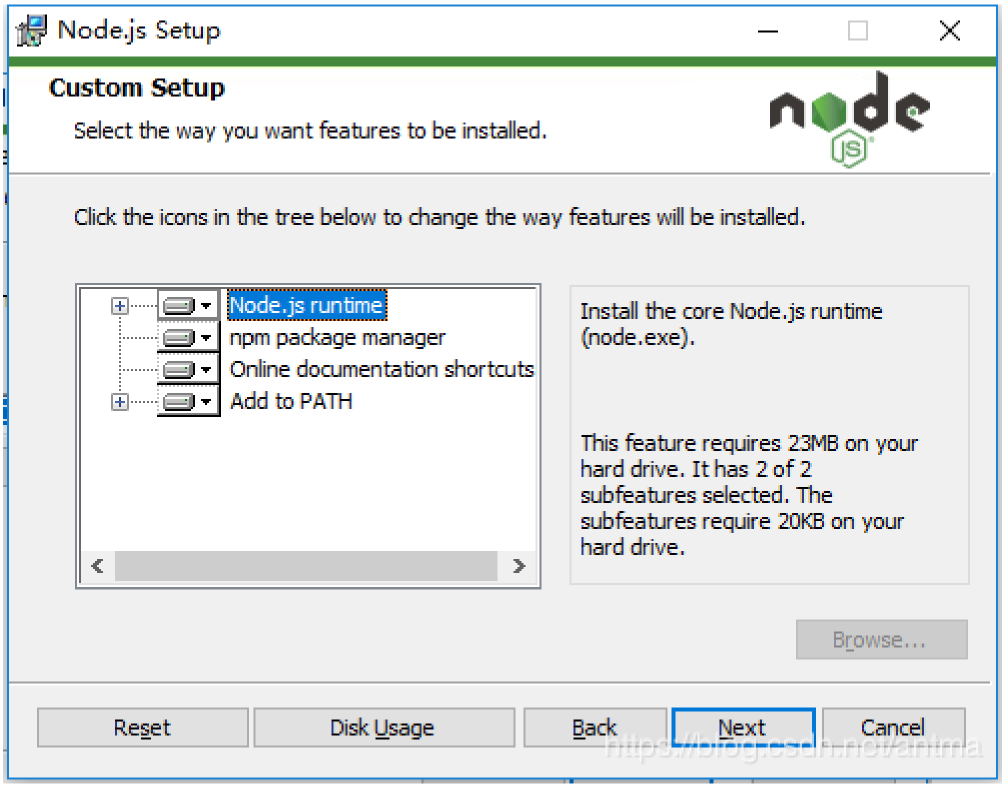


Destination Folder

Choose a custom location or click Next to install.

Install Node.js to:

D:\Program Files\nodejs\



Custom Setup

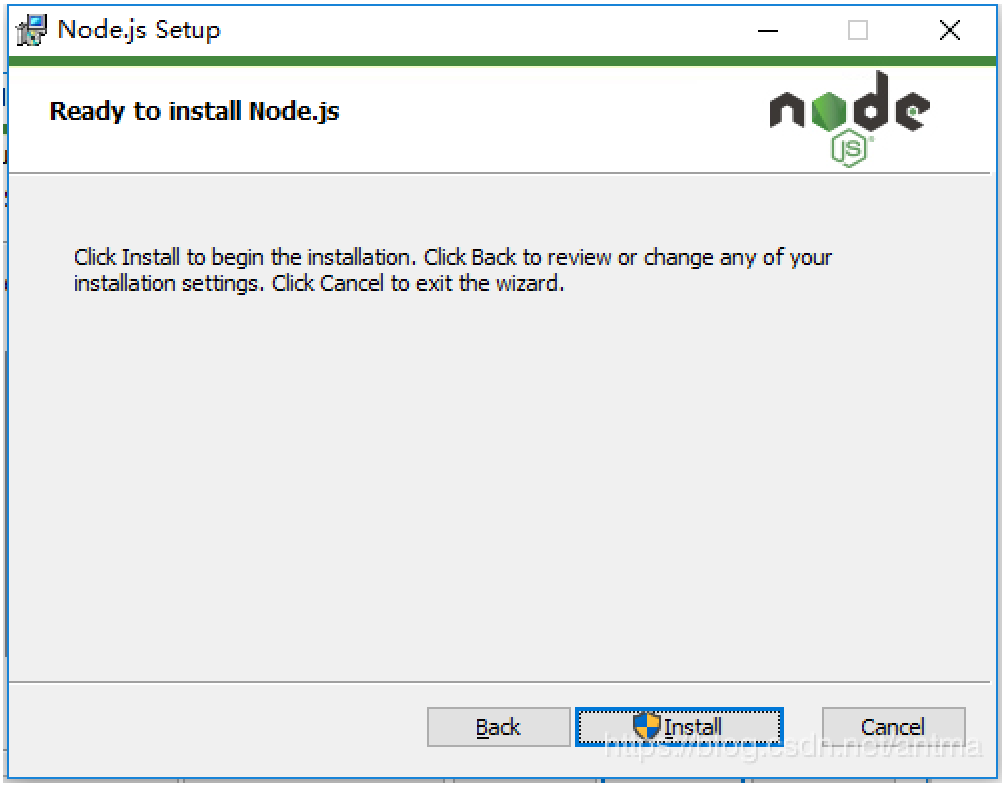
Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- Node.js runtime
- npm package manager
- Online documentation shortcuts
- Add to PATH

Install the core Node.js runtime (node.exe).

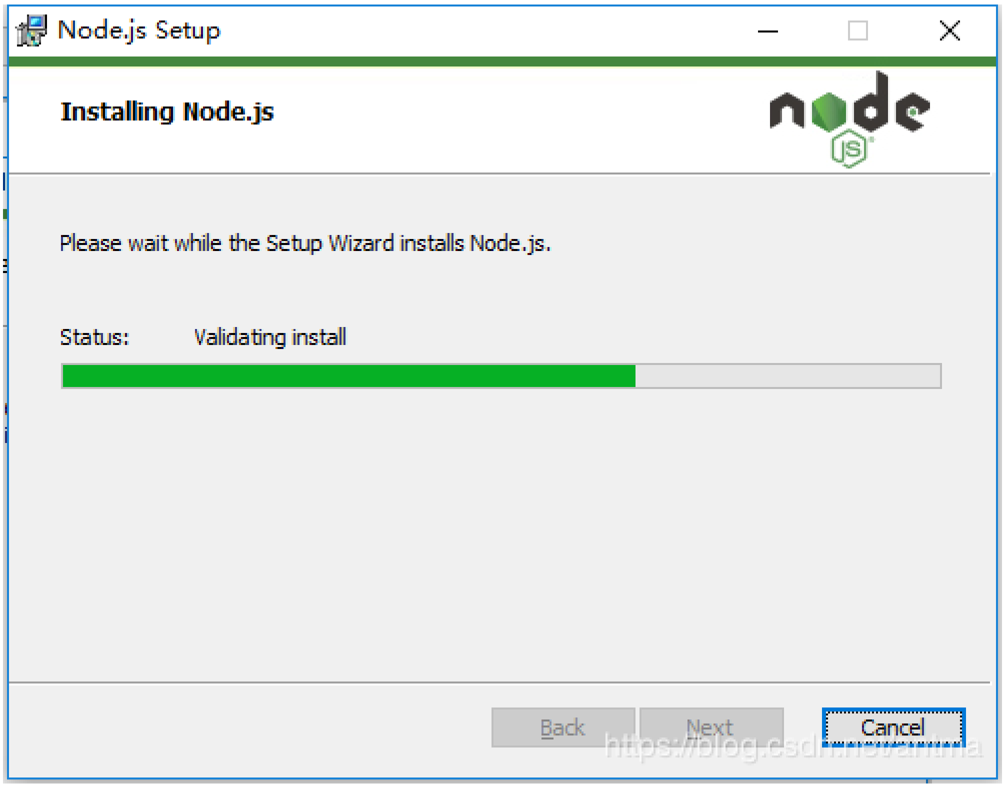
This feature requires 23MB on your hard drive. It has 2 of 2 subfeatures selected. The subfeatures require 20KB on your hard drive.



Node.js Setup

Ready to install Node.js

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

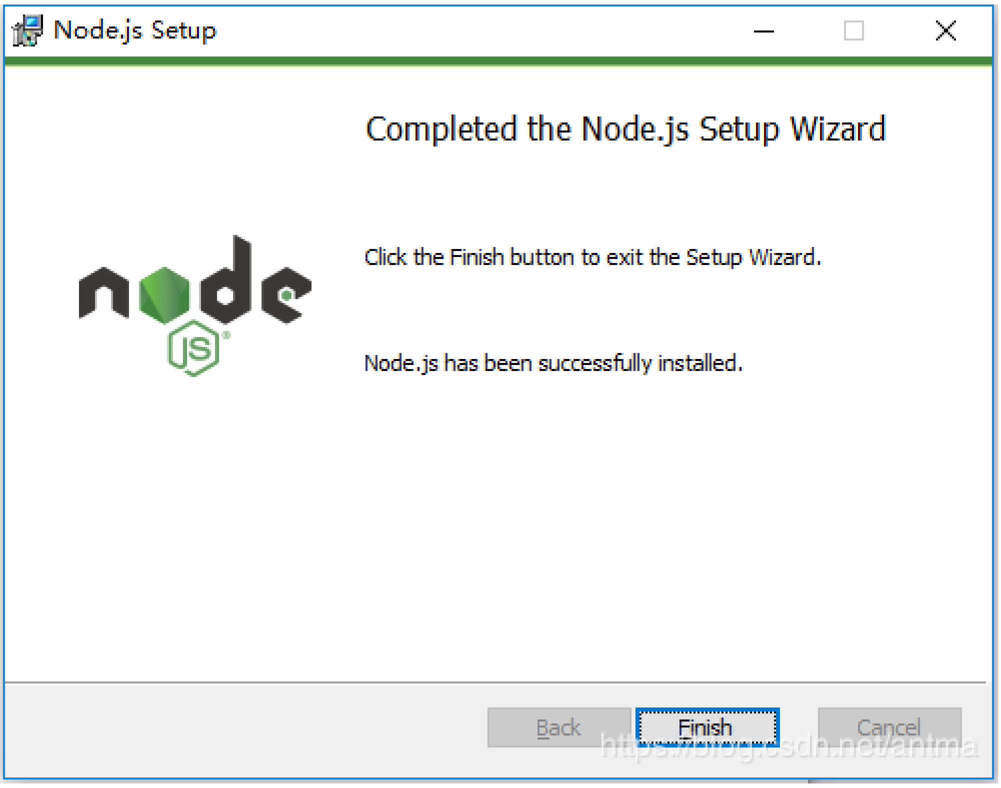


Node.js Setup

Installing Node.js

Please wait while the Setup Wizard installs Node.js.

Status: Validating install



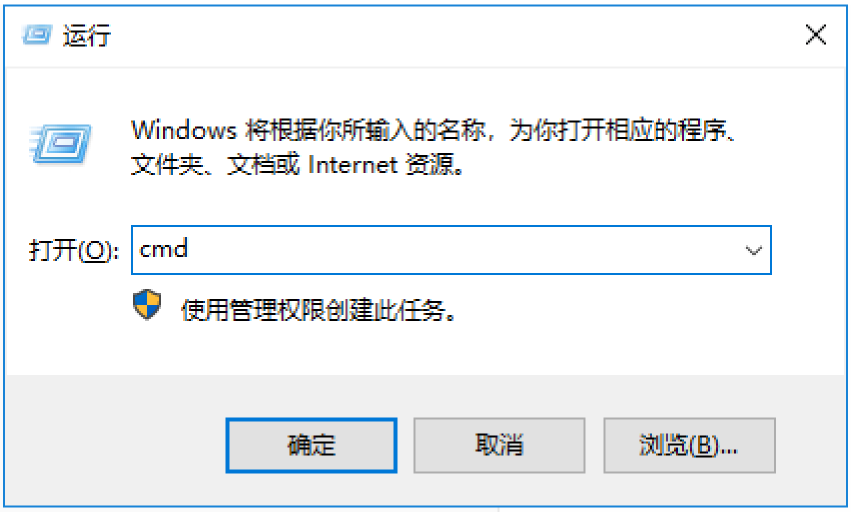
Node.js Setup

Completed the Node.js Setup Wizard

Click the Finish button to exit the Setup Wizard.

Node.js has been successfully installed.

1.2.4、 node -v 查看安装版本号，win10 系统 win+R组合键快速打开运行窗口，输入cmd，如下图



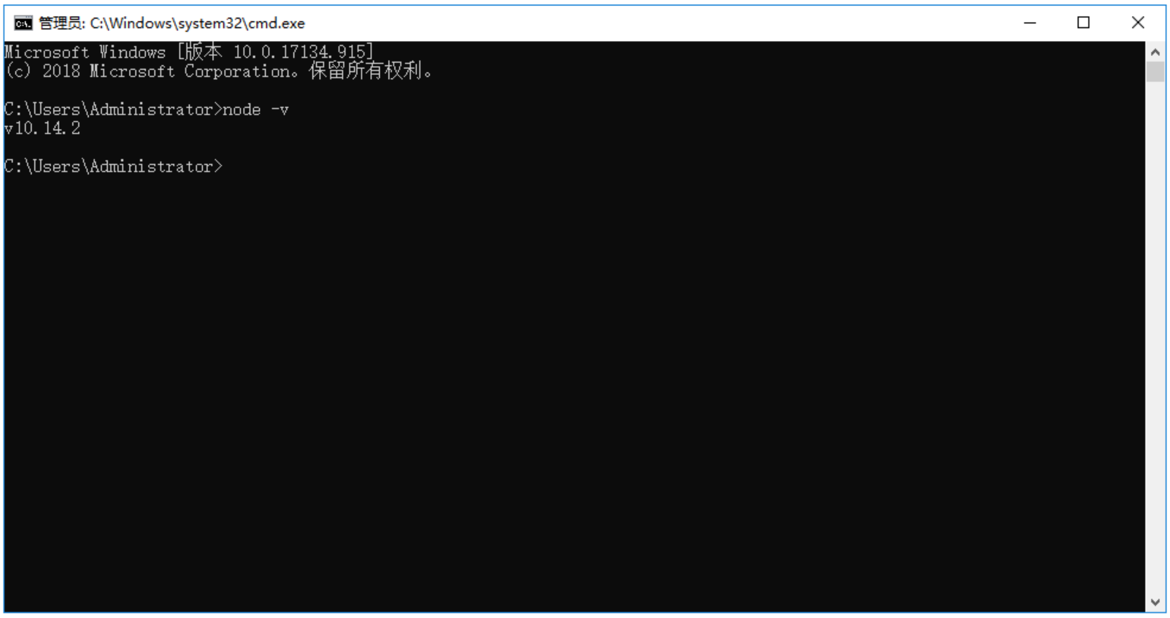
运行

Windows 将根据你所输入的名称，为你打开相应的程序、文件夹、文档或 Internet 资源。

打开(O): cmd

使用管理权限创建此任务。

1.2.5、确认进入命令行窗口 输入node -v

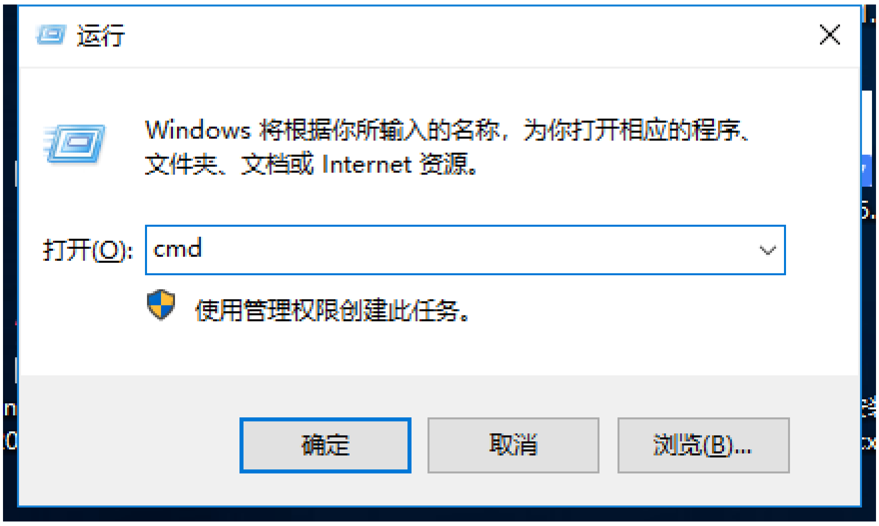


管理员: C:\Windows\system32\cmd.exe

```
Microsoft Windows [版本 10.0.17134.915]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>node -v
v10.14.2

C:\Users\Administrator>
```

1.3.、Vue CLI 脚手架

3.1安装命令　 npm i -g @vue/cli，　win10左下角右键，运行，输入cmd，进入命令行窗口 输入　npm i -g @vue/cli



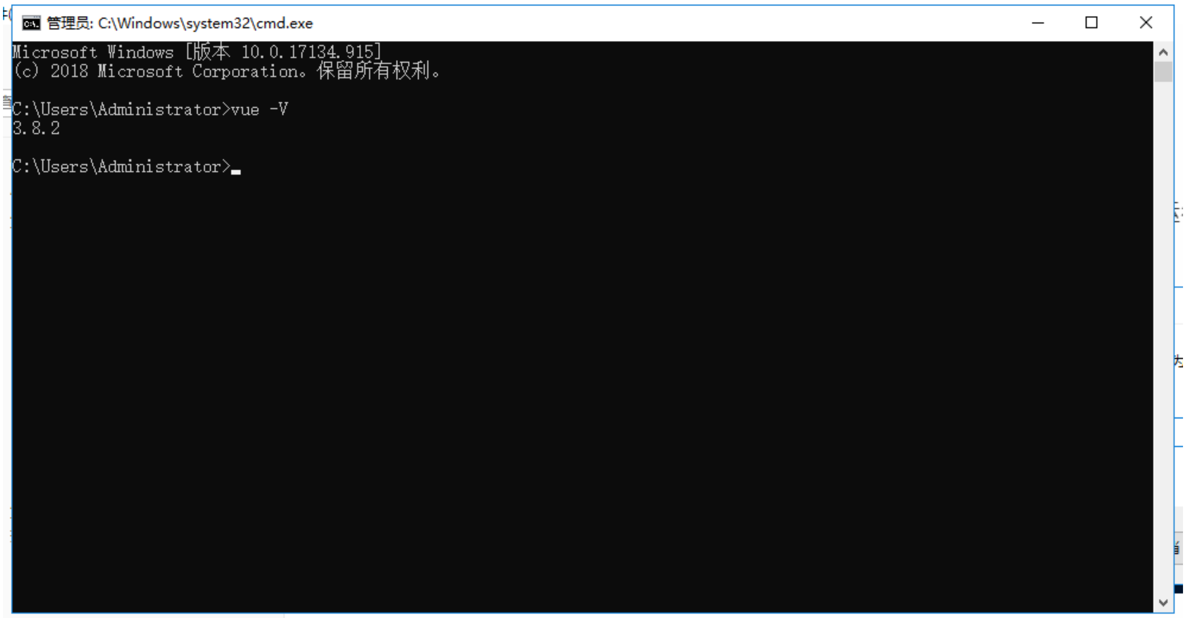
运行

Windows 将根据你所输入的名称，为你打开相应的程序、文件夹、文档或 Internet 资源。

打开(O): cmd

使用管理权限创建此任务。

3.2 输入 vue -V查看版本号



管理员: C:\Windows\system32\cmd.exe

```
Microsoft Windows [版本 10.0.17134.915]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>vue -V
3.8.2

C:\Users\Administrator>_
```

第03讲 前端Vue项目创建

1.1、vue cli 官网

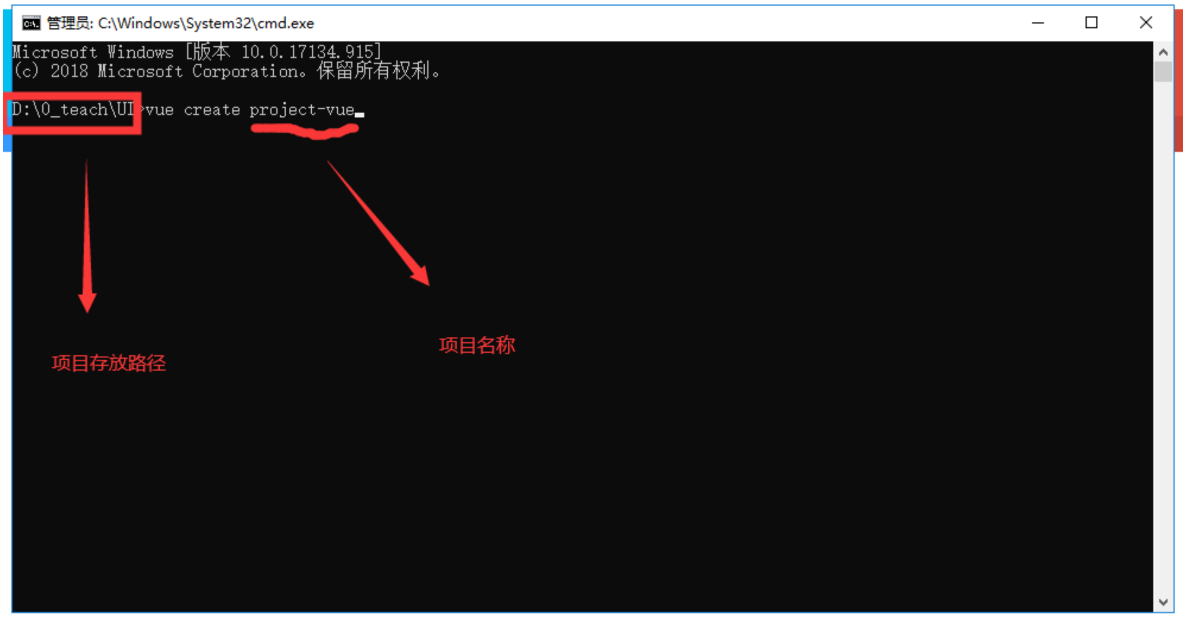
https://cli.vuejs.org/zh/guide/creating-a-project.html#vue-create

1.2、在自己电脑本地磁盘新建一个文件夹，用于保存项目， 文件夹命名最好使用英文名称，如下，

D:\0_teach\UI

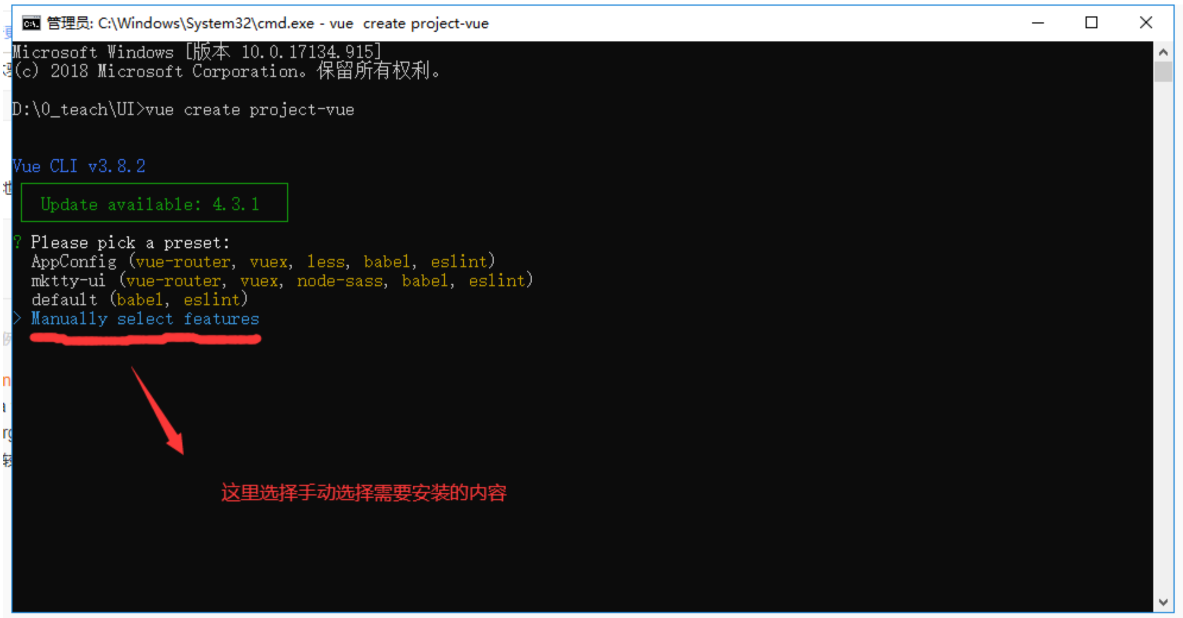
1.3、cd 进入文件夹，vue create 项目名称，输入 vue create project-vue 回车

1.3.1、 进入到项目保存的目录，输入 vue create project-vue 回车，



```
Microsoft Windows [版本 10.0.17134.915]
(c) 2018 Microsoft Corporation。保留所有权利。

D:\0_teach\UI>vue create project-vue
```

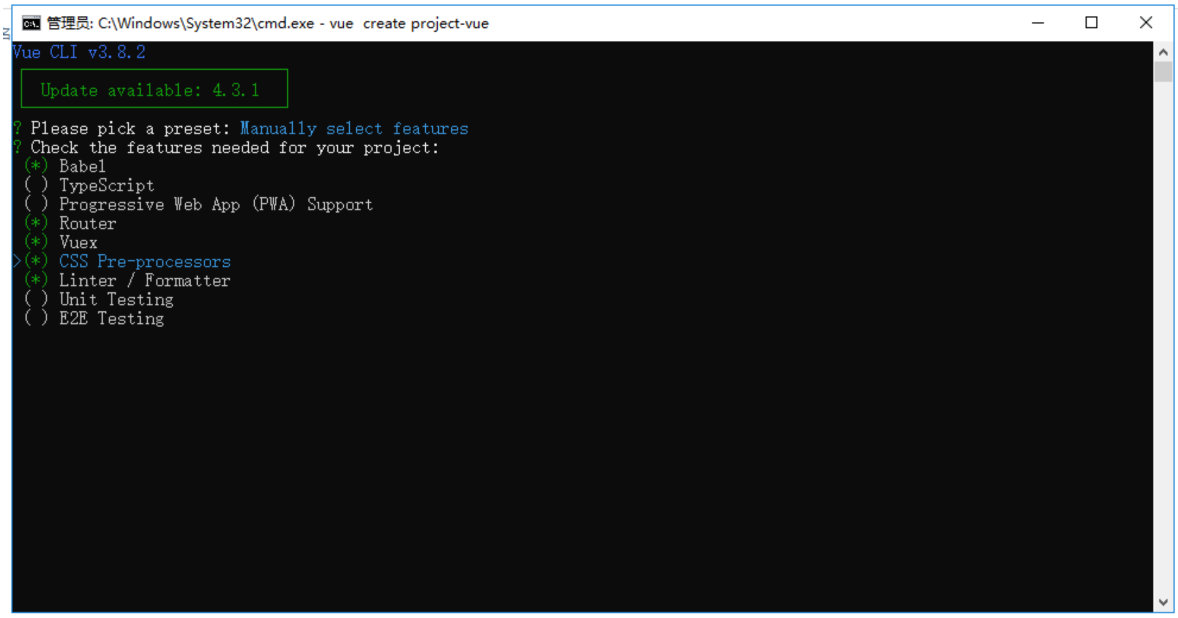
1.3.2、回车进入到如下界面，键盘上、下键选择需要的创建方式，我们这里选择 manually select features，手动选择配置的方式，然后 回车



管理员: C:\Windows\System32\cmd.exe - vue create project-vue

```
Microsoft Windows [版本 10.0.17134.915]
(c) 2018 Microsoft Corporation。保留所有权利。

D:\0_teach\UI>vue create project-vue


Vue CLI v3.8.2
 ┌─────────────────────────┐
 │  Update available: 4.3.1 │
 └─────────────────────────┘
? Please pick a preset:
  AppConfig (vue-router, vuex, less, babel, eslint)
  mktty-ui (vue-router, vuex, node-sass, babel, eslint)
  default (babel, eslint)
> Manually select features
```

1.3.3、回车进入到如下界面，键盘上、下键移动选择需要的项，按 空格 键可以选中。我们选择如下带*的项目，然后 回车


管理员: C:\Windows\System32\cmd.exe - vue create project-vue

```
Vue CLI v3.8.2
 ┌─────────────────────────┐
 │  Update available: 4.3.1  │
 └─────────────────────────┘
? Please pick a preset: Manually select features
? Check the features needed for your project:
 (*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 (*) Router
 (*) Vuex
>(*) CSS Pre-processors
 (*) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

选项说明

Babel：将ES6编译成ES5

TypeScript：使用TypeScript

Router和Vuex`：路由和状态管理

Linter/ Formatter：代码检查工具

CSS Pre-processors：css预编译

1.3.4、Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) y

路由使用历史模式? 这种模式充分利用 history.pushState API 来完成 URL 跳转而无须重新加载页面

1.3.5、 Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): 使用什么css预编译器?

选择 Sass/SCSS (with node-sass)

1.3.6、 Pick a linter / formatter config: 选择语法检测规范 选择 ESLint with error prevention only

eslint w...: 只进行报错提醒;

eslint + A...: 不严谨模式;

eslint + S...`: 正常模式;

eslint + P...`: 严格模式;

1.3.7、 Pick additional lint features：代码检查方式：

选择 Lint on save 保存时检查

1.3.8、 Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? (Use arrow keys)

选择配置信息存放位置，单独存放或者并入package.json

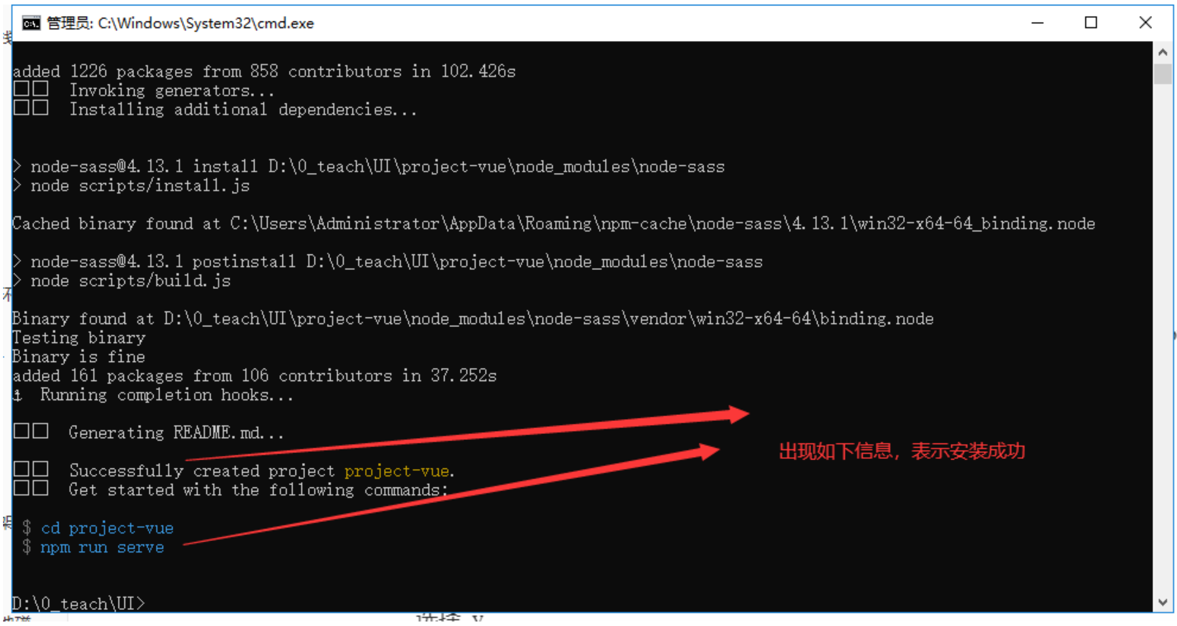
选择 In dedicated config files

1.3.9、 Save this as a preset for future projects? (y/N)

是否保存当前预设，下次构建无需再次配置

选择 n

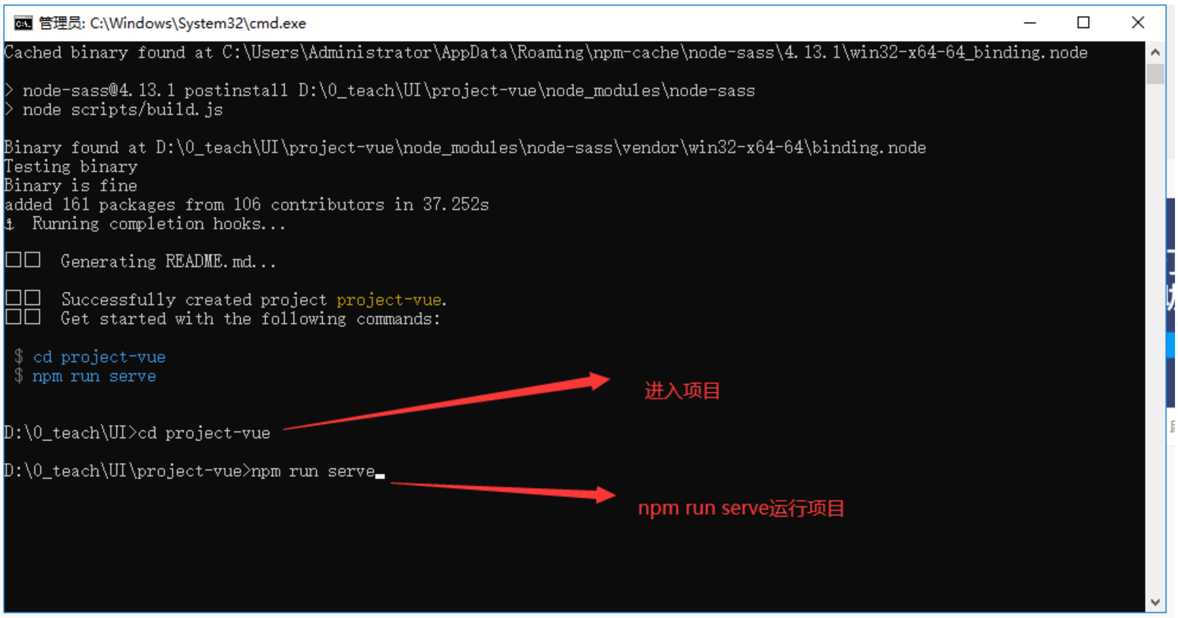
1.3.10、 回车，等待下载依赖



```
added 1226 packages from 858 contributors in 102.426s
□□  Invoking generators...
□□  Installing additional dependencies...


> node-sass@4.13.1 install D:\0_teach\UI\project-vue\node_modules\node-sass
> node scripts/install.js

Cached binary found at C:\Users\Administrator\AppData\Roaming\npm-cache\node-sass\4.13.1\win32-x64-64_binding.node

> node-sass@4.13.1 postinstall D:\0_teach\UI\project-vue\node_modules\node-sass
> node scripts/build.js

Binary found at D:\0_teach\UI\project-vue\node_modules\node-sass\vendor\win32-x64-64\binding.node
Testing binary
Binary is fine
added 161 packages from 106 contributors in 37.252s
⚓  Running completion hooks...

□□  Generating README.md...

□□  Successfully created project project-vue.
□□  Get started with the following commands:

 $ cd project-vue
 $ npm run serve


D:\0_teach\UI>
```

出现如下信息，表示安装成功

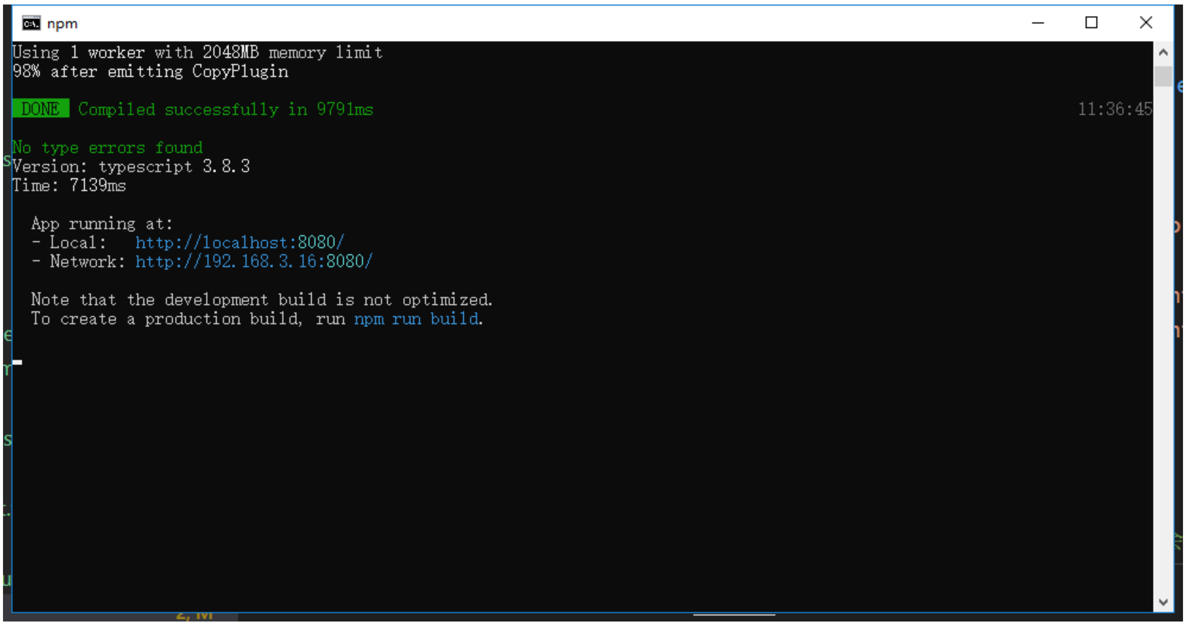
1.3.11、cd project-vue 进入到项目， npm run serve运行项目



```
Cached binary found at C:\Users\Administrator\AppData\Roaming\npm-cache\node-sass\4.13.1\win32-x64-64_binding.node

> node-sass@4.13.1 postinstall D:\0_teach\UI\project-vue\node_modules\node-sass
> node scripts/build.js

Binary found at D:\0_teach\UI\project-vue\node_modules\node-sass\vendor\win32-x64-64\binding.node
Testing binary
Binary is fine
added 161 packages from 106 contributors in 37.252s
⚓  Running completion hooks...

□□  Generating README.md...

□□  Successfully created project project-vue.
□□  Get started with the following commands:

 $ cd project-vue
 $ npm run serve


D:\0_teach\UI>cd project-vue

D:\0_teach\UI\project-vue>npm run serve_
```

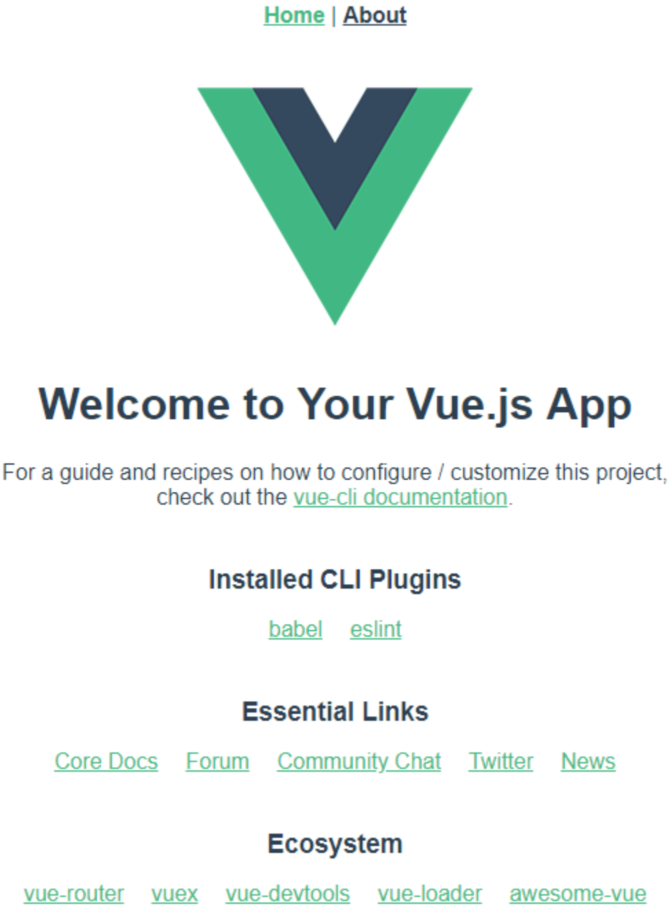
进入项目

npm run serve运行项目



```
Using 1 worker with 2048MB memory limit
98% after emitting CopyPlugin

 DONE  Compiled successfully in 9791ms                                                                       11:36:45

No type errors found
Version: typescript 3.8.3
Time: 7139ms

  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.3.16:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

浏览器 http://localhost:8080访问项目，看到如下页面表示项目创建成功



Home | About

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

Essential Links

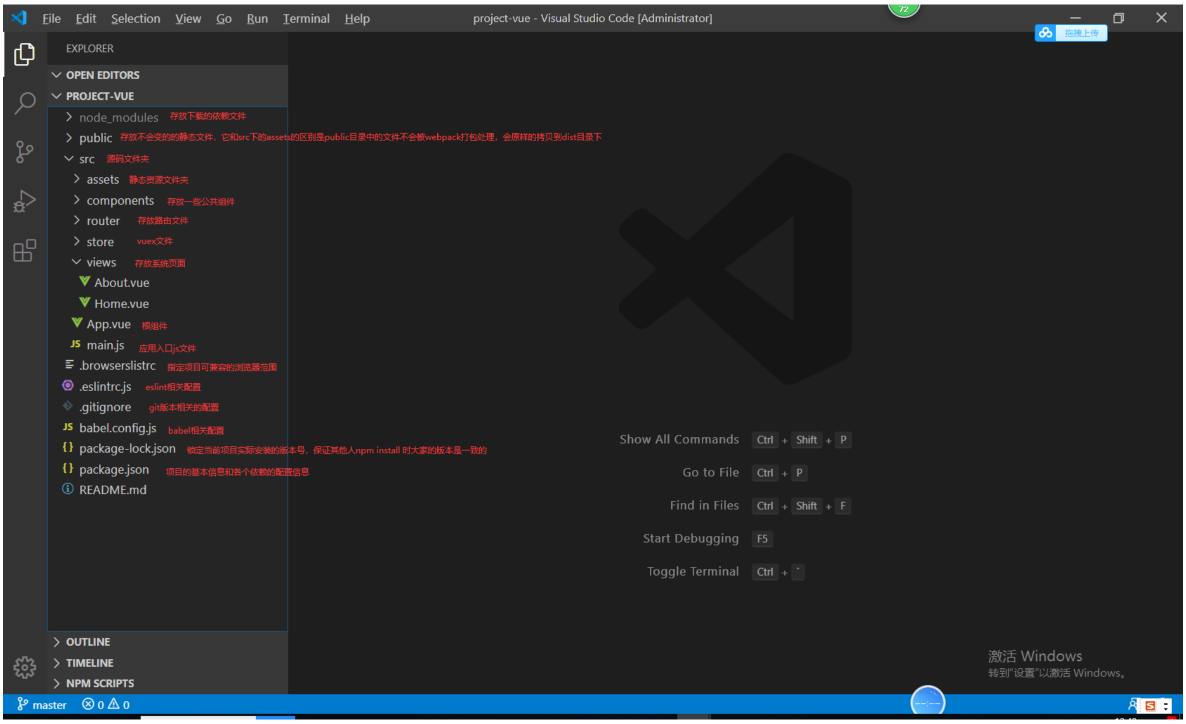
Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

第04讲 项目目录介绍与Element依赖引入

1.1、vue项目目录介绍

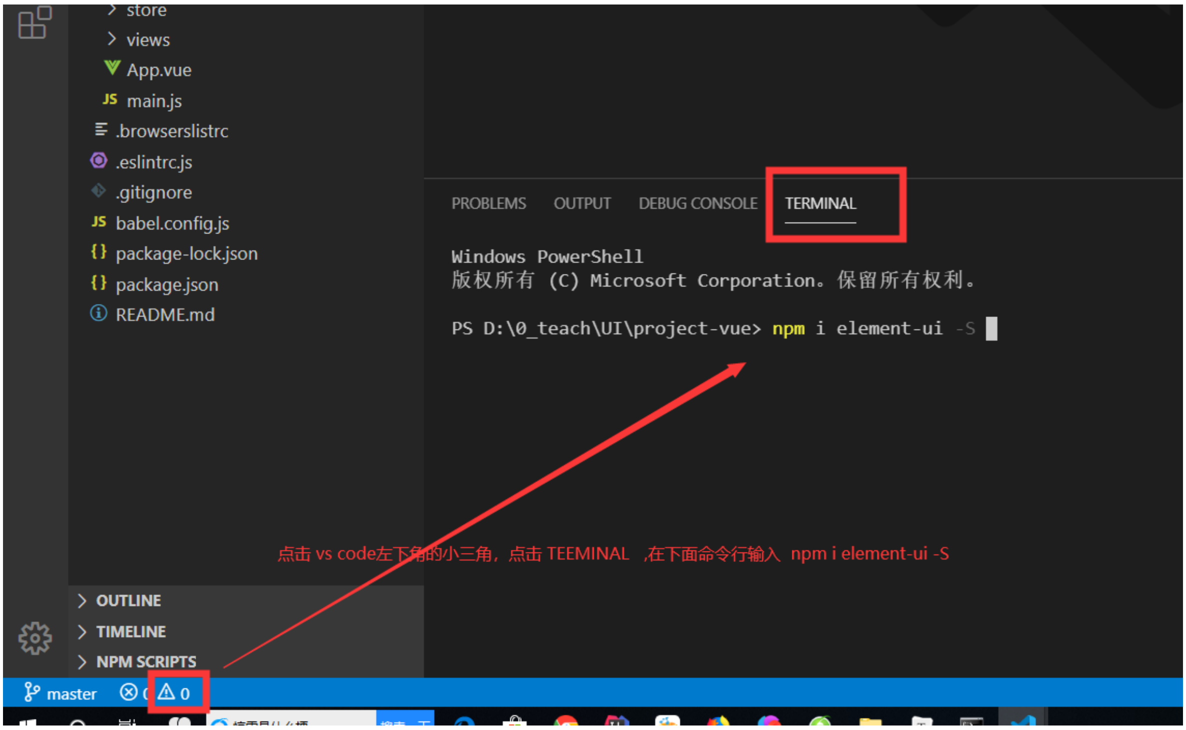

File Edit Selection View Go Run Terminal Help
project-vue - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS
PROJECT-VUE
OUTLINE
TIMELINE
NPM SCRIPTS
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Start Debugging F5
Toggle Terminal Ctrl + `
激活 Windows
转到"设置"以激活 Windows。
master ⊗ 0 ⚠ 0

```
> node_modules   存放下载的依赖文件
> public   存放不会变的的静态文件，它和src下的assets的区别是public目录中的文件不会被webpack打包处理，会原样的拷贝到dist目录下
∨ src   源码文件夹
  > assets   静态资源文件夹
  > components   存放一些公共组件
  > router   存放路由文件
  > store   vuex文件
  ∨ views   存放系统页面
      About.vue
      Home.vue
    App.vue   根组件
    main.js   应用入口js文件
  .browserslistrc   指定项目可兼容的浏览器范围
  .eslintrc.js   eslint相关配置
  .gitignore   git版本相关的配置
  babel.config.js   babel相关配置
  package-lock.json   锁定当前项目实际安装的版本号，保证其他人npm install 时大家的版本是一致的
  package.json   项目的基本信息和各个依赖的配置信息
  README.md
```

1.2、element安装

1.2.1、element UI 官网　https://element.eleme.cn/#/zh-CN/component/installation

1.2.2、安装element　 npm i element-ui -S



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

PS D:\0_teach\UI\project-vue> npm i element-ui -S
```

点击 vs code左下角的小三角，点击 TEEMINAL ，在下面命令行输入 npm i element-ui -S

1.3、element引入　https://element.eleme.cn/#/zh-CN/component/quickstart

1.3.1、在项目中main.js中引入

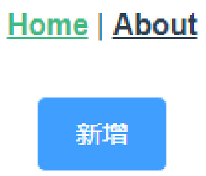
import ElementUI from 'element-ui';　//引入element组件库

import 'element-ui/lib/theme-chalk/index.css';　//引入element样式文件

Vue.use(ElementUI);　//使用element

1.4、测试element是否引入成功

在项目的页面 输入 新增，启动项目，如下效果说明element引入成功



1.3、cs code插件安装

Vetur 这个插件是 vscode 能优雅写 Vue 的核心，代码高亮，语法检查等

Vue VSCode Snippets 代码补全

Element UI VSCode Snippets vscode-element-helper element的代码提示

1.4、解决css属性在html标签里面不提示问题

1.4.1、设置 -> 搜索`prevent` -> 把`Snippets Prevent Quick Suggestions` 勾掉即可

1.4.2、如果是vue页面，需要把vs code右下角的页面模式选择为 html 即可

第05讲 css3弹性盒子基础讲解

盒子模型特点:

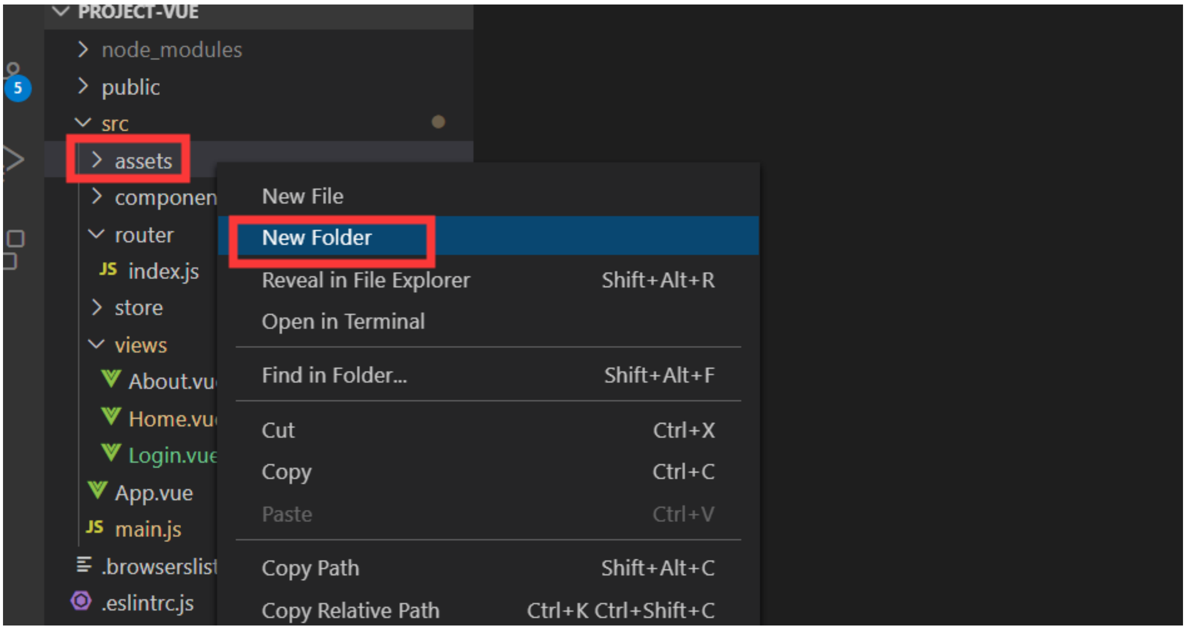
1.div默认是从上到下排列的

2.当把一个div变成一个盒子模型的时候，div子元素将横向排列

3.盒子模型默认存在两个轴，主轴(x轴，水平方向)和交叉轴(y轴，垂直方向)

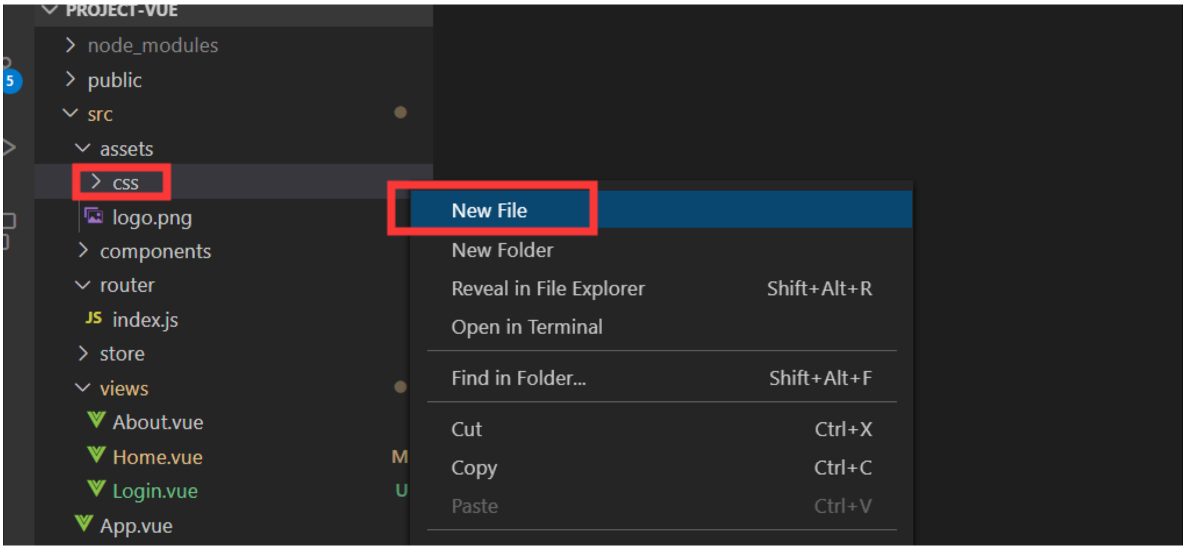
1.1、在assets目录下新建css目录

1.1.1、找到项目assets目录，右键->New Folder -> 输入 css 按回车



1.1.2、新建flex.css

找到上面新建的css目录，右键 -> New File 录入 flex.css 按回车



1.1.3、录入如下css

```
/*!css公共样式
弹性盒子常用布局
*/
.ub{   //把一个div变成一个盒子模型
    display: -webkit-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex
}

.row-left {  //设置div横向排列从左
    -webkit-box-orient: horizontal;
    -webkit-box-direction: normal;
    -webkit-flex-direction: row;
    -ms-flex-direction: row;
    flex-direction: row
}

.row-right {   //设置div横向从右边排列
```

```
    -webkit-box-orient: horizontal;
    -webkit-box-direction: reverse;
    -webkit-flex-direction: row-reverse;
    -ms-flex-direction: row-reverse;
    flex-direction: row-reverse;
    -webkit-box-pack: end
}

.column-top {  //设置div从上到下排列
    -webkit-box-orient: vertical;
    -webkit-box-direction: normal;
    -webkit-flex-direction: column;
    -ms-flex-direction: column;
    flex-direction: column
}

.column-bottom { //设置div从下到上排列
    -webkit-box-orient: vertical;
    -webkit-box-direction: reverse;
    -webkit-flex-direction: column-reverse;
    -ms-flex-direction: column-reverse;
    flex-direction: column-reverse;
    -webkit-box-pack: end
}

.main-left {
    -webkit-box-pack: start;
    -webkit-justify-content: flex-start;
    -ms-flex-pack: start;
    justify-content: flex-start
}

.main-right {
    -webkit-box-pack: end;
    -webkit-justify-content: flex-end;
    -ms-flex-pack: end;
    justify-content: flex-end
}

.main-justify {
    -webkit-box-pack: justify;
    -webkit-justify-content: space-between;
    -ms-flex-pack: justify;
    justify-content: space-between
}

.main-center {
    -webkit-box-pack: center;
    -webkit-justify-content: center;
    -ms-flex-pack: center;
    justify-content: center
}

.cross-top {
    -webkit-box-align: start;
    -webkit-align-items: flex-start;
    -ms-flex-align: start;
    align-items: flex-start
}

.cross-bottom {
    -webkit-box-align: end;
    -webkit-align-items: flex-end;
    -ms-flex-align: end;
    align-items: flex-end
}

.cross-center {
    -webkit-box-align: center;
    -webkit-align-items: center;
    -ms-flex-align: center;
    align-items: center
}

.ub-f0 {
    -webkit-box-flex: 0;
    -webkit-flex-grow: 0;
    -ms-flex-positive: 0;
    flex-grow: 0;
    -webkit-flex-shrink: 0;
    -ms-flex-negative: 0;
    flex-shrink: 0
}

.ub-f1 {
    -webkit-box-flex: 1;
    -webkit-flex-grow: 1;
    -ms-flex-positive: 1;
    flex-grow: 1;
    -webkit-flex-shrink: 1;
    -ms-flex-negative: 1;
    flex-shrink: 1
}

.ub-f2 {
    -webkit-box-flex: 2;
    -webkit-flex-grow: 2;
    -ms-flex-positive: 2;
    flex-grow: 2;
    -webkit-flex-shrink: 2;
    -ms-flex-negative: 2;
    flex-shrink: 2
}

.ub-f3 {
    -webkit-box-flex: 3;
    -webkit-flex-grow: 3;
    -ms-flex-positive: 3;
    flex-grow: 3;
    -webkit-flex-shrink: 3;
    -ms-flex-negative: 3;
    flex-shrink: 3
}

.ub-f4 {
    -webkit-box-flex: 4;
    -webkit-flex-grow: 4;
    -ms-flex-positive: 4;
    flex-grow: 4;
    -webkit-flex-shrink: 4;
    -ms-flex-negative: 4;
    flex-shrink: 4
```

```
}

.ub-f5 {
    -webkit-box-flex: 5;
    -webkit-flex-grow: 5;
    -ms-flex-positive: 5;
    flex-grow: 5;
    -webkit-flex-shrink: 5;
    -ms-flex-negative: 5;
    flex-shrink: 5
}

.ub-f6 {
    -webkit-box-flex: 6;
    -webkit-flex-grow: 6;
    -ms-flex-positive: 6;
    flex-grow: 6;
    -webkit-flex-shrink: 6;
    -ms-flex-negative: 6;
    flex-shrink: 6
}

.ub-f7 {
    -webkit-box-flex: 7;
    -webkit-flex-grow: 7;
    -ms-flex-positive: 7;
    flex-grow: 7;
    -webkit-flex-shrink: 7;
    -ms-flex-negative: 7;
    flex-shrink: 7
}

.ub-f8 {
    -webkit-box-flex: 8;
    -webkit-flex-grow: 8;
    -ms-flex-positive: 8;
    flex-grow: 8;
    -webkit-flex-shrink: 8;
    -ms-flex-negative: 8;
    flex-shrink: 8
}

.ub-f9 {
    -webkit-box-flex: 9;
    -webkit-flex-grow: 9;
    -ms-flex-positive: 9;
    flex-grow: 9;
    -webkit-flex-shrink: 9;
    -ms-flex-negative: 9;
    flex-shrink: 9
}

.ub-f10 {
    -webkit-box-flex: 10;
    -webkit-flex-grow: 10;
    -ms-flex-positive: 10;
    flex-grow: 10;
    -webkit-flex-shrink: 10;
    -ms-flex-negative: 10;
    flex-shrink: 10
}
```

1.1.4、在项目main.js中引入flex.css

1.1.5、弹性盒子模型常用属性讲解 https://www.cnblogs.com/qcloud1001/p/9848619.html

1.1.5.1、flex-direction 决定主轴的方向

row（默认值）：主轴为水平方向，起点在左端。

row-reverse：主轴为水平方向，起点在右端。

column：主轴为垂直方向，起点在上沿。

column-reverse：主轴为垂直方向，起点在下沿。

1.1.5.2、justify-content主轴上的对齐方式。

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

1.1.5.3、align-items属性定义在交叉轴上的对齐方式

flex-start：交叉轴的起点对齐。

flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

1.1.5.4、flex-grow属性定义div所占的份数

代码示例：

```
<div>
    <h3>div默认从上到下排列</h3>
    <div style="background-color:#d0d0d0;height:250px;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型默认从左到右排列</h3>
    <div style="height: 250px;display: flex;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型从右到左排列排列</h3>
    <div style="height: 250px;display: flex;flex-direction:row-reverse;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型右对齐方式</h3>
    <div style="height: 250px;display: flex;justify-content:flex-end;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型中间对齐方式</h3>
    <div style="height: 250px;display: flex;justify-content:center;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型左对齐方式</h3>
    <div style="height: 250px;display: flex;justify-content:flex-start;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型两边对齐方式</h3>
    <div
      style="height: 250px;display: flex;justify-content:space-between;background-color:#d0d0d0;"
    >
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>

    <h3>盒子模型两边对齐方式</h3>
    <div style="height: 250px;display: flex;justify-content:space-around;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>

    <h3>盒子模型交叉轴起点对齐</h3>
    <div style="height: 250px;display: flex;align-items:flex-start;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型交叉轴居中对齐</h3>
    <div style="height: 250px;display: flex;align-items:center;background-color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型交叉轴尾部对齐</h3>
    <div style="height: 250px;display: flex;align-items:flex-end;background-
color:#d0d0d0;">
      <div style="background-color: red;height:50px;width:50px;">1</div>
      <div style="background-color: blue;height:50px;width:50px;">2</div>
      <div style="background-color: #ff7670;height:50px;width:50px;">3</div>
      <div style="background-color: yellow;height:50px;width:50px;">4</div>
    </div>
    <h3>盒子模型平分</h3>
    <div style="height: 50px;display: flex;">
      <div style="background-color: red;flex-grow:1">1</div>
      <div style="background-color: blue;flex-grow:1">2</div>
      <div style="background-color: #ff7670;flex-grow:1">3</div>
      <div style="background-color: yellow;flex-grow:1">4</div>
    </div>
  </div>
```

第06讲 登录页面布局

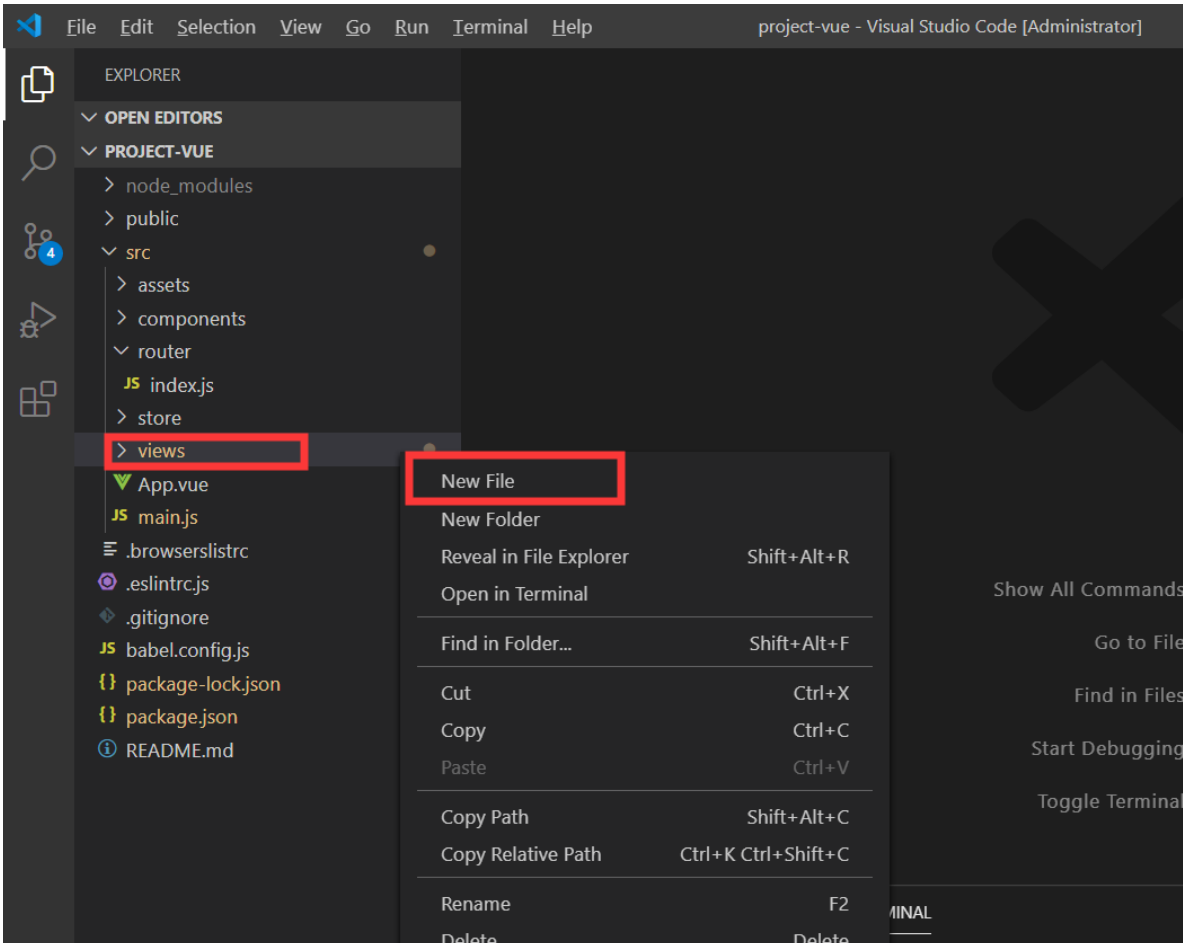
1.1、设置public下面index.html 高度100% ,内边距、外边距为 0

```
html,body{
  height: 100%;
  margin: 0;
  padding: 0;
}
```

1.2、设置home.vue 、app.vue高度为 100%

2.1、在views目录下新建Login.vue页面

2.1.1、找到项目目录views，右键->New File-> 输入 Login.vue   回车



2.2.2、实现代码

1、用到组件 el-form 、el-form-item、el-input、el-row

2、页面100%高度

3、表单

宽高：350 px 300px

表单阴影 box-shadow: 0 0 25px #cac6c6;

内边距 20px 35px

圆角：10px

el-form-item：左边外边距 0

按钮宽度：100%

2.2.3、最终代码

```
<template>
  <div class="login-container ub main-center cross-center">
    <el-form size="medium" class="login-form" :model="loginForm" ref="loginForm"
label-width="80px">
      <el-form-item label>
        <div class="ub main-center cross-center login-title">系统登录</div>
      </el-form-item>
      <el-form-item label>
        <el-input v-model="loginForm.username" placeholder="输入用户名"></el-input>
      </el-form-item>
      <el-form-item label>
        <el-input v-model="loginForm.password" placeholder="输入密码"></el-input>
      </el-form-item>
      <el-form-item label>
        <el-row :gutter="10">
          <el-col :span="16">
            <el-input placeholder="请输入验证码"></el-input>
          </el-col>
          <el-col :span="8">
            <el-input
              readonly
              v-model="loginForm.captcha"
              auto-complete="off"
              placeholder="单击图片刷新"
              style="width: 100%;"
            ></el-input>
          </el-col>
        </el-row>
      </el-form-item>
      <el-form-item>
        <el-row :gutter="20">
          <el-col :span="12">
            <el-button class="my-button" type="primary" @click="onSubmit">登录</el-button>
          </el-col>
          <el-col :span="12">
            <el-button class="my-button">重置</el-button>
          </el-col>
        </el-row>
      </el-form-item>
    </el-form>
  </div>
</template>

<script>
export default {
  data() {
    return {
      loginForm: {
        username: "",
        password: ""
      }
    };
  },
  methods: {
    //登录表单提交
    onSubmit() {}
  }
};
</script>

<style lang="css" scoped>
.login-title {
  font-size: 24px;
  font-weight: 600;
}
.login-container {
  height: 100%;
}
.login-form {
  height: 300px;
  width: 350px;
  border-radius: 10px;
  box-shadow: 0 0 25px #cac6c6;
  padding: 20px 35px;
}
.login-container /deep/ .el-form-item__content {
  margin-left: 0 !important;
}
.my-button {
  width: 100%;
}
</style>
```

第07讲 登录表单验证

1.1、表单非空验证

在不输入用户号码、密码、验证码的情况下，不能提交表单

1.2、验证规则

```
     prop，ref，model ，rules  这几个属性一定要添加，否则校验不生效，以及对应的值 对应

     <!--     ref  表单被引用时的名称，标识 this.$refs.shop.validate() 与这个对应-->
     <!--     model 表单数据对象 和data中shop对应-->
     <!--     rules 表单校验规则，和data中保持一致 submitRules-->
     <el-form ref="loginForm" :model="loginForm" :rules="submitRules" label-
width="120px">
     <!-- prop：表单域 model 字段，要和data中保持一致，在使用 validate、resetFields 方法的
情况下，该属性是必填的-->
       <el-form-item prop="username" label>
        <el-input v-model="loginForm.username" placeholder="输入用户名"></el-input>
      </el-form-item>
      <el-form-item prop="password" label>
        <el-input v-model="loginForm.password" placeholder="输入密码"></el-input>
      </el-form-item>
      <el-form-item prop="code" label>
        <el-row :gutter="10">
          <el-col :span="16">
            <el-input v-model="loginForm.code" placeholder="请输入验证码"></el-input>
          </el-col>
          <el-col :span="8">
            <el-input readonly auto-complete="off" placeholder="单击图片刷新"
style="width: 100%;"></el-input>
          </el-col>
        </el-row>
      </el-form-item>
     </el-form>

     // 检验规则
       submitRules: {
         username: [
          {
            required: true,
            trigger: "change",
            message: "请输入用户名"
          }
        ],
        password: [
          {
            required: true,
            trigger: "change",
            message: "请输入密码"
          }
        ],
        code: [
          {
            required: true,
            trigger: "change",
            message: "请输入验证码"
          }
        ]
       },

     }

      // 校验通过以后 掉接口 this.$refs.shop 和html中ref对应
      this.$refs.loginForm.validate(valid => {
            if(valid){
                //成功
            }
        });
```

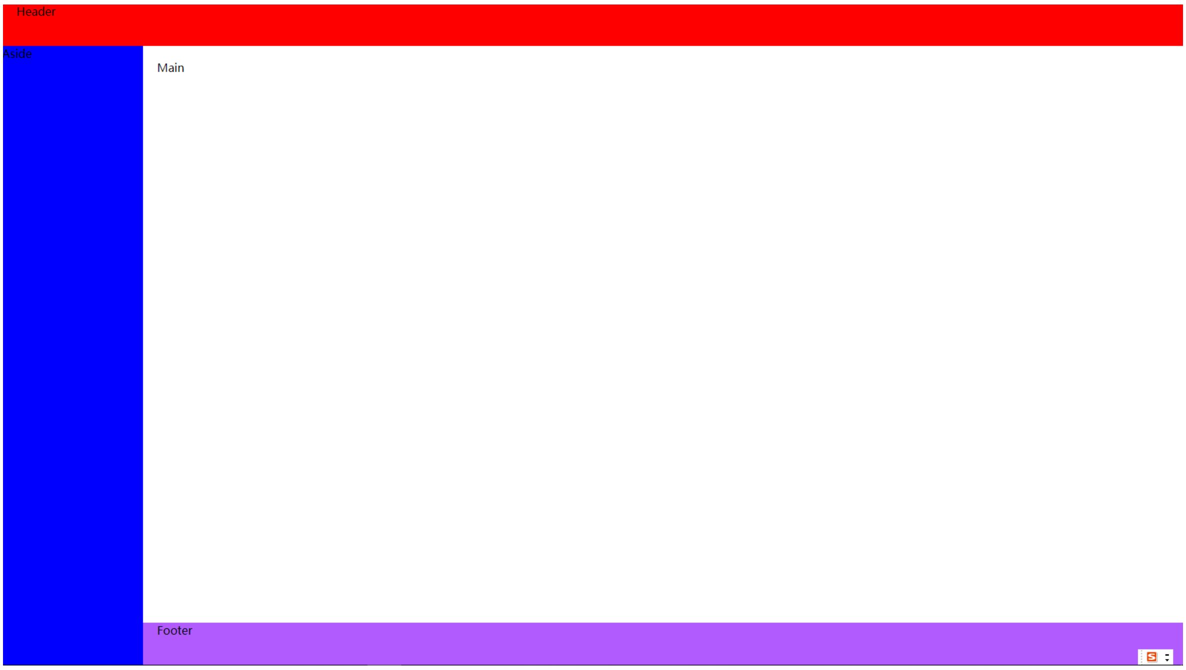
第08讲 主界面布局

1.1、采用上、下布局，下分为左右两部分

1.2、布局查看官方文档 https://element.eleme.io/#/zh-CN/component/installation

1.3、总体布局

1.3.1



样式需要设置高度为100%，才能占满屏幕高度

```
<template>
  <el-container class="home">
    <!-- 头部 -->
    <el-header style="background:red;">Header</el-header>
    <el-container>
      <!-- 左侧菜单 -->
      <el-aside width="200px" style="background:blue;">Aside</el-aside>
      <!-- 右侧内容显示区 -->
      <el-container style="background:#B15BFF">
        <el-main style='background:#FFF;'>Main</el-main>
        <el-footer>Footer</el-footer>
      </el-container>
    </el-container>
  </el-container>
</template>

<script>
export default {
  name: "home",
  components: {}
};
</script>
<style scoped>
.home {
  height: 100%;
}
</style>
```

1.3.2头部布局

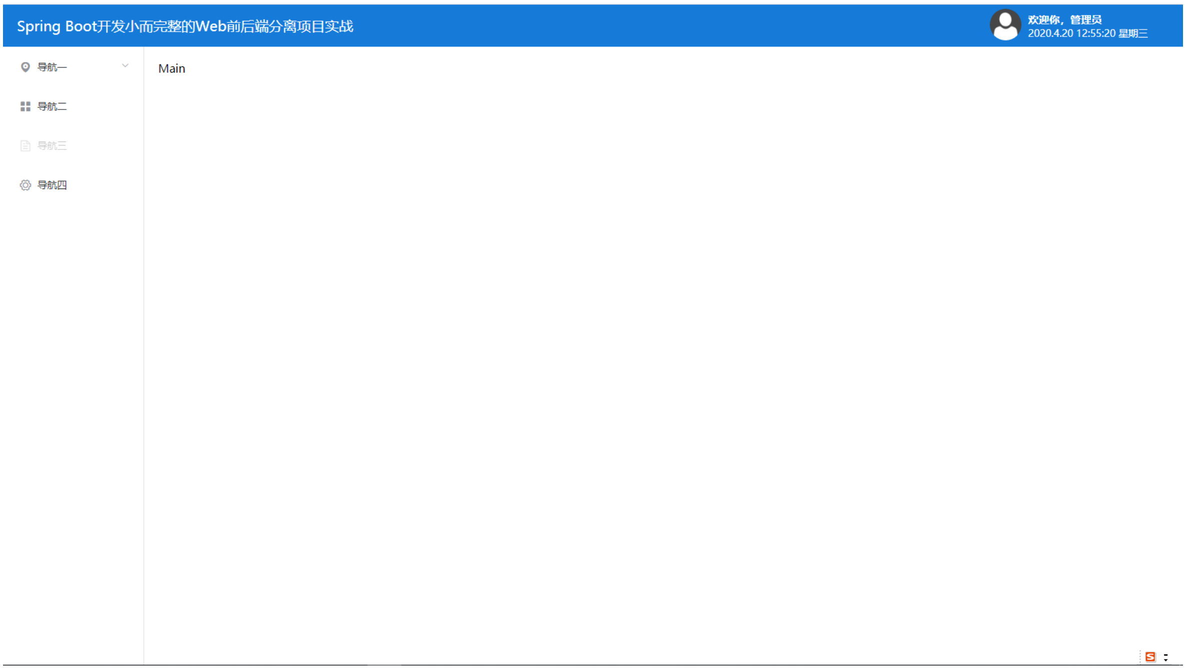
```
 <!-- 头部 -->
    <el-header class="header ub main-justify cross-center">
      <div class="header-title">Spring Boot开发小而完整的Web前后端分离项目实战</div>
      <div class="ub main-center cross-center header-right">
        <div>
          <el-dropdown placement='bottom-start'>
            <img class="header-img" src="../assets/images/avatar.jpg" alt="用户头像" />
            <el-dropdown-menu slot="dropdown">
              <el-dropdown-item>个人中心</el-dropdown-item>
              <el-dropdown-item>退出</el-dropdown-item>
            </el-dropdown-menu>
          </el-dropdown>
        </div>
        <div class="header-right-user">
          <div class="header-wollcom">欢迎你，管理员</div>
          <div class="header-time">2020.4.20 12:55:20 星期三</div>
        </div>
      </div>
    </el-header>
```

头部css样式

```
.header {
  background: #167bd8;
  color: #fff;
  padding: 0 20px;
}
.header-title {
  font-size: 20px;
}
.header-img {
  height: 45px;
  width: 45px;
  border-radius: 50%;
  cursor: pointer;
}
.header-wollcom {
  font-size: 15px;
  font-weight: 600;
}
.header-time {
  font-size: 14px;
}
.header-right-user {
  margin-left: 10px;
}
.header-right {
  margin-right: 30px;
}
```

第09讲 左侧菜单实现

1.1、左侧菜单布局


Spring Boot开发小而完整的Web前后端分离项目实战
欢迎你，管理员
2020.4.20 12:55:20 星期三
导航一
导航二
导航三
导航四
Main

1.2、用到的组件 el-menu 组件

主要属性：

default-active　当前激活菜单的 index

unique-opened　是否只保持一个子菜单的展开

router　是否使用 vue-router 的模式，启用该模式会在激活导航时以 index 作为 path 进行路由跳转

collapse　是否水平折叠收起菜单（仅在 mode 为 vertical 时可用）

示例代码：

```
<!-- 左侧菜单 -->
```

```
      <el-aside  width="auto">
        <el-menu
          default-active="1-4-1"
          class="menu-bar"
          @open="handleOpen"
          @close="handleClose"
          :collapse="isCollapse"
        >
          <el-submenu index="1">
            <template slot="title">
              <i class="el-icon-location"></i>
              <span slot="title">导航一</span>
            </template>
            <el-menu-item-group>
              <span slot="title">分组一</span>
              <el-menu-item index="1-1">选项1</el-menu-item>
              <el-menu-item index="1-2">选项2</el-menu-item>
            </el-menu-item-group>
            <el-menu-item-group title="分组2">
              <el-menu-item index="1-3">选项3</el-menu-item>
            </el-menu-item-group>
            <el-submenu index="1-4">
              <span slot="title">选项4</span>
              <el-menu-item index="1-4-1">选项1</el-menu-item>
            </el-submenu>
          </el-submenu>
          <el-menu-item index="2">
            <i class="el-icon-menu"></i>
            <span slot="title">导航二</span>
          </el-menu-item>
          <el-menu-item index="3" disabled>
            <i class="el-icon-document"></i>
            <span slot="title">导航三</span>
          </el-menu-item>
          <el-menu-item index="4">
            <i class="el-icon-setting"></i>
            <span slot="title">导航四</span>
          </el-menu-item>
        </el-menu>
      </el-aside>
```

```
.el-container /deep/ .el-menu {
  /* border-right: solid 1px #e6e6e6; */
  border-right: none !important;
}
.el-aside {
  border-right: solid 1px #e6e6e6;
}
/* 此样式用于设置  el-aside  width="auto" 宽度为auto的样式 */
.menu-bar:not(.el-menu--collapse) {
  width: 200px;
  min-height: 400px;
}
```

1.3、菜单组件重构

1、新建MenuBar组件

```
<template>
```

```
  <div>
    <el-menu class="menu-bar" default-active="$route.path"  unique-opened  router>
        <menu-item :menuList='menuList'></menu-item>
    </el-menu>
  </div>
</template>

<script>
import MenuItem from "./MenuItem";
export default {
  components: {
    MenuItem
  },
  data() {
    return {
      menuList: [
        {
          children: [
            {
              children: [],
              code: "sys:dept",
              createTime: 1586703509000,
              icon: "el-icon-copy-document",
              id: 33,
              isHome: 0,
              label: "机构管理",
              name: "departmentList",
              orderNum: 2,
              parentId: 17,
              path: "/departmentList",
              remark: "机构管理",
              type: "1",
              updateTime: 1586337139000,
              url: "/system/Department/DepartmentList"
            },
            {
              children: [],
              code: "sys:user",
              createTime: 1691464271000,
              icon: "el-icon-s-custom",
              id: 18,
              isHome: 0,
              label: "用户管理",
              name: "userList",
              orderNum: 3,
              parentId: 17,
              path: "/userList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/User/UserList"
            },
            {
              children: [],
              code: "sys:role",
              createTime: 1691464271000,
              icon: "el-icon-rank",
              id: 23,
              isHome: 0,
              label: "角色管理",
              name: "roleList",
              orderNum: 4,
              parentId: 17,
              path: "/roleList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/Role/RoleList"
            },
            {
              children: [],
              code: "sys:menu",
              createTime: 1691464271000,
              icon: "el-icon-menu",
              id: 28,
              isHome: 0,
              label: "权限管理",
              name: "menuList",
              orderNum: 5,
              parentId: 17,
              path: "/menuList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/Menu/MenuList"
            }
          ],
          code: "sys:manage",
          createTime: 1691464271000,
          icon: "el-icon-document",
          id: 17,
          isHome: 0,
          label: "系统管理",
          orderNum: 1,
          parentId: 0,
          path: "/system",
          type: "0",
          updateTime: 1691565988000
        },
        {
          children: [
            {
              children: [],
              code: "sys:goodsCategory",
              createTime: 1586703272000,
              icon: "el-icon-s-data",
              id: 36,
              isHome: 0,
              label: "分类管理",
              name: "goodCategory",
              orderNum: 1,
              parentId: 34,
              path: "/goodCategory",
              type: "1",
              updateTime: 1586683590000,
              url: "/goods/goodsCategory/goodsCategoryList"
            },
            {
              children: [],
              code: "sys:goodsBrand",
              createTime: 1586683924000,
              icon: "el-icon-tickets",
              id: 37,
              isHome: 0,
              label: "品牌管理",
              name: "goodsBrand",
              orderNum: 2,
              parentId: 34,
              path: "/goodsBrand",
              type: "1",
              updateTime: 1586683924000,
              url: "/goods/goodsBrand/goodsBrandList"
            }
          ],
          code: "sys:goods",
          createTime: 1586702987000,
          icon: "el-icon-picture",
          id: 34,
          isHome: 0,
          label: "商品管理",
          name: "",
          orderNum: 2,
          parentId: 0,
          path: "/goods",
          type: "0",
          updateTime: 1586683323000
        },
        {
          children: [
            {
              children: [],
              code: "sys:systemCode",
              createTime: 1587012282000,
              icon: "el-icon-files",
              id: 43,
              isHome: 0,
              label: "代码生成",
              name: "systemCode",
              orderNum: 0,
              parentId: 42,
              path: "/systemCode",
              type: "1",
              updateTime: 1586684646000,
              url: "/system/config/code"
            },
            {
              children: [],
              code: "sys:document",
              createTime: 1586748705000,
              icon: "el-icon-s-operation",
              id: 77,
              isHome: 0,
              label: "接口文档",
              name: "document",
              orderNum: 0,
              parentId: 42,
              path: "/document",
              type: "1",
              updateTime: 1586748705000,
              url: "/system/config/systemDocument"
            }
          ],
          code: "sys:systenConfig",
          createTime: 1586703003000,
          icon: "el-icon-receiving",
          id: 42,
          isHome: 0,
          label: "系统工具",
          name: "",
          orderNum: 3,
          parentId: 0,
          path: "/systenConfig",
          type: "0",
          updateTime: 1586684441000
        }
      ]
    };
  }
};
</script>

<style lang="css" scoped>
/* 此样式用于设置  el-aside  width="auto" 宽度为auto的样式 */
.menu-bar:not(.el-menu--collapse) {
  width: 200px;
  min-height: 400px;
}
</style>
```

2、新建MenuItem组件

```
<template>
  <div>
    <template v-for="menu in menuList">
      <el-submenu  v-if='menu.children.length > 0 ' :index="menu.path"
:key='menu.path'>
        <template slot="title">
          <i :class="menu.icon"></i>
          <span style=" font-size: 15px;font-weight: 600;" slot="title">{{menu.label}}
</span>
        </template>
        <menu-item :menuList='menu.children' />
        <!-- <el-submenu index="1-4">
          <span slot="title">选项4</span>
          <el-menu-item index="1-4-1">选项1</el-menu-item>
        </el-submenu> -->
      </el-submenu>
      <el-menu-item @click="selectMenu(menu)" v-else :index="menu.path"
:key='menu.path'>
        <i :class="menu.icon"></i>
        <span slot="title">{{menu.label}}</span>
      </el-menu-item>
    </template>
  </div>
</template>

<script>
import MenuItem from './MenuItem.vue'
export default {
  name:'MenuItem',
  props: ["menuList"],
  components:{
      MenuItem
  },
  created(){
    console.log('555555')
    console.log(this.menuList)
  },
  methods:{
    //菜单点击事件
    selectMenu(item){
    //  1.把点击的菜单设置到tabs
        this.$store.commit('selectMenu',item);
    //  2.跳转到路由
        // this.$router.push({
        //   name:item.name
        // })
    }
  }
};
</script>

<style lang="scss" scoped>
</style>
```

第10讲 tabs选项卡实现与菜单联动

1.1、tabs选项卡组件:

1、 tabs 组件

2、常用属性:

value ： 选中选项卡的name

type ： 选项卡风格　可选择　card / border-card

closable ： 选项卡是否可以关闭

1.2、实现的效果：点击左侧菜单，右边内容展示区要显示对应的tabs菜单选项卡，tabs选项卡可以关闭

1.3、实现原理：

1.3.1、点击左侧菜单，把当前点击的菜单对象，添加到tabs选项卡里面；

1.3.2、关闭tabs选项卡：如果为首页桌面不能关闭

1.4、tabs组件实现

```
<template>
  <div>
    <el-tabs :value="editableTabsValue" type="card" closable @tab-remove="removeTab">
      <el-tab-pane
        v-for="item in editableTabs"
        :key="item.name"
        :label="item.title"
        :name="item.name"
      >{{item.content}}</el-tab-pane>
    </el-tabs>
  </div>
</template>

<script>
// import {mapState} from 'vuex'
export default {
  name: "tabs",
  computed:{
    // ...mapState({
    //   //此处报 Computed property "editableTabs" was assigned to but it has no setter.
    //   //需要把v-model改为 :value即可
    //   editableTabs: state => state.MenuStore.tabs
    // }),
    editableTabs:({
      get(){
       return this.$store.state.MenuStore.tabs
      },
      set(val){
        this.$store.state.MenuStore.tabs = val;
      }
    }),
    editableTabsValue:{
      get(){
        return this.$store.state.MenuStore.editableTabsValue
      },
      set(val){
        this.$store.state.MenuStore.editableTabsValue = val;
      }
    }
  },
  data() {
    return {
      //选项卡的name
     // editableTabsValue: "2",
      // editableTabs: [
      //   {
      //     title: "Tab 1",
      //     name: "1",
      //     content: "Tab 1 content"
      //   },
      //   {
      //     title: "Tab 2",
      //     name: "2",
      //     content: "Tab 2 content"
      //   }
      // ],
      tabIndex: 2
    };
  },
  methods: {
    //关闭tabs  targetName为关闭选项卡的名称
    removeTab(targetName) {
      //首页不能关闭
      if(targetName === 'desktop'){
        return;
      }
      let tabs = this.editableTabs;
      //当前激活的选项卡
      let activeName = this.editableTabsValue;
      if (activeName === targetName) {
        tabs.forEach((tab, index) => {
          if (tab.name === targetName) {
            let nextTab = tabs[index + 1] || tabs[index - 1];
            if (nextTab) {
              activeName = nextTab.name;
            }
          }
        });
      }
      //当前激活的选项卡
      this.editableTabsValue = activeName;
       //路由跳到当前激活的选项卡
      this.editableTabs = tabs.filter(tab => tab.name !== targetName);


    }
  }
};
</script>

<style lang="scss" scoped>
</style>
```

由于tabs选项卡数据存放在store里面，所以要新建一个MenuStore.js，如下所示

```
import Vue from 'vue'
import Vuex from 'vuex'
Vue.use(Vuex)
export default {
    state: {
        editableTabsValue:'desktop',
        //tabs数据
        tabs: [
            {
                title: '首页',
                name: 'desktop'
            }
        ]
    },
    mutations: {
        selectMenu(state,val){
            console.log(val);
            //1.把点击的菜单加到tabs里面,如果不存在才添加
            let res = state.tabs.findIndex(item => item.name === val.name)
            if(res === -1){
                let obj = {}
                obj.title = val.label
                obj.name = val.name
                state.tabs.push(obj)
            }
            //当前激活的选项卡
            state.editableTabsValue = val.name;

        }
    },
    actions: {
    }
}
```

1.5、解决刷新浏览器，vuex 里的 tabs 不存在的问题

在点击菜单，添加tabs时，把当前tabsList存到sessionStorage中，进入路由前，从sessionStorage取出当前tabsList数据

1.5.1、在MenuStore.js中 selectMenu()中添加如下代码

```
//解决浏览器刷新tabs不存在的问题
sessionStorage.setItem('tabsList',JSON.stringify(state.tabs));
```

1.5.2、在MenuStore.js中添加两个方法，用于获取当前tabsList的数据和当前需要激活的tabs选项卡

```
       //刷新浏览器，进入路由时调用，获取tabs数据
        getTabs(state){
            let tabs = sessionStorage.getItem('tabsList');
            if(tabs){
                let currentTabsList = JSON.parse(tabs);
                state.tabs = currentTabsList;
            }
        },
        //用于设置当前激活的选项卡
        setActiveTabs(state,curent){
            state.editableTabsValue = curent;
        }
```

1.5.3、在main.js中添加路由拦截

```
router.beforeEach((to,from,next) => {
    //to 即将进入的路由
    //from  即将离开的路由
    //设置tabs数据
    store.commit('getTabs');
    //设置激活选项卡
    store.commit('setActiveTabs',to.name);
    next(); //继续往下执行
})
```

1.5.4、在views下面新建system 的各个页面

1.5.6、在路由home中添加上面新建的路由作为子路由

```
children: [
      {
        path: '/',
        name: 'desktop',
        component: () => import('@/views/Desktop.vue')
      },
      {
        path: '/userList',
        name: 'userList',
        component: () => import('@/views/system/User/UserList.vue')
      },
      {
        path: '/departmentList',
        name: 'departmentList',
        component: () => import('@/views/system/Department/DepartmentList.vue')
      },
      {
        path: '/roleList',
        name: 'roleList',
        component: () => import('@/views/system/Role/RoleList.vue')
      },
      {
        path: '/menuList',
        name: 'menuList',
        component: () => import('@/views/system/Menu/MenuList.vue')
      }
    ]
```

1.5.7、关闭tabs出现的错误

Computed property "editableTabsValue" was assigned to but it has no setter.

解决方式为：在tabs组件computed中为 editableTabsValue设置 get 和 set方法，如下所示：

1.5.8、解决message: "Navigating to current location ("/homePage") is not allowed",警告的问题

```


错误代码如下：

NavigationDuplicated {_name: "NavigationDuplicated", name: "NavigationDuplicated",
message: "Navigating to current location ("/index") is not allowed", stack: "Error↵ at
new NavigationDuplicated (webpack-int…e_modules/element-
ui/lib/mixins/emitter.js:29:22)"}
操作：
点击左侧菜单两次，路由切换两次

原因：
在路由跳转的时候同一个路由多次添加是不被允许的
```

解决方式:

1.路由返回3.0版本

2.router中 index.js添加如下代码:

```
const VueRouterPush = VueRouter.prototype.push
VueRouter.prototype.push = function push (to) {
    return VueRouterPush.call(this, to).catch(err => err)
}
```

1.6、选项卡点击事件

```
btnClick(tab) {
      var obj = {};
      if (this.editableTabsValue === "desktop") {
        obj.label = "首页";
      } else {
        obj.label = tab.label;
      }
       obj.name = this.editableTabsValue;
        this.$router.push({ name: this.editableTabsValue });
        this.$store.commit("selectMenu", obj);
    },
```

第11 讲 动态路由的生成

1.1、使用动态路由的原因

由于系统左侧菜单是根据不同用户，拥有不同的菜单，所以左侧菜单和路由是动态的生成的

1.2、实现原理

利用 vue-router 的 addRoutes 方法可以动态添加路由 ， router.addRoutes(routes: Array)，动态添加路由，参数必是一个符合 routes 选项要求的数组

1.3、实现思路

1.3.1、用户登录时，会根据用户自己拥有的权限，返回菜单数据和路由数据

1.3.2、登录成功，返回菜单和路由数据，前端保存到sessionStorage中；在Login.vue页面登录提交事件中添加如下代码，mentList表示后台返回成功的左侧菜单数据，routerList表示后台返回的路由数据

```
  let menuList = [
        {
          children: [
            {
              children: [],
              code: "sys:dept",
              createTime: 1586703509000,
              icon: "el-icon-copy-document",
              id: 33,
              isHome: 0,
              label: "机构管理",
              name: "departmentList",
              orderNum: 2,
              parentId: 17,
              path: "/departmentList",
              remark: "机构管理",
              type: "1",
              updateTime: 1586337139000,
              url: "/system/Department/DepartmentList"
            },
            {
              children: [],
              code: "sys:user",
              createTime: 1691464271000,
              icon: "el-icon-s-custom",
              id: 18,
              isHome: 0,
              label: "用户管理",
              name: "userList",
              orderNum: 3,
              parentId: 17,
              path: "/userList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/User/UserList"
            },
            {
              children: [],
              code: "sys:role",
              createTime: 1691464271000,
              icon: "el-icon-rank",
              id: 23,
              isHome: 0,
              label: "角色管理",
              name: "roleList",
              orderNum: 4,
              parentId: 17,
              path: "/roleList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/Role/RoleList"
            },
            {
              children: [],
              code: "sys:menu",
              createTime: 1691464271000,
              icon: "el-icon-menu",
              id: 28,
              isHome: 0,
              label: "权限管理",
              name: "menuList",
              orderNum: 5,
              parentId: 17,
              path: "/menuList",
              type: "1",
              updateTime: 1691565988000,
              url: "/system/Menu/MenuList"
            }
          ],
          code: "sys:manage",
          createTime: 1691464271000,
          icon: "el-icon-document",
          id: 17,
          isHome: 0,
          label: "系统管理",
          orderNum: 1,
          parentId: 0,
          path: "/system",
          type: "0",
          updateTime: 1691565988000
        },
        {
          children: [
            {
              children: [],
              code: "sys:goodsCategory",
              createTime: 1586703272000,
              icon: "el-icon-s-data",
              id: 36,
              isHome: 0,
              label: "分类管理",
              name: "goodCategory",
              orderNum: 1,
              parentId: 34,
              path: "/goodCategory",
              type: "1",
              updateTime: 1586683590000,
              url: "/goods/goodsCategory/goodsCategoryList"
            },
            {
              children: [],
              code: "sys:goodsBrand",
              createTime: 1586683924000,
              icon: "el-icon-tickets",
              id: 37,
              isHome: 0,
              label: "品牌管理",
              name: "goodsBrand",
              orderNum: 2,
              parentId: 34,
              path: "/goodsBrand",
              type: "1",
              updateTime: 1586683924000,
              url: "/goods/goodsBrand/goodsBrandList"
            }
          ],
          code: "sys:goods",
          createTime: 1586702987000,
          icon: "el-icon-picture",
          id: 34,
          isHome: 0,
          label: "商品管理",
          name: "",
          orderNum: 2,
          parentId: 0,
          path: "/goods",
          type: "0",
          updateTime: 1586683323000
        },
        {
          children: [
            {
              children: [],
              code: "sys:systemCode",
              createTime: 1587012282000,
              icon: "el-icon-files",
              id: 43,
              isHome: 0,
              label: "代码生成",
              name: "systemCode",
              orderNum: 0,
              parentId: 42,
              path: "/systemCode",
              type: "1",
              updateTime: 1586684646000,
              url: "/system/config/code"
            },
            {
```

```
        children: [],
        code: "sys:document",
        createTime: 1586748705000,
        icon: "el-icon-s-operation",
        id: 77,
        isHome: 0,
        label: "接口文档",
        name: "document",
        orderNum: 0,
        parentId: 42,
        path: "/document",
        type: "1",
        updateTime: 1586748705000,
        url: "/system/config/systemDocument"
      }
    ],
    code: "sys:systenConfig",
    createTime: 1586703003000,
    icon: "el-icon-receiving",
    id: 42,
    isHome: 0,
    label: "系统工具",
    name: "",
    orderNum: 3,
    parentId: 0,
    path: "/systenConfig",
    type: "0",
    updateTime: 1586684441000
  }
];
//路由数据
let routerList= [{
      "children": [],
      "code": "sys:systemCode",
      "createTime": 1587012282000,
      "icon": "el-icon-files",
      "id": 43,
      "isHome": 0,
      "label": "代码生成",
      "name": "systemCode",
      "orderNum": 0,
      "parentId": 42,
      "path": "/systemCode",
      "type": "1",
      "updateTime": 1586684646000,
      "url": "/system/config/code"
  }, {
      "children": [],
      "code": "sys:document",
      "createTime": 1586748705000,
      "icon": "el-icon-s-operation",
      "id": 77,
      "isHome": 0,
      "label": "接口文档",
      "name": "document",
      "orderNum": 0,
      "parentId": 42,
      "path": "/document",
      "type": "1",
      "updateTime": 1586748705000,
      "url": "/system/config/systemDocument"
```

```
        }, {
            "children": [],
            "code": "sys:goodsCategory",
            "createTime": 1586703272000,
            "icon": "el-icon-s-data",
            "id": 36,
            "isHome": 0,
            "label": "分类管理",
            "name": "goodCategory",
            "orderNum": 1,
            "parentId": 34,
            "path": "/goodCategory",
            "type": "1",
            "updateTime": 1586683590000,
            "url": "/goods/goodsCategory/goodsCategoryList"
        }, {
            "children": [],
            "code": "sys:goodsBrand",
            "createTime": 1586683924000,
            "icon": "el-icon-tickets",
            "id": 37,
            "isHome": 0,
            "label": "品牌管理",
            "name": "goodsBrand",
            "orderNum": 2,
            "parentId": 34,
            "path": "/goodsBrand",
            "type": "1",
            "updateTime": 1586683924000,
            "url": "/goods/goodsBrand/goodsBrandList"
        }, {
            "children": [],
            "code": "sys:dept",
            "createTime": 1586703509000,
            "icon": "el-icon-copy-document",
            "id": 33,
            "isHome": 0,
            "label": "机构管理",
            "name": "departmentList",
            "orderNum": 2,
            "parentId": 17,
            "path": "/departmentList",
            "remark": "机构管理",
            "type": "1",
            "updateTime": 1586337139000,
            "url": "/system/Department/DepartmentList"
        }, {
            "children": [],
            "code": "sys:user",
            "createTime": 1691464271000,
            "icon": "el-icon-s-custom",
            "id": 18,
            "isHome": 0,
            "label": "用户管理",
            "name": "userList",
            "orderNum": 3,
            "parentId": 17,
            "path": "/userList",
            "type": "1",
            "updateTime": 1691565988000,
            "url": "/system/User/UserList"
        }, {
            "children": [],
            "code": "sys:role",
            "createTime": 1691464271000,
            "icon": "el-icon-rank",
            "id": 23,
            "isHome": 0,
            "label": "角色管理",
            "name": "roleList",
            "orderNum": 4,
            "parentId": 17,
            "path": "/roleList",
            "type": "1",
            "updateTime": 1691565988000,
            "url": "/system/Role/RoleList"
        }, {
            "children": [],
            "code": "sys:menu",
            "createTime": 1691464271000,
            "icon": "el-icon-menu",
            "id": 28,
            "isHome": 0,
            "label": "权限管理",
            "name": "menuList",
            "orderNum": 5,
            "parentId": 17,
            "path": "/menuList",
            "type": "1",
            "updateTime": 1691565988000,
            "url": "/system/Menu/MenuList"
        }];
//保存菜单数据
sessionStorage.setItem("menuList", JSON.stringify(menuList));
//保存路由数据
sessionStorage.setItem("routerList", JSON.stringify(routerList));
//动态生成路由
this.$store.commit("getMenuList", this.$router);
//跳转到home页面
this.$router.push("home");
```

router.js中的routes只留如下代码:

```
routes:[
    {
      path: "/login",
      name: "login",
      component: () => import('@/views/Login')
    },
    {
      path: "/home",
      name: "home",
      component: Home,
      children: [
         {
          path:'/',
          name:'desktop',
          component:()=>import('./views/Desktop.vue')
        }
      ]
    }
  ]
```

1.3.3、调用store中动态的生成菜单和路由，在MenuStore.js的mutations中添加如下方法，代码如下

```
       //获取菜单数据和生成路由
        getMenuList(state, router) {
            //当前存在的路由
            let newRoutes = router.options.routes;
            //从sessionStorage获取menuList数据
            let menuList = JSON.parse(sessionStorage.getItem('menuList'));
            //把后端返回的数据设置到state中的menu_data中
            state.menu_data = menuList;
            let routerList = JSON.parse(sessionStorage.getItem('routerList'));
            routerList.forEach(item => {
                //生成 component: () => import('@/views/Login.vue')
                item.component = () => import(`@/views${item.url}.vue`);
                //newRoutes[1] 表示获取到home路由,把后台返回的路由添加到该路由的子路由
                newRoutes[1].children.push(item);
            });
            router.addRoutes(newRoutes);
        }
```

1.3.4、在MenuBar.vue组件中获取store中的menu_data数据,代码如下

```
 computed:{
    ...mapState({
      menuList: state => state.MenuStore.menu_data
    })
  },
```

1.3.5、防止刷新后store中的menu_data不存在，那么需要在main.js做如下判断，如果store中的menu_data不存在，要重新加载一个sessionStorage中的数据,在main.js中做如下更改:

```
router.beforeEach((to, from, next) => {
  console.log(to);
  console.log(from);
  //设置tabs数据
  store.commit('getTabs');
  //设置激活选项卡
  store.commit('setActiveTabs', to.name);
  //如果store中的菜单数据menu_data被刷新了，那么从新加载
  if (store.state.MenuStore.menu_data.length == 0) {
    store.commit('getMenuList', router);
    next({ path: to.path })
  } else {
    next();
  }
})
```

第12讲 实现左侧菜单的展开和收缩

1.1、实现效果

点击收缩图标，左侧菜单收缩

1.2、实现原理

利用 el-menu中的属性 collapse可以设置菜单展开和收缩

1.3、收缩图标制作

展开时图标 el-icon-s-fold

收缩时图标 el-icon-s-unfold

1.3.1、在home页面 el-main 中tabs前面添加收缩图标,代码如下:

```
    <i class="el-icon-s-fold arrow-icon" @click="arrowBtn"></i>
```

样式

```
//解决图标不能点击的问题
.el-tabs__header{
  position: static;
}
.arrow-icon {
  float: left;
  background: #eaedf1;
  border: 1px solid transparent;
  font-size: 23px;
  height: 39px;
  line-height: 39px!important;
  width: 40px;
  text-align: center;
}
```

1.3.2、设置菜单收缩属性 isCollapse

1、在MenuStore.js的state中添加 isCollapse:false

2、在MenuStore.js的mutations中添加如下代码

```
//设置图标收缩属性
        setOpenOrClose(state){
            state.isCollapse = !state.isCollapse;
        }
```

`   3、在MenuBar.vue中通过计算属性获取 isCollapse，把原来的注释，代码如下

```
 computed:{
    ...mapState({

      isCollapse: state => state.MenuStore.isCollapse
    })
  },
```

4、在Home.vue页面收缩图标点击事件中调用MenuStore.js 的setOpenOrClose方法

```
    import {mapMutations} from 'vuex'
    methods:{
        方法一
        //菜单收缩
         ...mapMutations({
           arrowBtn: "setOpenOrClose"
         }),
        方法二
        arrowBtn(){
          console.log('点击图标')
          this.$store.commit('setOpenOrClose')
        }
    }
```

1.3.3、让点击图标自动切换方向

1、获取MenuStore.js中的isCollapse

```
 computed:{
    ...mapState({
      isCollapse: state => state.MenuStore.isCollapse
    })
  },
```

2、动态设置图标样式

```
 :class="[collapse ? 'el-icon-s-unfold' : ' el-icon-s-fold']"
```

1.4、解决图标收缩时，字不能影藏的问题

1.4.1、安装 npm install --save vue-fragment

1.4.2、引入fragement

```
// main.js
import Fragment from 'vue-fragment'
Vue.use(Fragment.Plugin)
```

1.4.2、修改MenuItem.vue，用fragment包裹，代码如下

```
<template>
  <fragment>
    <template v-for="menu in menuList">
      <el-submenu v-if="menu.children.length > 0 " :index="menu.path"
:key="menu.path">
        <template slot="title">
          <i :class="menu.icon"></i>
          <span style="font-size:15px;font-weight:600;" slot="title">{{menu.label}}
</span>
        </template>
        <menu-item :menuList="menu.children"></menu-item>
      </el-submenu>
      <el-menu-item @click="clickBtn(menu)" v-else :index="menu.path"
:key="menu.path">
```

```
        <i :class="menu.icon"></i>
        <span slot="title">{{menu.label}}</span>
      </el-menu-item>
    </template>
  </fragment>
</template>
```

第13讲 首页和角色管理列表讲解

1.1、首页代码

```
<template>
  <div style="margin:20px 20px;">
    <el-row :gutter="20" type="flex" class="row-bg" justify="center">
      <el-col :span="12">
        <div class="grid-content bg-purple ub column-top"
style='background:#00c0ef;color:#FFF;height:120px;border-radius:5px'>
            <div class='ub-f1' style='font-size:38px;font-weight:
bold;padding:20px;'>0</div>
            <div class='' style="background: #3399FF;height:30px;text-align:'center">
今日支付订单</div>
        </div>
      </el-col>
      <el-col :span="12">
        <div class="grid-content bg-purple ub column-top"
style='background:#00c0ef;color:#FFF;height:120px;border-radius:5px'>
            <div class='ub-f1' style='font-size:38px;font-weight:
bold;padding:20px;'>0</div>
            <div class='' style="background: #3399FF;height:30px;text-align:'center">
今日待发货</div>
        </div>
      </el-col>
    </el-row>
    <el-row :gutter="20" type="flex" class="row-bg" justify="center" style="margin-
top:20px;">
      <el-col :span="12">
        <div class="grid-content bg-purple ub column-top"
style='background:#00c0ef;color:#FFF;height:120px;border-radius:5px'>
            <div class='ub-f1' style='font-size:38px;font-weight:
bold;padding:20px;'>0</div>
            <div class='' style="background: #3399FF;height:30px;text-align:'center">
今日已发货订单</div>
        </div>
      </el-col>
      <el-col :span="12">
        <div class="grid-content bg-purple ub column-top"
style='background:#dd4b39;color:#FFF;height:120px;border-radius:5px'>
            <div class='ub-f1' style='font-size:38px;font-weight:
bold;padding:20px;'>0</div>
            <div class='' style="background: rgba(0, 0, 0, 0.1);height:30px;text-
align:'center">今日异常</div>
        </div>
      </el-col>
    </el-row>
  </div>
</template>

<script>
```

```
export default {
  data() {
    return {
      userInfo: {
        userName: ""
      }
    };
  }
};
</script>

<style lang="scss" scoped>
</style>
```

1.2、角色搜索表单制作

1.2.1、列表搜索框 搜索按钮 新增按钮 实现

1.2.2、使用组件 el-form el-input el-row，代码实现如下

```
<el-form size="mini" :model="searchForm" label-width="80px">
      <el-row>
        <el-col :span="5">
          <el-form-item label="名称">
            <el-input v-model="searchForm.roleName" placeholder="请输入角色名称"></el-
input>
          </el-form-item>
        </el-col>
        <el-button class="searchBtn"  type="primary" size="mini" icon="el-icon-search"
>查询</el-button>
        <el-button class="searchBtn"  type="primary" size="mini" icon="el-icon-search"
>新增</el-button>
      </el-row>
    </el-form>
```

1.3、角色列表制作

1.3.1、使用组件　table组件 分页组件

1.3.2、使用固定表头的table组件　只要在el-table元素中定义了height属性，即可实现固定表头的表格，而不需要额外的代码，官方代码运行如下

```
 <el-table
    :data="tableData"
    height="250"
    border
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
```

主要属性

:data 绑定表格数据

size: 表格尺寸　可选 medium / small / mini

stripe　是否为斑马线

height 表格高度

1.3.3、设置表格显示高度

```
      //表格高度　window.innerHeight窗口文档显示高度
      tableHeight:window.innerHeight



      //　该钩子函数执行时所有的DOM挂载和渲染都已完成，此时在该钩子函数中进行任何DOM操作都不
会有问题
    // 在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，
    // 这个操作都应该放进Vue.nextTick()的回调函数中
    mounted() {
    this.$nextTick(() => {
      this.tableHeight = window.innerHeight - 210; //后面的50：根据需求空出的高度，自行调
整
    });
  }
```

1.3.4、表格分页组件　Pagination

```
  <el-pagination
      @size-change="handleSizeChange"
      @current-change="handleCurrentChange"
      :current-page.sync="currentPage1"
      :page-size="100"
      layout="total, prev, pager, next"
      :total="1000">
    </el-pagination>
```

组件属性

size-change: 当page-sizes 改变时触发事件

current-change：当页数发生变化时触发事件，

current-page：当前是第几页

page-size：页容量，也就是每页多少条数据

total:总共有多少条数据,后台返回数据数值

注意：

1.静态数据的时候，table的data要从新计算，表格数据才会改变

2.:current-page.sync 要加 sync 才会自动改变

currentPage的数据,上一页和下一页时，表格数据才会改变

3. 中的prop要跟返回的数据字段对应才能显示

1.3.5、表格编辑、删除按钮

```
<el-table-column label="操作" width="160" align="center">
        <template slot-scope="scope">
          <el-button
            @click.native.prevent="editRow(scope.$index, tableData)" type="primary" size="mini" >编辑
          </el-button>
           <el-button
            @click.native.prevent="deleteRow(scope.$index, tableData)" type="danger" size="mini" >删除
          </el-button>
        </template>
      </el-table-column>
```

1.4、表格最终代码

```
<template>
  <el-main>
    <!-- 搜索表单 -->
    <el-form size="mini" :model="searchForm" label-width="80px">
      <el-row>
        <el-col :span="5">
          <el-form-item label="名称">
            <el-input v-model="searchForm.roleName" placeholder="请输入角色名称"></el-input>
          </el-form-item>
        </el-col>
        <el-button class="searchBtn" type="primary" size="mini" icon="el-icon-search">查询</el-button>
        <el-button class="searchBtn" type="primary" size="mini" icon="el-icon-search">新增</el-button>
      </el-row>
    </el-form>
    <!-- 角色列表 -->
    <el-table
      :data="tableData"
      size="mini"
      :stripe="true"
      :height="tableHeight"
      border
      style="width: 100%"
    >
      <el-table-column prop="date" label="日期"></el-table-column>
      <el-table-column prop="name" label="姓名"></el-table-column>
      <el-table-column prop="address" label="地址"></el-table-column>
      <el-table-column label="操作" width="160" align="center">
        <template slot-scope="scope">
          <el-button
            @click.native.prevent="editRow(scope.$index, tableData)" type="primary" size="mini" >编辑
          </el-button>
           <el-button
```

```
            @click.native.prevent="deleteRow(scope.$index, tableData)" type="danger" size="mini" >删除
          </el-button>
        </template>
      </el-table-column>
    </el-table>
    <!-- 分页组件 -->
    <el-pagination
      @size-change="handleSizeChange"
      @current-change="handleCurrentChange"
      :current-page.sync="currentPage"
      :page-size="pageSize"
      layout="total, prev, pager, next"
      :total="tableData.length"
    ></el-pagination>
  </el-main>
</template>

<script>
export default {
  data() {
    return {
      //当前页
      currentPage: 1,
      pageSize: 10,
      //搜索表单数据绑定
      searchForm: {
        roleName: ""
      },
      //表格数据
      tableData: [
        {
          date: "2016-05-03",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-02",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-04",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-01",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-08",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-06",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-07",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        }
      ],
      //表格高度  window.innerHeight窗口文档显示高度
      tableHeight: window.innerHeight
    };
  },
  methods: {
    //删除按钮
    deleteRow(index,row){

    },
    //编辑按钮
    editRow(index,row){

    },
    handleSizeChange(val) {
      console.log(`每页 ${val} 条`);
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`);
    }
  },
  //   该钩子函数执行时所有的DOM挂载和渲染都已完成，此时在该钩子函数中进行任何DOM操作都不会有问题
  // 在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，
  // 这个操作都应该放进Vue.nextTick()的回调函数中
  mounted() {
    this.$nextTick(() => {
      this.tableHeight = window.innerHeight - 240; //后面的50：根据需求空出的高度，自行调整
    });
  }
};
</script>

<style lang="scss" scoped>
.searchBtn {
  margin-left: 15px;
}
</style>
```

第15讲 角色分配权限弹框制作及整合ztree

1.1、弹框实现原理:

控制 dialogVisible 为 true或false来控制弹框显示和影藏

1.2、安装 vue-giant-tree

npm i vue-giant-tree --save

1.3、在需要ztree树的页面引入

import tree from "vue-giant-tree";

1.4、使用ztree

1.4.1、注册ztree

```
components: {
   tree
 },
```

1.4.2、配置ztree

```
innerVisible:false, //控制弹框显示
ztreeObj: null,
 setting: {
   check: {
     enable: true
   },
   data: {
     simpleData: {
       enable: true,
       idKey: "id",
       pIdKey: "pid",
       rootPId: "0"
     }
   },
   callback: {
     onCheck: this.ztreeOnCheck
   }
 },
```

setting配置说明:

ztreeObj:当前树对象，树创建成功后返回

check.enabe : 树是否显示 复选框或单选按钮

simpleData.enable: 是否使用简单数据模式

如果设置为 true，必须设置 setting.data.simpleData 内的其他参数: idKey / pIdKey / rootPId，并且让数据满足父子关系。

callback:回调函数

onCheck：树选中时回调，用于获取选中的节点数据

```
ztreeOnCheck() {
     let checked = this.ztreeObj.getCheckedNodes(true);
     this.checkPermissions = checked;
     console.log(checked);
   },
```

1.4.3、使用ztree

https://github.com/tower1229/Vue-Giant-Tree

https://github.com/tower1229/Vue-Giant-Tree/blob/master/src/App.vue

```
<tree
       :nodes="treeDatas"
       :setting="setting"
       @onCheck="ztreeOnCheck"
       @onCreated="handleCreated"
     />
```

说明: nodes 树展示数据列表

setting：树的配置，参照ztree树官方网站 http://www.treejs.cn/v3/api.php

@onCheck 用于捕获 checkbox / radio 被勾选 或 取消勾选的事件回调函数

```
ztreeOnCheck() {
     let checked = this.ztreeObj.getCheckedNodes(true);
     this.checkPermissions = checked;
     console.log(checked);
   },
```

@onCreated 树创建时回调函数

```
handleCreated: function(ztreeObj) {
     console.log("加载树完成");
     this.ztreeObj = ztreeObj;

     console.log(this.ztreeObj);
     //  let firstTree = ztreeObj.getNodes()[0];
     //默认选中第一个
     // ztreeObj.selectNode(firstTree);
     //设置节点全部展开
     ztreeObj.expandAll(true);
     //加载完自动点击第一个，加载右边表格
     // this.setting.callback.onClick(null, firstTree.id, firstTree);
   },
```

1.4.4、点击分配权限按钮，弹出弹框

```
    assignRole(row) {
     this.roldId = row.id;
     this.dialogTitle = '为【'+row.name+'】分配权限';
     this.treeDatas = [{
       "id": 17,
       "pid": 0,
       "name": "系统管理",
       "open": null,
       "checked": true
   }, {
       "id": 18,
       "pid": 17,
       "name": "用户管理",
       "open": null,
       "checked": true
   }, {
       "id": 20,
       "pid": 18,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 21,
       "pid": 18,
       "name": "修改",
       "open": null,
       "checked": true
   }, {
       "id": 22,
       "pid": 18,
       "name": "删除",
       "open": null,
       "checked": true
   }, {
       "id": 23,
       "pid": 17,
       "name": "角色管理",
       "open": null,
       "checked": true
   }, {
       "id": 25,
       "pid": 23,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 26,
       "pid": 23,
       "name": "修改",
       "open": null,
       "checked": true
   }, {
       "id": 27,
       "pid": 23,
       "name": "删除",
       "open": null,
       "checked": true
   }, {
       "id": 28,
       "pid": 17,
       "name": "权限管理",
       "open": null,
       "checked": true
   }, {
       "id": 30,
       "pid": 28,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 31,
       "pid": 28,
       "name": "修改",
       "open": null,
       "checked": true
   }, {
       "id": 32,
       "pid": 28,
       "name": "删除",
       "open": null,
       "checked": true
   }, {
       "id": 33,
       "pid": 17,
       "name": "机构管理",
       "open": null,
       "checked": true
   }, {
       "id": 34,
       "pid": 0,
       "name": "商品管理",
       "open": null,
       "checked": true
   }, {
       "id": 36,
       "pid": 34,
       "name": "分类管理",
       "open": null,
       "checked": true
   }, {
       "id": 37,
       "pid": 34,
       "name": "品牌管理",
       "open": null,
       "checked": true
   }, {
       "id": 38,
       "pid": 36,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 39,
       "pid": 36,
       "name": "编辑",
       "open": null,
       "checked": true
   }, {
       "id": 40,
       "pid": 37,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 41,
       "pid": 37,
       "name": "编辑",
       "open": null,
       "checked": true
   }, {
       "id": 42,
       "pid": 0,
       "name": "系统工具",
       "open": null,
       "checked": true
   }, {
       "id": 43,
       "pid": 42,
       "name": "代码生成",
       "open": null,
       "checked": true
   }, {
       "id": 46,
       "pid": 33,
       "name": "新增",
       "open": null,
       "checked": true
   }, {
       "id": 76,
       "pid": 33,
       "name": "编辑",
       "open": null,
       "checked": true
   }, {
       "id": 77,
       "pid": 42,
       "name": "接口文档",
       "open": null,
       "checked": true
   }, {
       "id": 78,
       "pid": 33,
       "name": "删除",
       "open": null,
       "checked": true
   }, {
       "id": 79,
       "pid": 23,
       "name": "分配权限",
       "open": null,
       "checked": true
   }, {
       "id": 80,
       "pid": 18,
       "name": "分配角色",
       "open": null,
       "checked": true
   }];
     this.innerVisible = true;
```

1.5、弹框代码

```
<el-dialog class="self_dialog" width="25%" :title="dialogTitle"
:visible.sync="innerVisible">
      <tree
        :nodes="treeDatas"
        :setting="setting"
        @onCheck="ztreeOnCheck"
        @onCreated="handleCreated"
      />
      <div slot="footer" class="dialog-footer">
        <el-button @click="innerVisible = false">取 消</el-button>
        <el-button type="primary" @click="saveAssign">确 定</el-button>
      </div>
    </el-dialog>
```

```
.self_dialog {
  display: flex;
  justify-content: center;
  align-items: Center;
  overflow: hidden;
}
.self_dialog /deep/ .el-dialog {
  margin: 0 auto !important;
  height: 90%;
  overflow: hidden;
  display: flex;
  flex-direction: column;
  padding-left: 15px;
}
.self_dialog /deep/ .el-dialog .el-dialog__body {
  padding-top: 5px !important;
  overflow: hidden;
  overflow-y: auto;
  margin-bottom: 40px;
}
.self_dialog /deep/ .el-dialog .el-dialog__footer{
  left: 40%;
  bottom: 0;
  position: absolute;
}
```

第16讲 组织管理列表布局

1.1、采用左右侧布局列表

使用组件： 表格组件　　　分页组件

```
<el-container>
      <el-aside width="200px" style="border-right: 1px solid #d2d6de;border-
left:none;">菜单</el-aside>
      <el-main>
        <el-form size="mini" :model="searchForm" ref="form" label-width="80px">
          <el-row>
            <el-col :span="5">
              <el-form-item label="名称">
                <el-input v-model="searchForm.depaName"></el-input>
              </el-form-item>
            </el-col>
            <el-col :span="5">
              <el-form-item label="电话">
                <el-input v-model="searchForm.deptPhone"></el-input>
              </el-form-item>
            </el-col>
            <el-button style="margin-left:20px;" size="mini" type="primary" icon="el-
icon-search">查询</el-button>
            <el-button size="mini" type="primary" icon="el-icon-plus">新增</el-button>
          </el-row>
        </el-form>
        <el-table size="mini" :data="tableData" :height="tableHeight" border
style="width: 100%">
          <el-table-column prop="date" label="日期" width="180"></el-table-column>
          <el-table-column prop="name" label="姓名" width="180"></el-table-column>
          <el-table-column prop="address" label="地址"></el-table-column>
        </el-table>
        <el-pagination
```

```
          @size-change="handleSizeChange"
          @current-change="handleCurrentChange"
          :current-page.sync="currentPage"
          :page-size="100"
          layout="total, prev, pager, next"
          :total="1000"
        ></el-pagination>
      </el-main>
    </el-container>
```

```
export default {
  //计算表格高度
  mounted() {
    this.$nextTick(() => {
      this.tableHeight = window.innerHeight - 230; //后面的50：根据需求空出的高度，自行调整
    });
  },
  data() {
    return {
      //当前页树
      currentPage: 1,
      //表格高度
      tableHeight: 0,
      //搜索数据绑定
      searchForm: {
        depaName: "",
        deptPhone: ""
      },
      tableData: [
        {
          date: "2016-05-03",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-02",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-04",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-01",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-08",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-06",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        },
        {
          date: "2016-05-07",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        }
      ]
    };
  },
  methods: {
       handleSizeChange(val) {
        console.log(`每页 ${val} 条`);
      },
      handleCurrentChange(val) {
        console.log(`当前页: ${val}`);
      }
  }
};
</script>
```

第17讲 机构管理组织树讲解

1.1、界面布局：

采用左右布局方式，左侧放ztree树，右侧放部门列表

```
 <el-container style="height: 100%; border: 1px solid #eee">
     <el-aside width="200px" style="border-right: 1px solid #d2d6de;border-left:none;">
        <div style="padding-top:5px;padding-left:5px;">
          <i class="el-icon-menu"></i>
          <span style="padding-left:3px;">组织机构</span>
        </div>
        <tree
          style="padding-left: 0px;padding-top: 10px;"
          :nodes="nodes"
          :setting="setting"
          @onCreated="handleCreated"
        />
      </el-aside>
     <el-main>
      <!--内容展示区-->
      内容展示
     </el-main>
 <el-container>
```

1.2、左侧部门树配置

1.2.1、引入ztree组件　import tree from "vue-giant-tree";

1.2.2、注册组件

```
name: "departmentList",
 components: {
   tree
 },
```

1.2.3、配置树setting

```
//树插件配置
     ztreeObj: null,
     setting: {
       view: {
         showLine: true,
         showIcon: false,
         fontCss: { "font-size": "12px", color: "#333" }
       },
       //设置这里会显示复选框
       // check: {
       //   enable: true
       // },
       data: {
         simpleData: {
           enable: true,
           idKey: "id",
           pIdKey: "pid",
           rootPId: "0"
         }
       },
       callback: {
         onClick: this.ztreeOnClick
       }
     },
     nodes: [{
       "id": "1000000362292826",
       "pid": "1000001251633881",
       "likeId": "0,100000177618509910000012516338811000000362292826",
       "parentName": "销售部门",
       "manager": null,
       "name": "销售1",
       "deptCode": "",
       "deptAddress": "",
       "deptPhone": "",
       "orderNum": 0
   }, {
       "id": "1000001251633881",
       "pid": "1000001776185099",
       "likeId": "0,10000017761850991000001251633881",
       "parentName": "秘咖科技有限公司",
       "manager": null,
       "name": "销售部门",
       "deptCode": null,
       "deptAddress": null,
       "deptPhone": null,
       "orderNum": null
   }, {
       "id": "1000001341234088",
       "pid": "1000001776185099",
       "likeId": "0,1000001776185099",
       "parentName": "秘咖网络科技有限公司",
       "manager": null,
       "name": "人才管理部1",
       "deptCode": "RCGL",
       "deptAddress": "",
       "deptPhone": "",
       "orderNum": 0
   }, {
       "id": "1000001620535597",
       "pid": "1000001776185099",
       "likeId": "0,10000017761850991000001620535597",
       "parentName": "秘咖网络科技有限公司",
       "manager": null,
       "name": "软件研发部",
       "deptCode": null,
       "deptAddress": null,
       "deptPhone": null,
       "orderNum": null
   }, {
       "id": "1000001776185099",
       "pid": "0",
       "likeId": "0,1000001776185099",
       "parentName": "顶级部门",
       "manager": null,
       "name": "秘咖网络科技有限公司",
       "deptCode": null,
       "deptAddress": null,
       "deptPhone": null,
       "orderNum": null
   }, {
       "id": "1000002097176073",
       "pid": "1000001776185099",
       "likeId": "0,10000017761850991000002097176073",
       "parentName": "秘咖网络科技有限公司",
       "manager": "464156",
       "name": "售后服务部",
       "deptCode": "SHFWB",
       "deptAddress": "昆明",
       "deptPhone": "18687171906",
       "orderNum": null
   }],
```

1.3、写树执行事件

1.3.1、树创建成功回调事件

```
handleCreated: function(ztreeObj) {
     this.ztreeObj = ztreeObj;
     let firstTree = this.ztreeObj.getNodes()[0];
     //默认选中第一个
     this.ztreeObj.selectNode(firstTree);
     //设置节点全部展开
     ztreeObj.expandAll(true);
     //加载完自动点击第一个，加载右边表格
     if (firstTree) {
       //此处需要判断，否则会报错
       this.setting.callback.onClick(null, firstTree.id, firstTree);
     }
   },
```

1.3.2、左侧部门树点击事件

点击左侧部门树，加载对应部门下的数据

```
 // 树点击事件
    ztreeOnClick: function(evt, treeId, treeNode) {
      console.log(treeNode);
      //此处根据点中部门树id查询下级部门
    },
```

第18讲 新增部门布局讲解

1.1、实现原理：

控制弹框显示和影藏

1.2、上级部门选择：

点击上级部门，弹出上级部门的树形弹框选择

注意事项：上级部门弹框 属于嵌套弹框，内层对话框 需要 添加 append-to-body 属性

1.3、新增弹框代码

```
<!--新增部门弹框-->
    <el-dialog :title="deptDialogTitle" :visible.sync="dialogVisible" width="30%">
      <el-form size="mini" :model="addForm" ref="addForm" label-width="80px">
        <el-form-item label="上级部门">
          <el-input v-model="addForm.parentName"></el-input>
        </el-form-item>
        <el-form-item label="部门名称">
          <el-input v-model="addForm.name"></el-input>
        </el-form-item>
        <el-form-item label="部门编码">
          <el-input v-model="addForm.deptCode"></el-input>
        </el-form-item>
        <el-form-item label="部门电话">
          <el-input v-model="addForm.deptPhone"></el-input>
        </el-form-item>
        <el-form-item label="部门地址">
          <el-input v-model="addForm.deptAddress"></el-input>
        </el-form-item>
        <el-form-item label="序号">
          <el-input-number v-model="addForm.orderNum" placeholder></el-input-number>
        </el-form-item>
      </el-form>
      <span slot="footer" class="dialog-footer">
        <el-button @click="dialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="dialogVisible = false">确 定</el-button>
      </span>
    </el-dialog>
```

```
//新增部门数据绑定
     addForm: {
       id: "",
       pid: "",
       parentName: "",
       name: "",
       deptCode: "",
       deptPhone: "",
       deptAddress: "",
       orderNum: ""
     },
```

新增按钮点击事件

```
//新增部门
   addDept() {
       this.deptDialogTitle = '新增部门';
     this.dialogVisible = true;
   },
```

弹框样式

```
.el-dialog__wrapper /deep/ .el-dialog__body {
  padding-top: 5px !important;
}
```

1.4、上级部门弹框

注意： 需要添加 append-to-body 属性

1.4.1、上级部门输入框添加点击事件

```
@click.native="selectDept()"
```

```
this.parentNodes = [
       {
         id: "0",
         pid: "-1",
         likeId: "0,",
         parentName: null,
         manager: null,
         name: "顶级部门",
         deptCode: null,
         deptAddress: null,
         deptPhone: null,
         orderNum: null
       },
       {
         id: "1000000362292826",
         pid: "1000001251633881",
         likeId: "0,100000177618509910000012516338811000000362292826",
         parentName: "销售部门",
         manager: null,
         name: "销售1",
         deptCode: "",
          deptAddress: "",
          deptPhone: "",
          orderNum: 0
        },
        {
          id: "1000001251633881",
          pid: "1000001776185099",
          likeId: "0,10000017761850991000001251633881",
          parentName: "秘咖科技有限公司",
          manager: null,
          name: "销售部门",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000001341234088",
          pid: "1000001776185099",
          likeId: "0,1000001776185099",
          parentName: "秘咖网络科技有限公司",
          manager: null,
          name: "人才管理部1",
          deptCode: "RCGL",
          deptAddress: "",
          deptPhone: "",
          orderNum: 0
        },
        {
          id: "1000001620535597",
          pid: "1000001776185099",
          likeId: "0,10000017761850991000001620535597",
          parentName: "秘咖网络科技有限公司",
          manager: null,
          name: "软件研发部",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000001776185099",
          pid: "0",
          likeId: "0,1000001776185099",
          parentName: "顶级部门",
          manager: null,
          name: "秘咖网络科技有限公司",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000002097176073",
          pid: "1000001776185099",
          likeId: "0,10000017761850991000002097176073",
          parentName: "秘咖网络科技有限公司",
          manager: "464156",
          name: "售后服务部",
          deptCode: "SHFWB",
          deptAddress: "昆明",
         deptPhone: "18687171906",
         orderNum: null
       }
     ];
     this.deptDialogTitle = "新增部门";
     this.dialogVisible = true;
```

1.4.2、data中添加

```
    //控制上级部门显示和影藏
     parentViale: false,
     parentNodes: [], //上级部门树数据
```

1.4.3、ztree配置

```
parentZtreeObj: null,
parentNodes: [], //上级部门树数据
//上级部门树配置
parentSetting: {
       view: {
         showLine: true,
         showIcon: false,
         fontCss: { "font-size": "12px", color: "#333" }
       },
       //设置这里会显示复选框
       // check: {
       //   enable: true
       // },
       data: {
         simpleData: {
           enable: true,
           idKey: "id",
           pIdKey: "pid",
           rootPId: "0"
         }
       },
       callback: {
         onClick: this.ztreeParentOnClick
       }
     },
```

1.4.4、tree显示

```
<tree :nodes="parentNodes" :setting="parentSeeting"></tree>
```

1.4.5、点击事件

```
 //上级部门树点击事件
   ztreeParentOnClick(evt, treeId, treeNode){
       console.log(treeNode.name);
       this.addForm.parentName = treeNode.name;
       console.log(evt,treeId,treeNode);
   },
```

第19讲 用户管理列表布局讲解

1.1、布局方式：

左右布局方式，左边部门树，右边用户列表

1.2、相关布局参照部门管理

第20讲 新增用户和分配角色布局讲解

1.1、新增用户布局实现

弹框、输入框、ztree实现上级部门选择

1.2、实现代码

1.2.1、弹框实现

```
 <el-dialog :title="addTitle" :visible.sync="dialogVisible" width="30%">
      <el-form size="mini" :model="form" ref="form" label-width="80px">
        <el-form-item label="部门">
          <el-input @click.native="selectDept" v-model="userForm.deptName"></el-input>
        </el-form-item>
        <el-form-item label="姓名">
          <el-input v-model="userForm.loginname"></el-input>
        </el-form-item>
        <el-form-item label="性别">
          <el-input v-model="userForm.sex"></el-input>
        </el-form-item>
        <el-form-item label="电话">
          <el-input v-model="userForm.phone"></el-input>
        </el-form-item>
        <el-form-item label="登录名">
          <el-input v-model="userForm.username"></el-input>
        </el-form-item>
        <el-form-item label="密码">
          <el-input v-model="userForm.password"></el-input>
        </el-form-item>
      </el-form>
      <span slot="footer" class="dialog-footer">
        <el-button @click="dialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="dialogVisible = false">确 定</el-button>
      </span>
    </el-dialog>
```

1.2.2、ztree上级部门实现

```
  <el-dialog title="选择部门" :visible.sync="parentDialogVisible" width="25%">
      <tree :nodes="parentNodes" @onCreated='createdParent' :setting="parentSetting"></tree>
      <span slot="footer" class="dialog-footer">
        <el-button @click="parentDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="parentDialogVisible = false">确 定</el-button>
      </span>
    </el-dialog>
```

```
//选择上级部门
    selectDept(){
      this.parentNodes = [
        {
          id: "1000000362292826",
          pid: "1000001251633881",
          likeId: "0,100000177618509910000012516338811000000362292826",
          parentName: "销售部门",
          manager: null,
          name: "销售1",
          deptCode: "",
          deptAddress: "",
          deptPhone: "",
          orderNum: 0
        },
        {
          id: "1000001251633881",
          pid: "1000001776185099",
          likeId: "0,10000017761850991000001251633881",
          parentName: "秘咖科技有限公司",
          manager: null,
          name: "销售部门",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000001341234088",
          pid: "1000001776185099",
          likeId: "0,1000001776185099",
          parentName: "秘咖网络科技有限公司",
          manager: null,
          name: "人才管理部1",
          deptCode: "RCGL",
          deptAddress: "",
          deptPhone: "",
          orderNum: 0
        },
        {
          id: "1000001620535597",
          pid: "1000001776185099",
          likeId: "0,10000017761850991000001620535597",
          parentName: "秘咖网络科技有限公司",
          manager: null,
          name: "软件研发部",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000001776185099",
          pid: "0",
          likeId: "0,1000001776185099",
          parentName: "顶级部门",
          manager: null,
          name: "秘咖网络科技有限公司",
          deptCode: null,
          deptAddress: null,
          deptPhone: null,
          orderNum: null
        },
        {
          id: "1000002097176073",
          pid: "1000001776185099",
          likeId: "0,100000177618509910000002097176073",
          parentName: "秘咖网络科技有限公司",
          manager: "464156",
          name: "售后服务部",
          deptCode: "SHFWB",
          deptAddress: "昆明",
          deptPhone: "18687171906",
          orderNum: null
        }
      ],
      this.parentDialogVisible = true;
    },
    //上级部门树创建成功回调
    createdParent(treePree) {
      this.parentZtreeObj = treePree;
      treePree.expandAll(true);
    },
    //上级部门数选择点击事件
    ztreeParentOnClick(event, treeId, treeNode) {
      this.userForm.deptId = treeId;
      this.userForm.deptName = treeNode.name;
      console.log(event);
      console.log(treeId);
      console.log(treeNode);
    },
    //新增用户
    addUser() {
      this.addTitle = "新增用户";
      this.dialogVisible = true;
    },
```

```
      parentDialogVisible:false,
      parentZtreeObj: null,
      parentNodes: [], //上级部门树数据
      //上级部门树配置
      parentSetting: {
        view: {
          showLine: true,
          showIcon: false,
          fontCss: { "font-size": "12px", color: "#333" }
        },
        //设置这里会显示复选框
        // check: {
        //   enable: true
        // },
        data: {
          simpleData: {
            enable: true,
            idKey: "id",
            pIdKey: "pid",
            rootPId: "0"
          }
        },
        callback: {
          onClick: this.ztreeParentOnClick
```

```
    }
   },
   //新增用户数据绑定
   userForm: {
    username: "",
    sex: "",
    phone: "",
    loginname: "",
    password: "",
    deptId: "",
    deptName: ""
   },
   //新增弹框显示或影藏控制
   dialogVisible: false,
   //新增弹框标题
   addTitle: "",
```

1.3、分配角色布局

1.3.1、实现方式：对话框、table方式

1.3.2、代码实现：

```
 <!-- 分配角色对话框 -->
  <el-dialog  class="roleClass" title="分配角色" :visible.sync="roleDialogVisible"
width="30%">
   <el-table  @current-change="selectRoleRow"  highlight-current-row border
height="250"  :data="roleTableData" style="width: 100%">
    <el-table-column prop="id" label="序号" width="180"></el-table-column>
    <el-table-column prop="roleName" label="角色名称" ></el-table-column>
   </el-table>
   <span slot="footer" class="dialog-footer">
    <el-button @click="roleDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="roleDialogVisible = false">确 定</el-button>
   </span>
  </el-dialog>
```

```
  //角色列表数据
   roleTableData:[],
   //分配角色对话框显示
   roleDialogVisible: false,


 //选中角色
  selectRoleRow(row){
   console.log(row)
  },
  //分配角色
  assignRole() {
   this.roleTableData =[
    {id:'1',roleName:'超级管理员'},
    {id:'2',roleName:'系统管理员'},
    {id:'3',roleName:'财务管理员'},
   ]
   this.roleDialogVisible = true;
  },
.roleClass /deep/ .el-table__body tr.current-row > td {
  background: #409eff !important;
  color: #fff;
}
```

第21讲 菜单管理列表讲解

1.1、列表布局

1.1.1、组件

1.1.2、组件添加

1.1.3、代码实现

```
 <el-table
      :data="menuList"
      style="width: 100%;"
      row-key="id"
      border
      :tree-props="{children: 'children'}"
    >
      <el-table-column prop="label" label="名称" sortable width="180"></el-table-column>
      <el-table-column prop="icon" label="图标" sortable width="180"></el-table-column>
      <el-table-column prop="type" label="类型"></el-table-column>
      <el-table-column prop="url" label="菜单URL"></el-table-column>
      <el-table-column prop="path" label="路由地址"></el-table-column>
      <el-table-column prop="code" label="权限标识"></el-table-column>
      <el-table-column prop="orderNum" label="序号"></el-table-column>
    </el-table>
```

```
  menuList: [
        {
          id: 17,
          parentId: 0,
          parentName: "顶级菜单",
          label: "系统管理",
          code: "sys:manage",
          path: "/system",
          name: null,
          url: null,
          orderNum: 1,
          type: "0",
          icon: "el-icon-document",
          remark: null,
          createTime: "2023-08-08T03:11:11.000+0000",
          updateTime: "2023-08-09T07:26:28.000+0000",
          isHome: 0,
          children: [
            {
              id: 33,
              parentId: 17,
              parentName: "系统管理",
              label: "机构管理",
              code: "sys:dept",
              path: "/departmentList",
              name: "departmentList",
              url: "/system/Department/DepartmentList",
              orderNum: 2,
              type: "1",
              icon: "el-icon-copy-document",
              remark: "机构管理",
              createTime: "2020-04-12T14:58:29.000+0000",
              updateTime: "2020-04-08T09:12:19.000+0000",
              isHome: 0,
              children: [
                {
                  id: 46,
                  parentId: 33,
                  parentName: null,
                  label: "新增",
                  code: "sys:addDepartment",
                  path: "",
                  name: "",
                  url: null,
                  orderNum: 0,
                  type: "2",
                  icon: "",
                  remark: null,
                  createTime: "2020-04-12T11:58:48.000+0000",
                  updateTime: "2020-04-12T11:58:48.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 76,
                  parentId: 33,
                  parentName: null,
                  label: "编辑",
                  code: "sys:editDept",
                  path: "",
                  name: "",
                  url: null,
                  orderNum: 1,
                  type: "2",
                  icon: "",
                  remark: null,
                  createTime: "2020-04-12T12:42:20.000+0000",
                  updateTime: "2020-04-12T12:42:20.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 78,
                  parentId: 33,
                  parentName: "机构管理",
                  label: "删除",
                  code: "sys:deleteDept",
                  path: "",
                  name: "",
                  url: "",
                  orderNum: 3,
                  type: "2",
                  icon: "",
                  remark: null,
                  createTime: "2020-04-18T02:25:55.000+0000",
                  updateTime: "2020-04-18T02:25:55.000+0000",
                  isHome: 0,
                  children: []
                }
              ]
            },
            {
              id: 18,
              parentId: 17,
              parentName: null,
              label: "用户管理",
              code: "sys:user",
              path: "/userList",
              name: "userList",
              url: "/system/User/UserList",
              orderNum: 3,
              type: "1",
              icon: "el-icon-s-custom",
              remark: null,
              createTime: "2023-08-08T03:11:11.000+0000",
              updateTime: "2023-08-09T07:26:28.000+0000",
              isHome: 0,
              children: [
                {
                  id: 20,
                  parentId: 18,
                  parentName: null,
                  label: "新增",
                  code: "sys:user:add",
                  path: null,
                  name: null,
                  url: "",
                  orderNum: null,
                  type: "2",
                  icon: "",
                  remark: "新增用户",
                  createTime: "2023-08-08T03:11:11.000+0000",
                  updateTime: "2023-08-09T07:26:28.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 21,
                  parentId: 18,
                  parentName: null,
                  label: "修改",
                  code: "sys:user:edit",
                  path: null,
                  name: null,
                  url: "",
                  orderNum: null,
                  type: "2",
                  icon: "",
                  remark: "修改用户",
                  createTime: "2023-08-08T03:11:11.000+0000",
                  updateTime: "2023-08-09T07:26:28.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 22,
                  parentId: 18,
                  parentName: null,
                  label: "删除",
                  code: "sys:user:delete",
                  path: null,
                  name: null,
                  url: "",
                  orderNum: null,
                  type: "2",
                  icon: "",
                  remark: "删除用户",
                  createTime: "2023-08-08T03:11:11.000+0000",
                  updateTime: "2023-08-09T07:26:28.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 80,
                  parentId: 18,
                  parentName: "用户管理",
                  label: "分配角色",
                  code: "sys:user:assign",
                  path: "",
                  name: "",
                  url: "",
                  orderNum: 0,
                  type: "2",
                  icon: "",
                  remark: null,
                  createTime: "2020-04-18T02:50:14.000+0000",
                  updateTime: "2020-04-18T02:50:14.000+0000",
                  isHome: 0,
                  children: []
                }
              ]
            },
            {
              id: 23,
              parentId: 17,
              parentName: null,
              label: "角色管理",
              code: "sys:role",
              path: "/roleList",
              name: "roleList",
              url: "/system/Role/RoleList",
              orderNum: 4,
              type: "1",
              icon: "el-icon-rank",
              remark: null,
              createTime: "2023-08-08T03:11:11.000+0000",
              updateTime: "2023-08-09T07:26:28.000+0000",
              isHome: 0,
              children: [
                {
                  id: 25,
                  parentId: 23,
                  parentName: null,
                  label: "新增",
                  code: "sys:role:add",
                  path: null,
                  name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: "",
                      remark: "新增角色",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 26,
                      parentId: 23,
                      parentName: null,
                      label: "修改",
                      code: "sys:role:edit",
                      path: null,
                      name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: "",
                      remark: "修改角色",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 27,
                      parentId: 23,
                      parentName: null,
                      label: "删除",
                      code: "sys:role:delete",
                      path: null,
                      name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: "",
                      remark: "删除角色",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 79,
                      parentId: 23,
                      parentName: "角色管理",
                      label: "分配权限",
                      code: "sys:role:assign",
                      path: "",
                      name: "",
                      url: "",
                      orderNum: 0,
                      type: "2",
                      icon: "",
                      remark: null,
                      createTime: "2020-04-18T02:31:05.000+0000",
                      updateTime: "2020-04-18T02:31:05.000+0000",
                      isHome: 0,
                      children: []
                    }
                  ]
                },
                {
                  id: 28,
                  parentId: 17,
                  parentName: null,
                  label: "权限管理",
                  code: "sys:menu",
                  path: "/menuList",
                  name: "menuList",
                  url: "/system/Menu/MenuList",
                  orderNum: 5,
                  type: "1",
                  icon: "el-icon-menu",
                  remark: null,
                  createTime: "2023-08-08T03:11:11.000+0000",
                  updateTime: "2023-08-09T07:26:28.000+0000",
                  isHome: 0,
                  children: [
                    {
                      id: 30,
                      parentId: 28,
                      parentName: null,
                      label: "新增",
                      code: "sys:menu:add",
                      path: null,
                      name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: null,
                      remark: "新增权限",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 31,
                      parentId: 28,
                      parentName: null,
                      label: "修改",
                      code: "sys:menu:edit",
                      path: null,
                      name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: null,
                      remark: "修改权限",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 32,
                      parentId: 28,
                      parentName: null,
                      label: "删除",
                      code: "sys:menu:delete",
                      path: null,
                      name: null,
                      url: "",
                      orderNum: null,
                      type: "2",
                      icon: "",
                      remark: "删除权限",
                      createTime: "2023-08-08T03:11:11.000+0000",
                      updateTime: "2023-08-09T07:26:28.000+0000",
                      isHome: 0,
                      children: []
                    }
                  ]
                }
              ]
            },
            {
              id: 34,
              parentId: 0,
              parentName: "顶级菜单",
              label: "商品管理",
              code: "sys:goods",
              path: "/goods",
              name: "",
              url: null,
              orderNum: 2,
              type: "0",
              icon: "el-icon-picture",
              remark: null,
              createTime: "2020-04-12T14:49:47.000+0000",
              updateTime: "2020-04-12T09:22:03.000+0000",
              isHome: 0,
              children: [
                {
                  id: 36,
                  parentId: 34,
                  parentName: "商品管理",
                  label: "分类管理",
                  code: "sys:goodsCategory",
                  path: "/goodCategory",
                  name: "goodCategory",
                  url: "/goods/goodsCategory/goodsCategoryList",
                  orderNum: 1,
                  type: "1",
                  icon: "el-icon-s-data",
                  remark: null,
                  createTime: "2020-04-12T14:54:32.000+0000",
                  updateTime: "2020-04-12T09:26:30.000+0000",
                  isHome: 0,
                  children: [
                    {
                      id: 38,
                      parentId: 36,
                      parentName: null,
                      label: "新增",
                      code: "sys:addGoodsCategory",
                      path: "",
                      name: "",
                      url: null,
                      orderNum: 0,
                      type: "2",
                      icon: "",
                      remark: null,
                      createTime: "2020-04-12T09:33:58.000+0000",
                      updateTime: "2020-04-12T09:33:58.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 39,
                      parentId: 36,
                      parentName: null,
                      label: "编辑",
                      code: "sys:editGoodsCategory",
                      path: "",
                      name: "",
                      url: null,
                      orderNum: 1,
                      type: "2",
                      icon: "",
                      remark: null,
                      createTime: "2020-04-12T09:35:30.000+0000",
                      updateTime: "2020-04-12T09:35:30.000+0000",
                      isHome: 0,
                      children: []
                    }
                  ]
                },
                {
                  id: 37,
                  parentId: 34,
                  parentName: null,
                  label: "品牌管理",
                  code: "sys:goodsBrand",
                  path: "/goodsBrand",
                  name: "goodsBrand",
                  url: "/goods/goodsBrand/goodsBrandList",
                  orderNum: 2,
                  type: "1",
                  icon: "el-icon-tickets",
                  remark: null,
                  createTime: "2020-04-12T09:32:04.000+0000",
                  updateTime: "2020-04-12T09:32:04.000+0000",
                  isHome: 0,
                  children: [
                    {
                      id: 40,
                      parentId: 37,
                      parentName: null,
                      label: "新增",
                      code: "sys:addGoodsBrand",
                      path: "",
                      name: "",
                      url: null,
                      orderNum: 0,
                      type: "2",
                      icon: "",
                      remark: null,
                      createTime: "2020-04-12T09:36:14.000+0000",
                      updateTime: "2020-04-12T09:36:14.000+0000",
                      isHome: 0,
                      children: []
                    },
                    {
                      id: 41,
                      parentId: 37,
                      parentName: null,
                      label: "编辑",
                      code: "sys:editGoodsBrand",
                      path: "",
                      name: "",
                      url: null,
                      orderNum: 1,
                      type: "2",
                      icon: "",
                      remark: null,
                      createTime: "2020-04-12T09:36:46.000+0000",
                      updateTime: "2020-04-12T09:36:46.000+0000",
                      isHome: 0,
                      children: []
                    }
                  ]
                }
              ]
            },
            {
              id: 42,
              parentId: 0,
              parentName: "顶级菜单",
              label: "系统工具",
              code: "sys:systenConfig",
              path: "/systenConfig",
              name: "",
              url: null,
              orderNum: 3,
              type: "0",
              icon: "el-icon-receiving",
              remark: null,
              createTime: "2020-04-12T14:50:03.000+0000",
              updateTime: "2020-04-12T09:40:41.000+0000",
              isHome: 0,
              children: [
                {
                  id: 43,
                  parentId: 42,
                  parentName: "系统工具",
                  label: "代码生成",
                  code: "sys:systemCode",
                  path: "/systemCode",
                  name: "systemCode",
                  url: "/system/config/code",
                  orderNum: 0,
                  type: "1",
                  icon: "el-icon-files",
                  remark: null,
                  createTime: "2020-04-16T04:44:42.000+0000",
                  updateTime: "2020-04-12T09:44:06.000+0000",
                  isHome: 0,
                  children: []
                },
                {
                  id: 77,
                  parentId: 42,
                  parentName: "系统工具",
                  label: "接口文档",
                  code: "sys:document",
                  path: "/document",
                  name: "document",
                  url: "/system/config/systemDocument",
                  orderNum: 0,
                  type: "1",
                  icon: "el-icon-s-operation",
                  remark: null,
                  createTime: "2020-04-13T03:31:45.000+0000",
                  updateTime: "2020-04-13T03:31:45.000+0000",
                  isHome: 0,
                  children: []
                }
              ]
            }
          ]
```

1.2、图标实现

```
    <el-table-column header-align="center" align="center" label="图标">
            <template slot-scope="scope">
              <i :class="scope.row.icon || ''"></i>
            </template>
          </el-table-column>
```

1.3、类型实现

```
   <el-table-column prop="type" header-align="center" align="center" label="类型">
            <template slot-scope="scope">
              <el-tag v-if="scope.row.type === '0'" size="small">目录</el-tag>
              <el-tag v-else-if="scope.row.type === '1'" size="small" type="success">菜
单</el-tag>
              <el-tag v-else-if="scope.row.type === '2'" size="small" type="info">按钮
</el-tag>
            </template>
          </el-table-column>
```

第22讲 新增菜单和搜索布局讲解

1.1、搜索布局

代码实现

```
   <el-form
        style="margin-top:20px;"
        size="mini"
        :model="searchForm"
        ref="searchForm"
        label-width="80px"
      >
        <el-row>
          <el-col :span="5">
            <el-form-item label="名称">
              <el-input v-model="searchForm.name"></el-input>
            </el-form-item>
          </el-col>
          <el-button
            size="mini"
            style="margin-left:20px;"
            icon="el-icon-search"
            type="primary"
            @click="searBtn"
          >搜索</el-button>
          <el-button size="mini" icon="el-icon-plus" type="primary" @click="addBtn">新增</el-button>
        </el-row>
      </el-form>
```

```
 addBtn() {
      this.addTitle = "新增权限";
      this.dialogVisible = true;
    },
    searBtn() {}
```

1.2、新增菜单布局

```
 <!-- 新增权限弹框 -->
    <el-dialog
      :title="addTitle"
      :visible.sync="dialogVisible"
      width="40%"
      :before-close="handleClose"
    >
      <el-form :inline="true" size="mini" :model="addFrom" ref="addFrom" label-width="80px">
        <el-row>
          <el-col :span="24">
            <el-form-item label="菜单类型">
              <el-radio-group v-model="addFrom.type">
                <el-radio :label="0">目录</el-radio>
                <el-radio :label="1">菜单</el-radio>
                <el-radio :label="2">按钮</el-radio>
              </el-radio-group>
            </el-form-item>
          </el-col>
        </el-row>
        <el-form-item label="上级菜单">
          <el-input v-model="addFrom.parentName"></el-input>
        </el-form-item>
        <el-form-item label="菜单名称">
          <el-input v-model="addFrom.label"></el-input>
        </el-form-item>
        <el-form-item v-if="addFrom.type != '2'" label="菜单图标">
          <el-input v-model="addFrom.icon"></el-input>
        </el-form-item>
        <el-form-item v-if="addFrom.type == '1'" label="路由名称">
          <el-input v-model="addFrom.name"></el-input>
        </el-form-item>
        <el-form-item v-if="addFrom.type != '2'" label="路由地址">
          <el-input v-model="addFrom.path"></el-input>
        </el-form-item>
        <el-form-item v-if="addFrom.type == '1'" label="组件路径">
          <el-input v-model="addFrom.url"></el-input>
        </el-form-item>
        <el-form-item label="权限标识">
          <el-input v-model="addFrom.code"></el-input>
        </el-form-item>
        <el-form-item label="显示序号">
          <el-input-number v-model="addFrom.orderNum"></el-input-number>
        </el-form-item>
      </el-form>

      <span slot="footer" class="dialog-footer">
        <el-button @click="dialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="dialogVisible = false">确 定</el-button>
      </span>
    </el-dialog>
```

```
     addTitle: "",
      addFrom: {
        id: "", //编辑id
        label: "",
        name: "",
        type: 0,
        parentId: "",
        orderNum: "",
        parentName: "",
        path: "",
        code: "",
        icon: ""
      },
      //新增权限弹框
      dialogVisible: false,
```

1.3、上级菜单

1.3.1、引入tree组件

```
import tree from "vue-giant-tree";

components: {
    tree
  },
```

1.3.2、配置树

```
 //控制上级部门弹框显示
      parentDialogVisible: false,
      //上级树陪
      parentZtreeObj: null,
      parentNodes: [], //上级部门树数据
      //上级部门树配置
      parentSetting: {
        view: {
          showLine: true,
          showIcon: false,
          fontCss: { "font-size": "12px", color: "#333" }
        },
        //设置这里会显示复选框
        // check: {
        //   enable: true
        // },
        data: {
          simpleData: {
            enable: true,
            idKey: "id",
            pIdKey: "pid",
            rootPId: "0"
          }
        },
        callback: {
          onClick: this.ztreeParentOnClick
        }
      },
```

1.3.3、事件

```
//上级部门树点击事件
   ztreeParentOnClick(evt, treeId, treeNode) {
     this.addForm.parentName = treeNode.name;
     this.addForm.pid = treeNode.id;
     console.log(evt);
     console.log(treeId);
     console.log(treeNode);
   },
```

1.3.4、使用

```
<!-- 选择上级菜单树弹框 -->
   <el-dialog width="25%" title="上级菜单" :visible.sync="innerVisible" append-to-body>
     <tree :nodes="nodes" :setting="setting" />
     <div slot="footer" class="dialog-footer">
       <el-button @click="innerVisible = false">取 消</el-button>
       <el-button type="primary" @click="getCheckedNodes">确 定</el-button>
     </div>
   </el-dialog>
```

```
nodes: [
        {
          id: 0,
          pid: -1,
          name: "顶级菜单",
          open: true,
          checked: false
        },
        {
          id: 17,
          pid: 0,
          name: "系统管理",
          open: true,
          checked: false
        },
        {
          id: 18,
          pid: 17,
          name: "用户管理",
          open: true,
          checked: false
        },
        {
          id: 23,
          pid: 17,
          name: "角色管理",
          open: true,
          checked: false
        },
        {
          id: 28,
          pid: 17,
          name: "权限管理",
          open: true,
          checked: false
        },
        {
          id: 33,
          pid: 17,
          name: "机构管理",
          open: true,
          checked: false
        },
        {
          id: 34,
          pid: 0,
          name: "商品管理",
          open: true,
          checked: false
        },
        {
          id: 36,
          pid: 34,
          name: "分类管理",
          open: true,
          checked: false
        },
        {
          id: 37,
          pid: 34,
          name: "品牌管理",
          open: true,
          checked: false
        },
        {
          id: 42,
          pid: 0,
          name: "系统工具",
          open: true,
          checked: false
        },
        {
          id: 43,
          pid: 42,
          name: "代码生成",
          open: true,
          checked: false
        },
        {
          id: 77,
          pid: 42,
          name: "接口文档",
          open: true,
          checked: false
        }
      ],
```

第23讲 Spring Security 简介

1.1、Spring Security简介

1、什么是Spring Security

Spring Security是Spring提供的一个安全框架，提供认证(（Authentication)和授权(Authorization)功能，核心技术使用了servlet、Ioc和Aop。

2、什么是认证

认证简单的说就是登录，当用户访问系统时，需要到数据库查看，有没有这个用户，有用户了才允许进入系统。

3、什么是授权

授权指的就是用户拥有哪些权限，如用户可以操作的菜单、按钮、数据等权限。

1.2、Spring Security和Shiro选择

1.2.1、shiro特点

1、Apache的强大灵活的开源安全框架，简单易用。

2、可以提供认证、授权、企业会话管理、安全加密、缓存管理等功能，可以非常快的完成项目中权限管理模块的开发。

3、简单灵活，可以脱离Spirng,权限控制粒度较粗;

1.2.2、Spring Security特点

1、Spring Security上手比shiro复杂，但是功能比Shiro更强大。

2、Spring Security是Spring家族，整合更加方便。如果是spring boot项目，推荐使用Spring Security。

3、Spring Security 社区资源相对比 Shiro 更加丰富;

4、Spring Security对Oauth、OpenID也有支持,Shiro则需要自己手动实现。而且Spring Security的权限细粒度更高。

第24讲 后端模块化项目搭建

1.1项目依赖版本号**

JDk1.8及以上、Maven3.5以上、MySq5.7以上

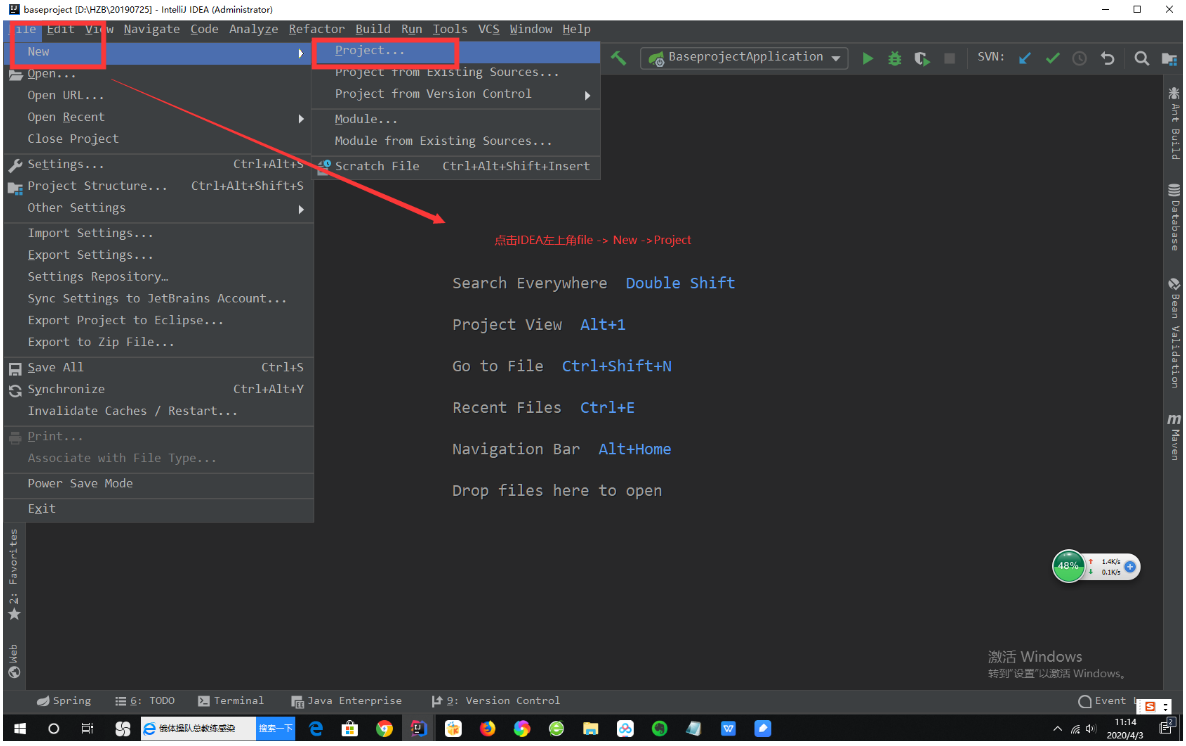
1.2、Spring Security项目构建

1.2.1、项目结构

itmk-base-parent	**父模块，pom类型、统一管理子模块**
itmk-base-common	公共模块、如通用工具的封装等
itmk-base-web	前端接口模块，提供api接口

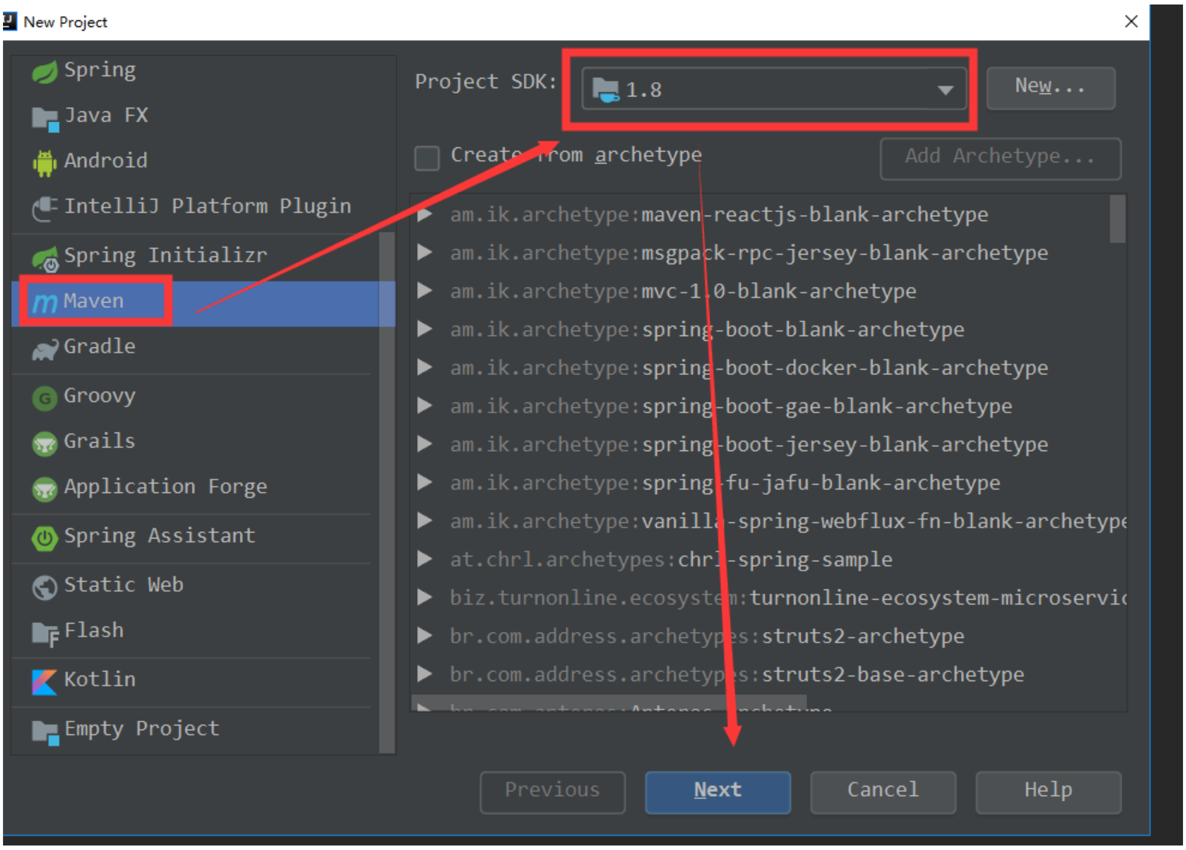
1.2.2、创建itmk-base-parent父模块

1、打开IDEA，file -> new - > project



点击IDEA左上角file -> New ->Project

2、选择maven，JDK选择1.8以上，点击next

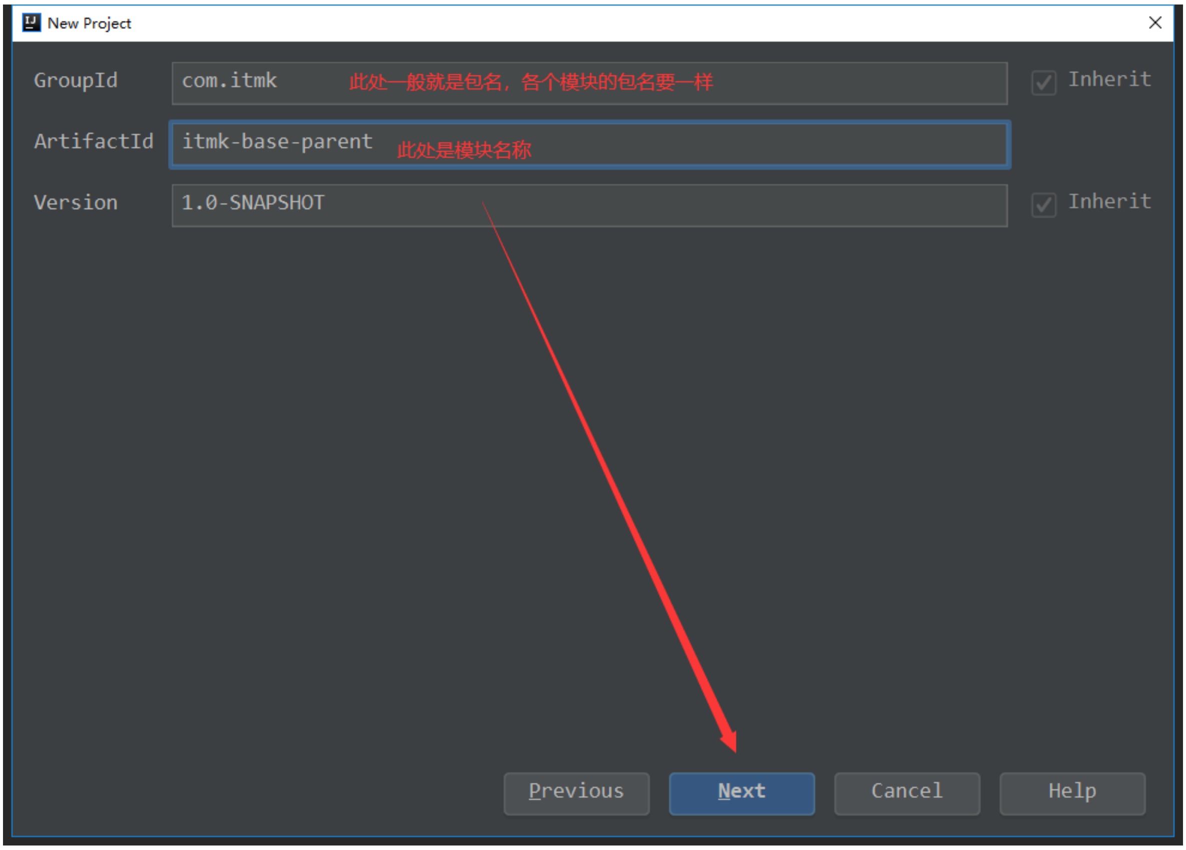


Project SDK: 1.8

□ Create from archetype

- am.ik.archetype:maven-reactjs-blank-archetype
- am.ik.archetype:msgpack-rpc-jersey-blank-archetype
- am.ik.archetype:mvc-1.0-blank-archetype
- am.ik.archetype:spring-boot-blank-archetype
- am.ik.archetype:spring-boot-docker-blank-archetype
- am.ik.archetype:spring-boot-gae-blank-archetype
- am.ik.archetype:spring-boot-jersey-blank-archetype
- am.ik.archetype:spring-fu-jafu-blank-archetype
- am.ik.archetype:vanilla-spring-webflux-fn-blank-archetype
- at.chrl.archetypes:chrl-spring-sample
- biz.turnonline.ecosystem:turnonline-ecosystem-microservic
- br.com.address.archetypes:struts2-archetype
- br.com.address.archetypes:struts2-base-archetype

3、在GroupId输入包名，注意各个模块的包名一致，在ArtifactId输入模块名称，点击next

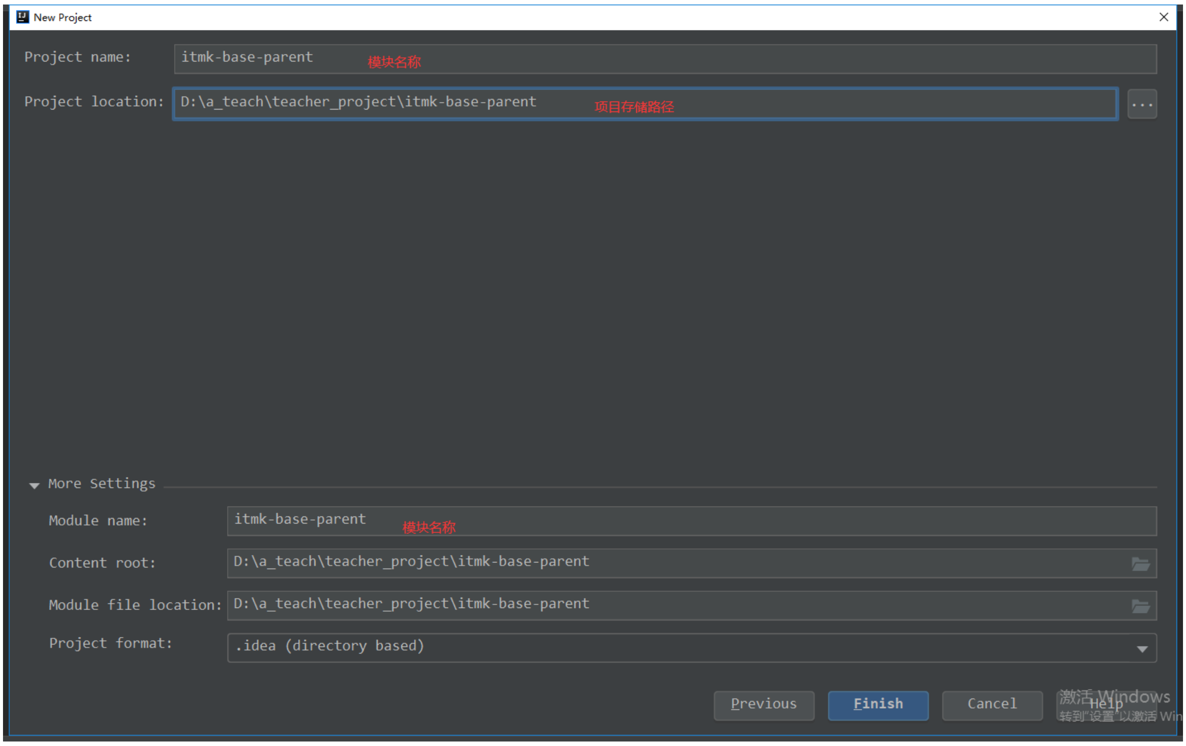


GroupId: com.itmk 此处一般就是包名，各个模块的包名要一样 ☑ Inherit

ArtifactId: itmk-base-parent 此处是模块名称

Version: 1.0-SNAPSHOT ☑ Inherit

4、Project name输入模块名称，Project location选择项目存储路径、在more Settings 录入Model name，点击fiish，完成父模块构建



Project name: itmk-base-parent 模块名称

Project location: D:\a_teach\teacher_project\itmk-base-parent 项目存储路径

More Settings

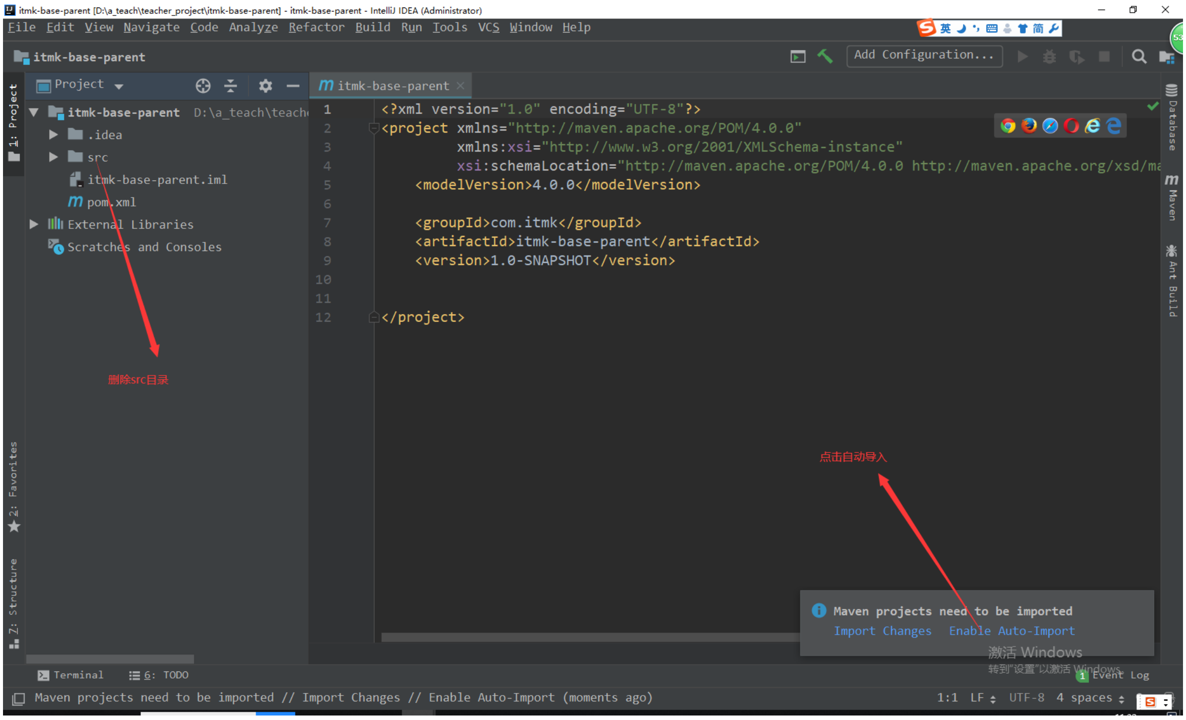
Module name: itmk-base-parent 模块名称

Content root: D:\a_teach\teacher_project\itmk-base-parent

Module file location: D:\a_teach\teacher_project\itmk-base-parent

Project format: .idea (directory based)

5、删除父模块src目录



itmk-base-parent [D:\a_teach\teacher_project\itmk-base-parent] - itmk-base-parent - IntelliJ IDEA (Administrator)
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
itmk-base-parent
Add Configuration...
Project
itmk-base-parent D:\a_teach\teach
.idea
src
itmk-base-parent.iml
pom.xml
External Libraries
Scratches and Consoles
删除src目录
点击自动导入
Terminal
6: TODO
Event Log

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/ma
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.itmk</groupId>
    <artifactId>itmk-base-parent</artifactId>
    <version>1.0-SNAPSHOT</version>


</project>
```

Maven projects need to be imported
Import Changes Enable Auto-Import

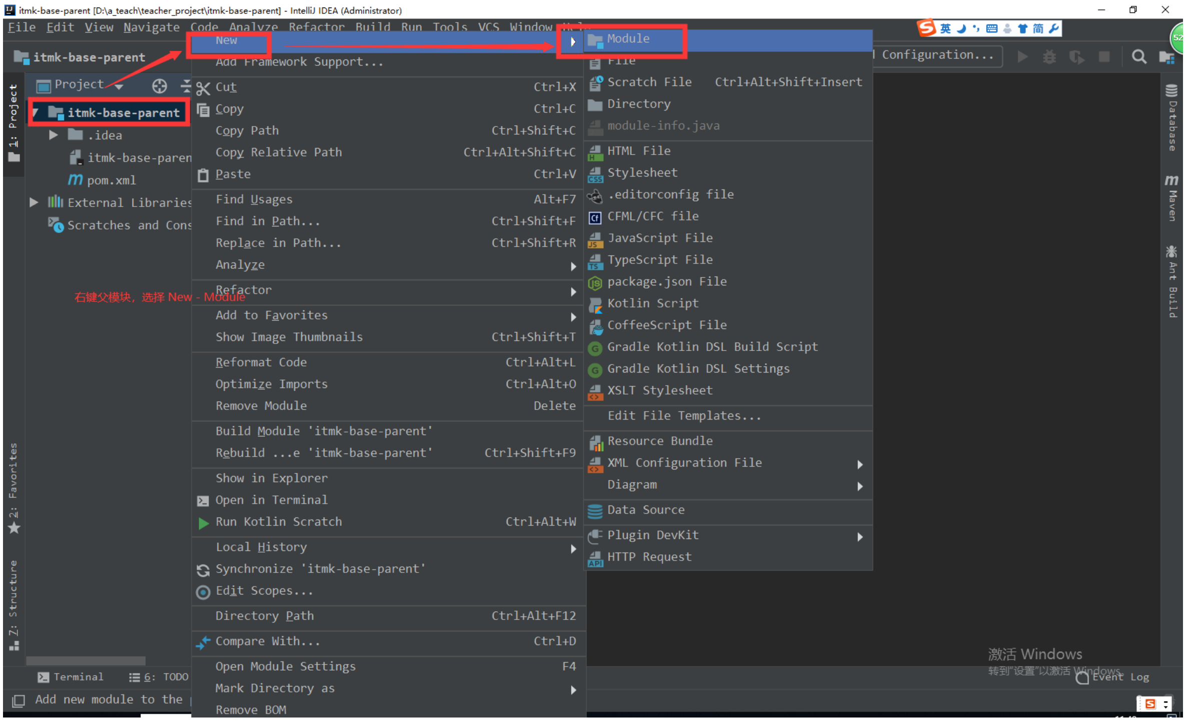
Maven projects need to be imported // Import Changes // Enable Auto-Import (moments ago)

1:1 LF UTF-8 4 spaces

6、在itmk-base-parent.pom.xml文件添加 pom，指定打包模式为pom模式。

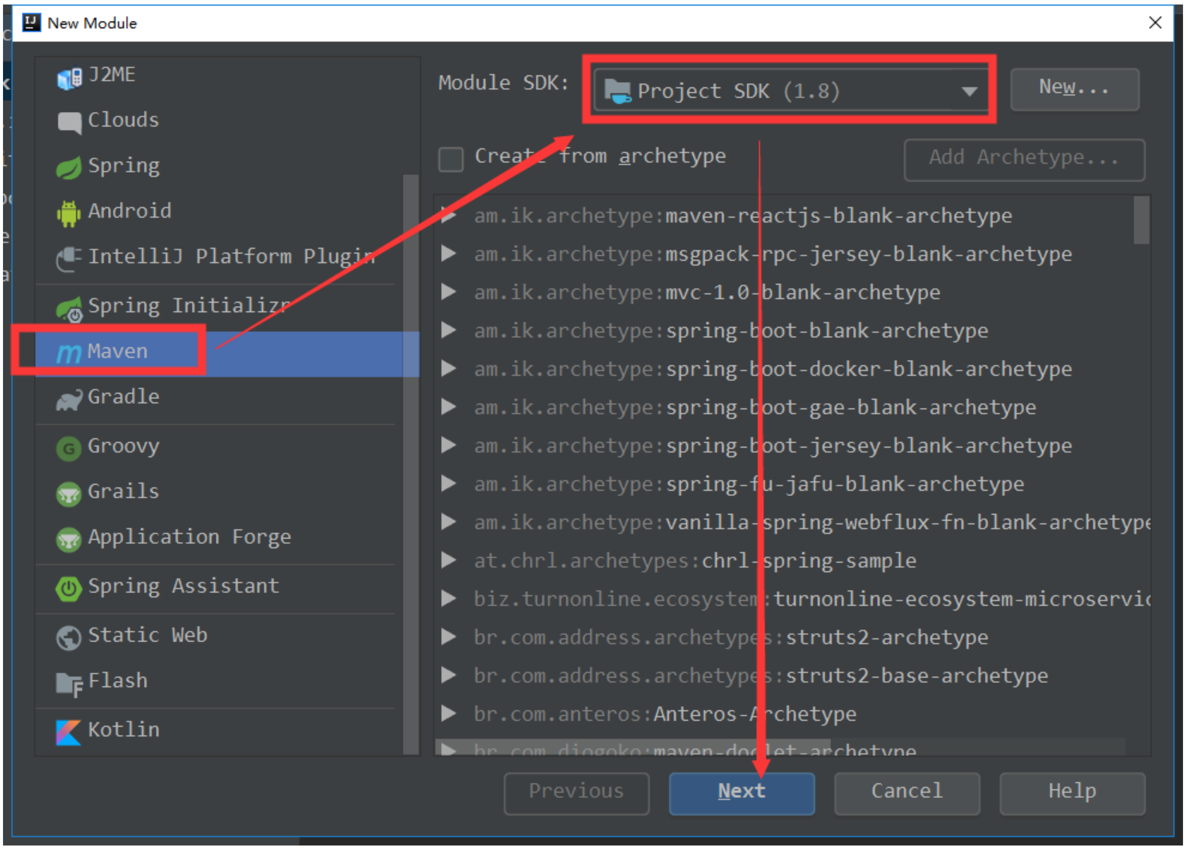
2.2.3、创建itmk-base-common模块

1、点击父模块，右键 -> New -> Module

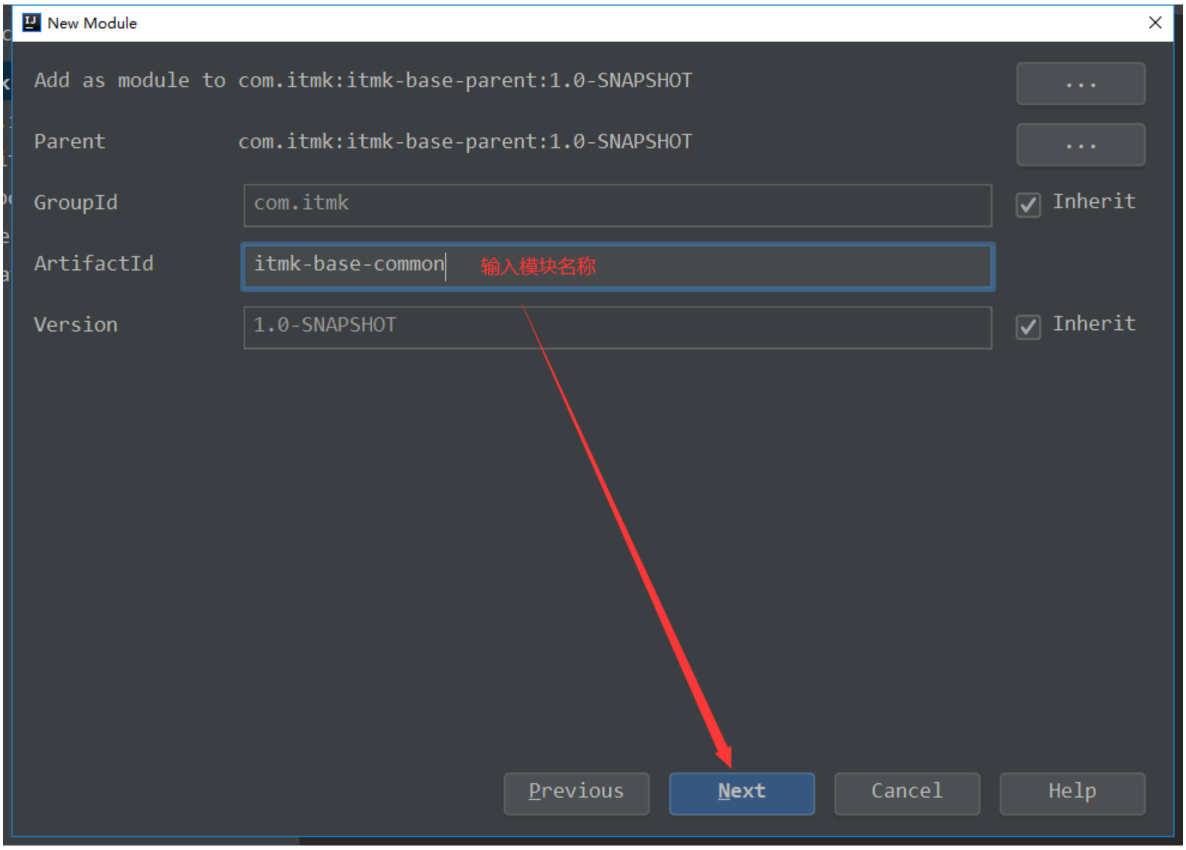


右键父模块，选择 New - Module

2、选择maven ,SDK选择1.8，点击next


New Module
J2ME
Clouds
Spring
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Grails
Application Forge
Spring Assistant
Static Web
Flash
Kotlin
Module SDK:
Project SDK (1.8)
New...
Create from archetype
Add Archetype...
am.ik.archetype:maven-reactjs-blank-archetype
am.ik.archetype:msgpack-rpc-jersey-blank-archetype
am.ik.archetype:mvc-1.0-blank-archetype
am.ik.archetype:spring-boot-blank-archetype
am.ik.archetype:spring-boot-docker-blank-archetype
am.ik.archetype:spring-boot-gae-blank-archetype
am.ik.archetype:spring-boot-jersey-blank-archetype
am.ik.archetype:spring-fu-jafu-blank-archetype
am.ik.archetype:vanilla-spring-webflux-fn-blank-archetype
at.chrl.archetypes:chrl-spring-sample
biz.turnonline.ecosystem:turnonline-ecosystem-microservic
br.com.address.archetypes:struts2-archetype
br.com.address.archetypes:struts2-base-archetype
br.com.anteros:Anteros-Archetype
br.com.diogoko:maven-doclet-archetype
Previous
Next
Cancel
Help

3、ArtifactId 录入模块名称，点击next



Add as module to com.itmk:itmk-base-parent:1.0-SNAPSHOT

Parent com.itmk:itmk-base-parent:1.0-SNAPSHOT

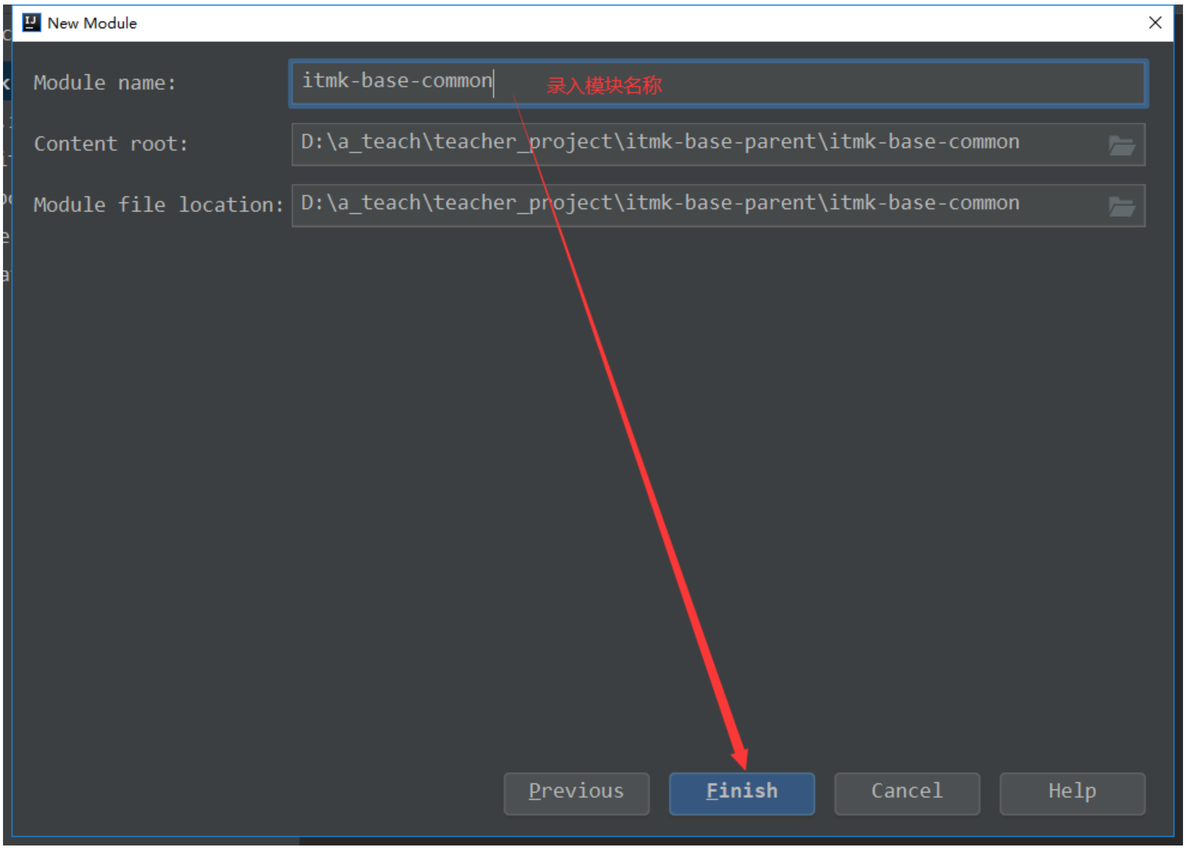
GroupId com.itmk ☑ Inherit

ArtifactId itmk-base-common

输入模块名称

Version 1.0-SNAPSHOT ☑ Inherit

4、Module name录入模块名称，点击next，完成itmk-base-common模块创建



Module name: itmk-base-common

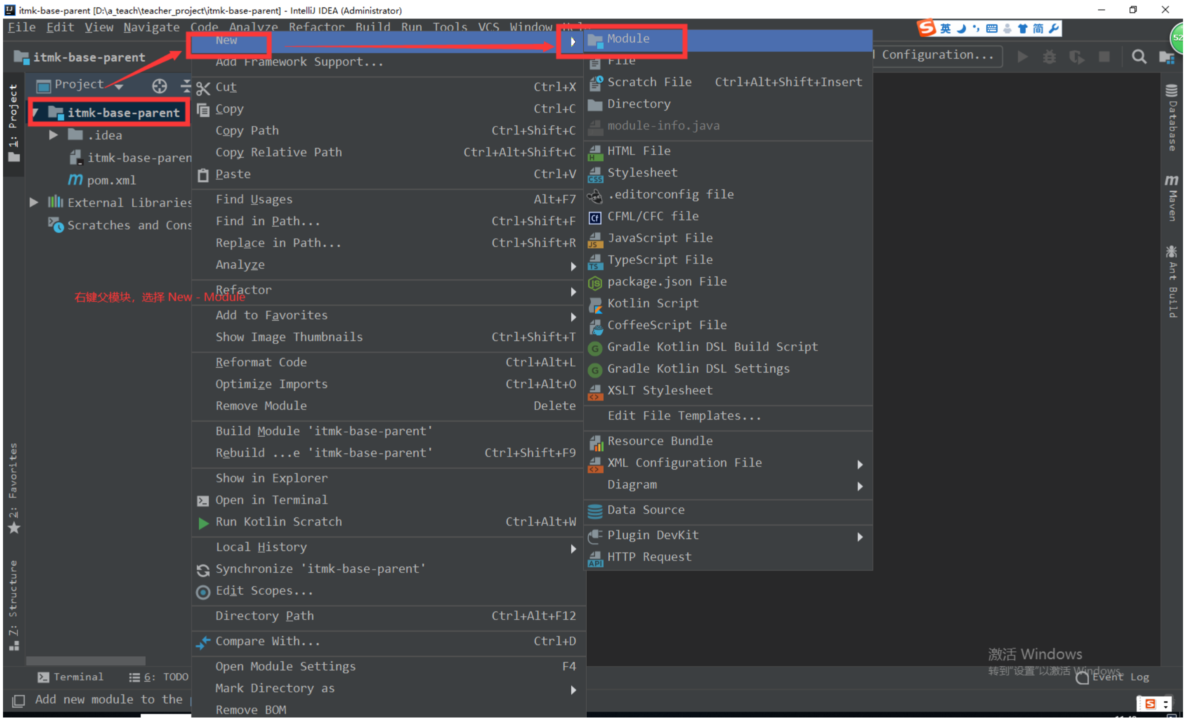
录入模块名称

Content root: D:\a_teach\teacher_project\itmk-base-parent\itmk-base-common

Module file location: D:\a_teach\teacher_project\itmk-base-parent\itmk-base-common

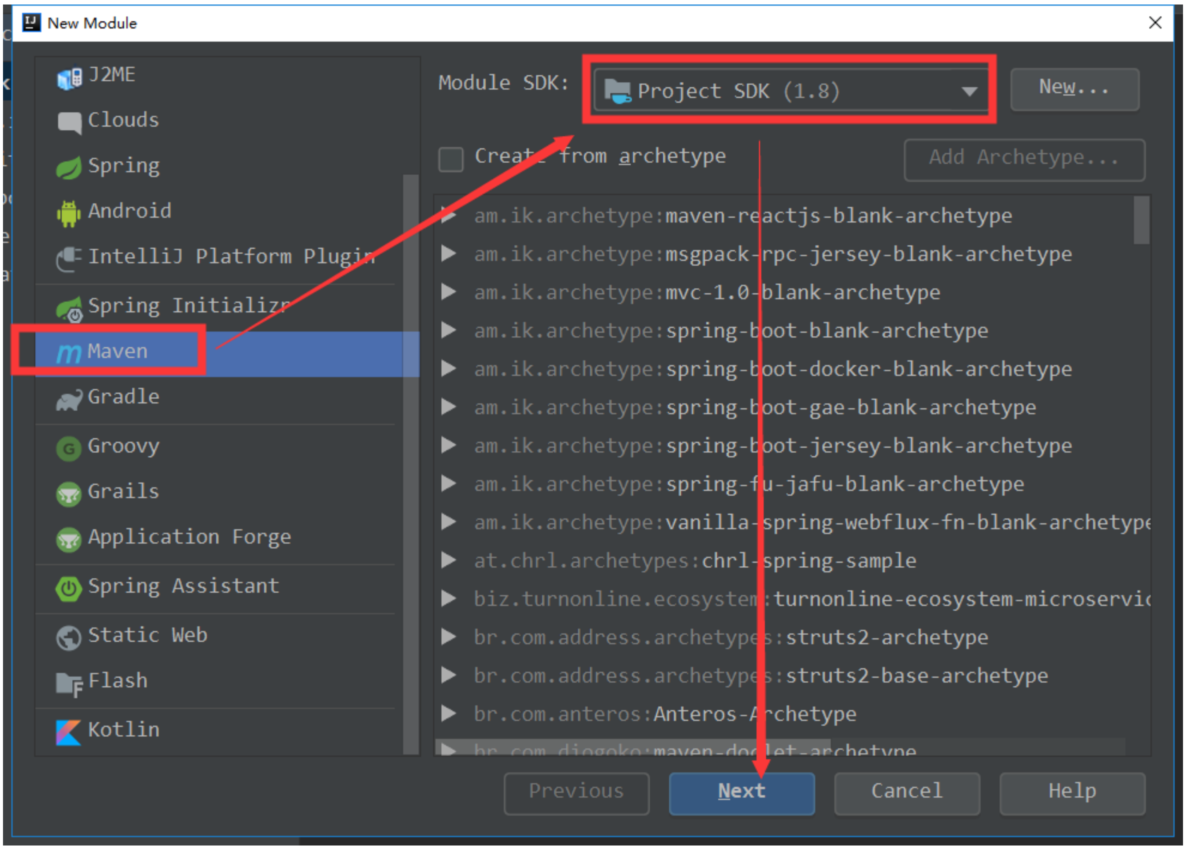
1.2.3、创建itmk-base-web模块

1、点击父模块，右键 -> New -> Module



右键父模块，选择 New - Module

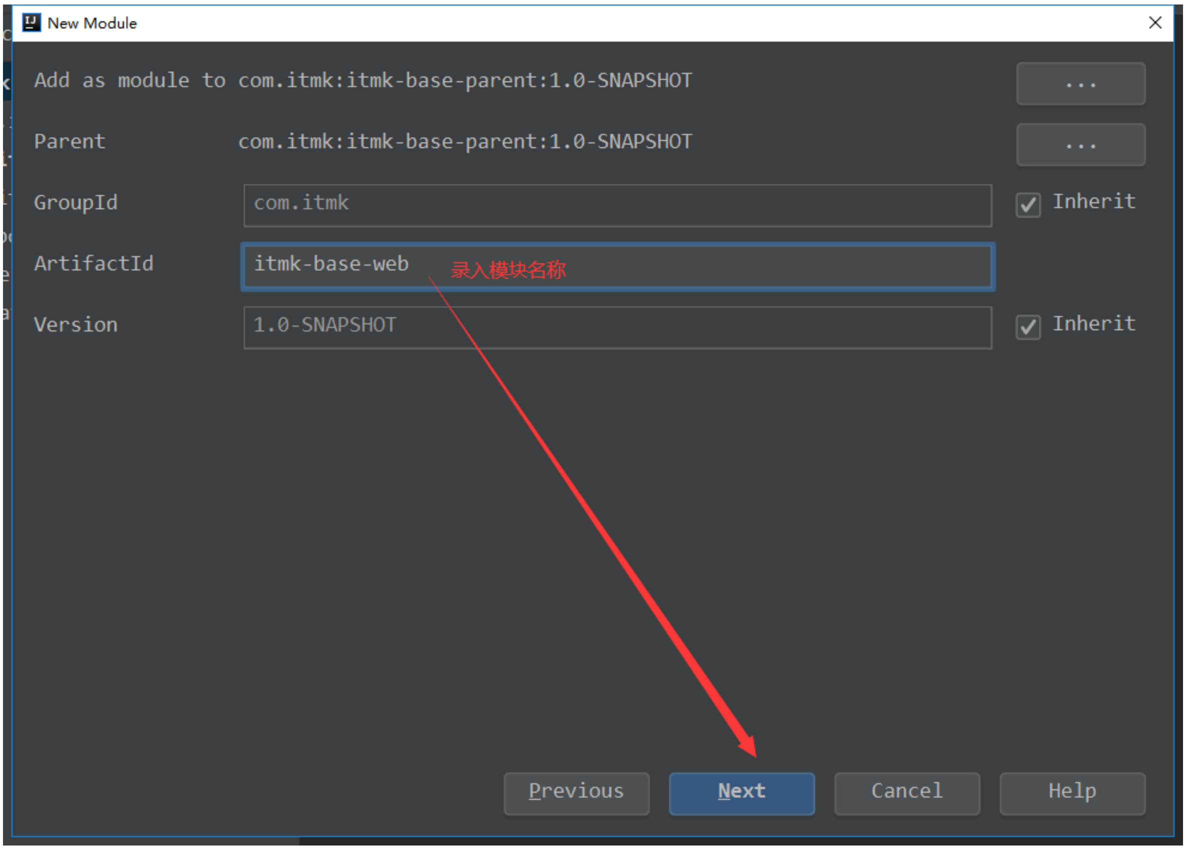
2、选择maven ,SDK选择1.8，点击next


New Module
J2ME
Clouds
Spring
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Grails
Application Forge
Spring Assistant
Static Web
Flash
Kotlin
New...
Add Archetype...
am.ik.archetype:maven-reactjs-blank-archetype
am.ik.archetype:msgpack-rpc-jersey-blank-archetype
am.ik.archetype:mvc-1.0-blank-archetype
am.ik.archetype:spring-boot-blank-archetype
am.ik.archetype:spring-boot-docker-blank-archetype
am.ik.archetype:spring-boot-gae-blank-archetype
am.ik.archetype:spring-boot-jersey-blank-archetype
am.ik.archetype:spring-fu-jafu-blank-archetype
am.ik.archetype:vanilla-spring-webflux-fn-blank-archetype
at.chrl.archetypes:chrl-spring-sample
biz.turnonline.ecosystem:turnonline-ecosystem-microservic
br.com.address.archetypes:struts2-archetype
br.com.address.archetypes:struts2-base-archetype
br.com.anteros:Anteros-Archetype
br.com.diogoko:maven-doclet-archetype
Previous
Next
Cancel
Help

Module SDK: Project SDK (1.8)

□ Create from archetype

3、ArtifactId 录入模块名称，点击next



New Module

Add as module to com.itmk:itmk-base-parent:1.0-SNAPSHOT

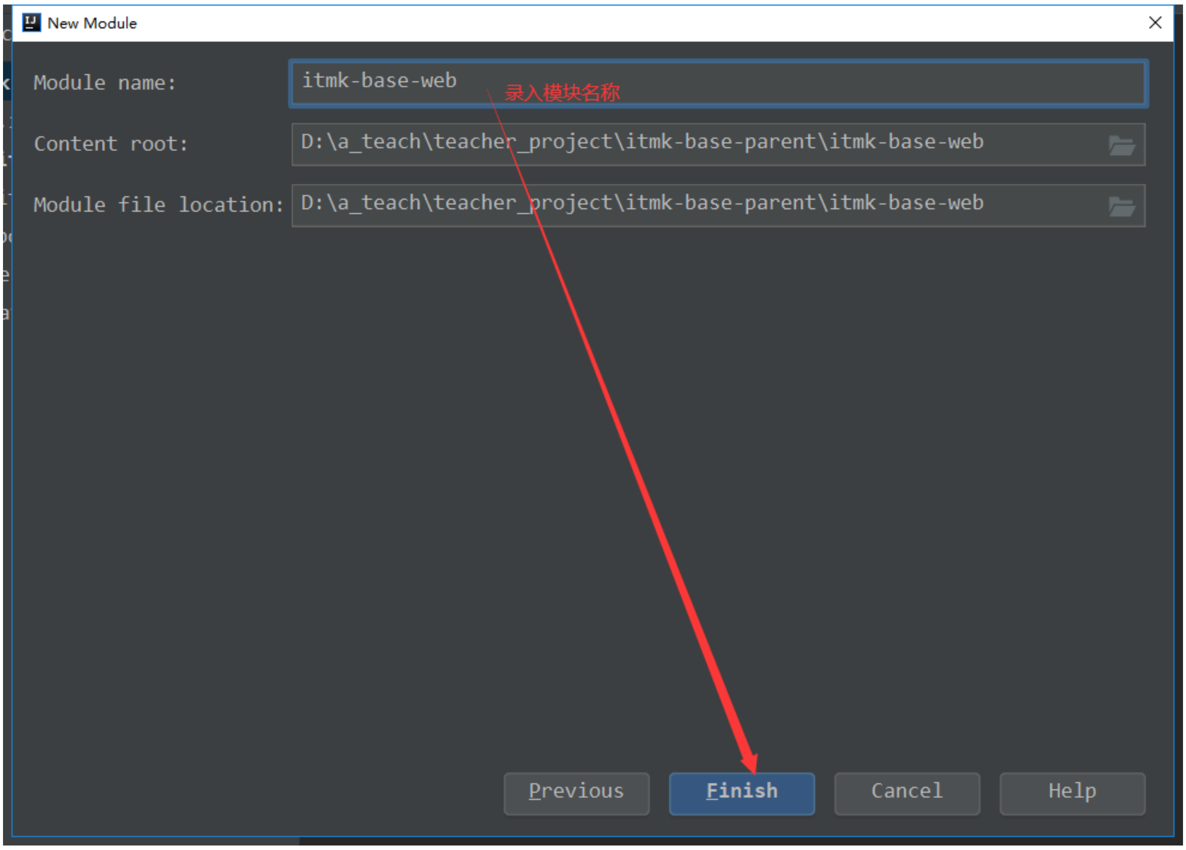
Parent com.itmk:itmk-base-parent:1.0-SNAPSHOT

GroupId: com.itmk ☑ Inherit

ArtifactId: itmk-base-web

Version: 1.0-SNAPSHOT ☑ Inherit

4、Module name 录入模块名称，点击finish完成itmk-base-web模块的创建



New Module

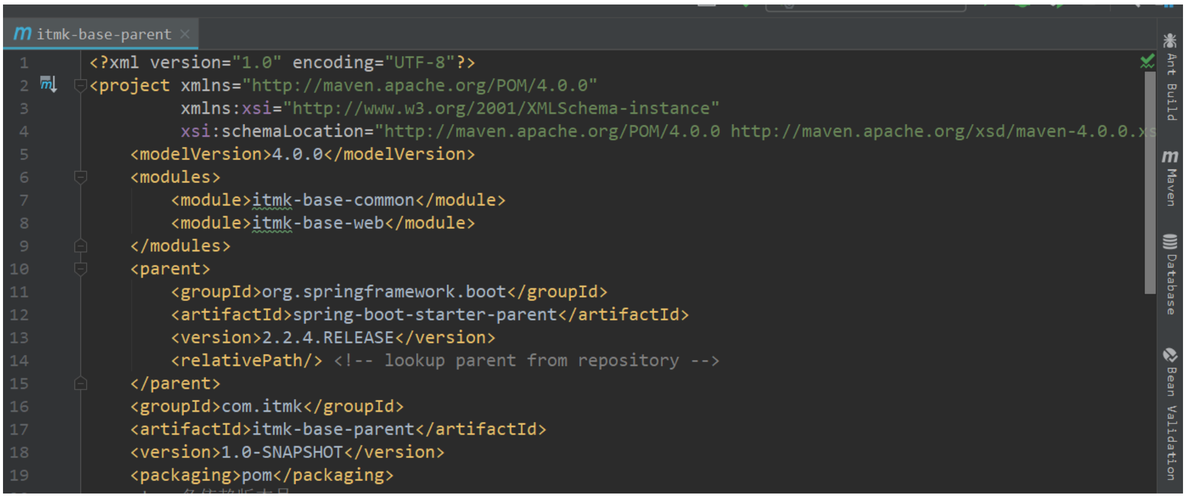
Module name: itmk-base-web

Content root: D:\a_teach\teacher_project\itmk-base-parent\itmk-base-web

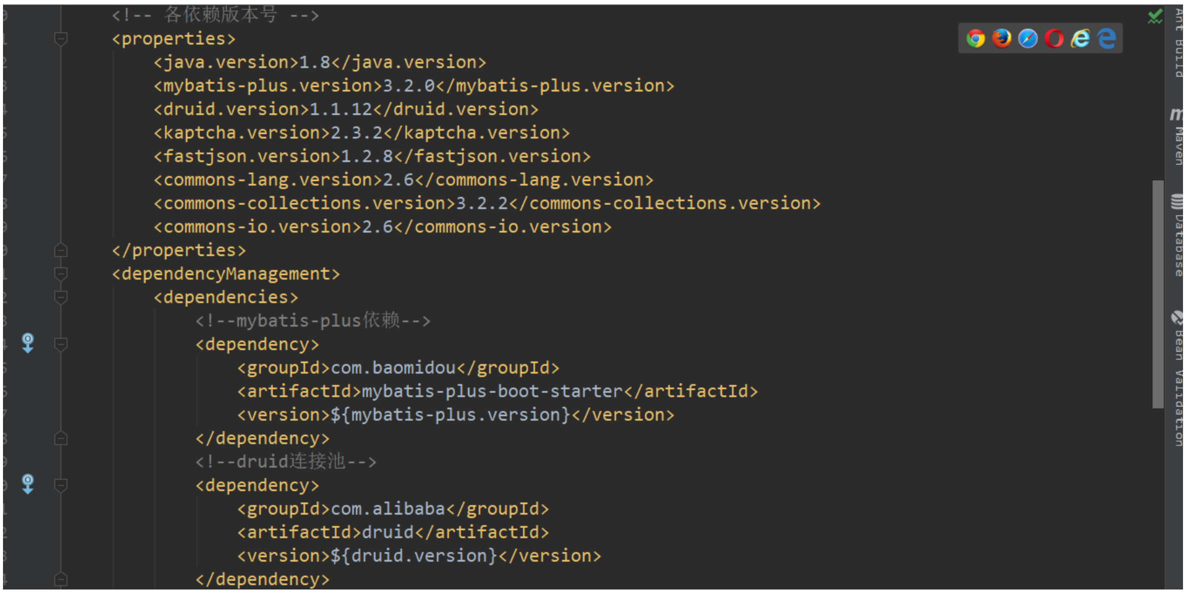
Module file location: D:\a_teach\teacher_project\itmk-base-parent\itmk-base-web

第25讲 添加模块依赖

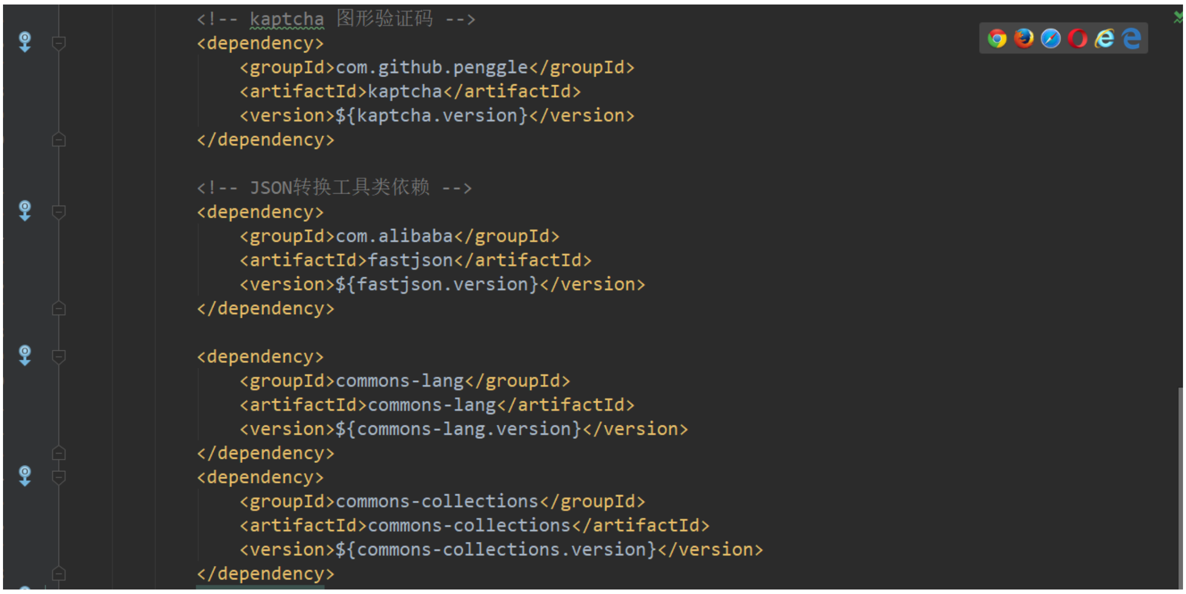
1.1、添加itmk-base-parent pom.xml 依赖



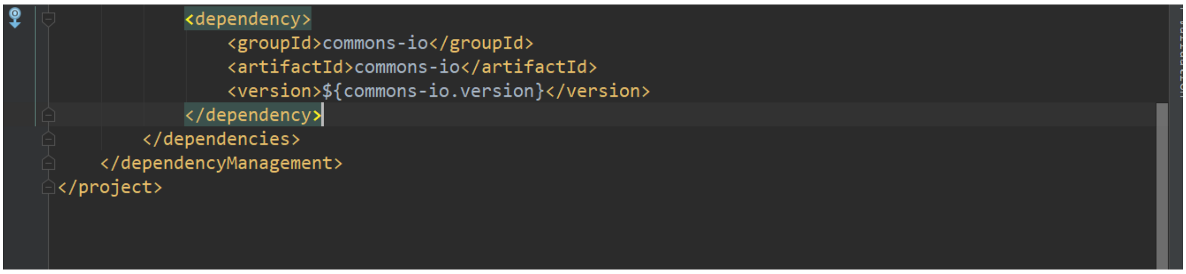
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xs
    <modelVersion>4.0.0</modelVersion>
    <modules>
        <module>itmk-base-common</module>
        <module>itmk-base-web</module>
    </modules>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.itmk</groupId>
    <artifactId>itmk-base-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>
```



```xml
    <!-- 各依赖版本号 -->
    <properties>
        <java.version>1.8</java.version>
        <mybatis-plus.version>3.2.0</mybatis-plus.version>
        <druid.version>1.1.12</druid.version>
        <kaptcha.version>2.3.2</kaptcha.version>
        <fastjson.version>1.2.8</fastjson.version>
        <commons-lang.version>2.6</commons-lang.version>
        <commons-collections.version>3.2.2</commons-collections.version>
        <commons-io.version>2.6</commons-io.version>
    </properties>
    <dependencyManagement>
        <dependencies>
            <!--mybatis-plus依赖-->
            <dependency>
                <groupId>com.baomidou</groupId>
                <artifactId>mybatis-plus-boot-starter</artifactId>
                <version>${mybatis-plus.version}</version>
            </dependency>
            <!--druid连接池-->
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>${druid.version}</version>
            </dependency>
```



```xml
            <!-- kaptcha 图形验证码 -->
            <dependency>
                <groupId>com.github.penggle</groupId>
                <artifactId>kaptcha</artifactId>
                <version>${kaptcha.version}</version>
            </dependency>

            <!-- JSON转换工具类依赖 -->
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>fastjson</artifactId>
                <version>${fastjson.version}</version>
            </dependency>

            <dependency>
                <groupId>commons-lang</groupId>
                <artifactId>commons-lang</artifactId>
                <version>${commons-lang.version}</version>
            </dependency>
            <dependency>
                <groupId>commons-collections</groupId>
                <artifactId>commons-collections</artifactId>
                <version>${commons-collections.version}</version>
            </dependency>
```

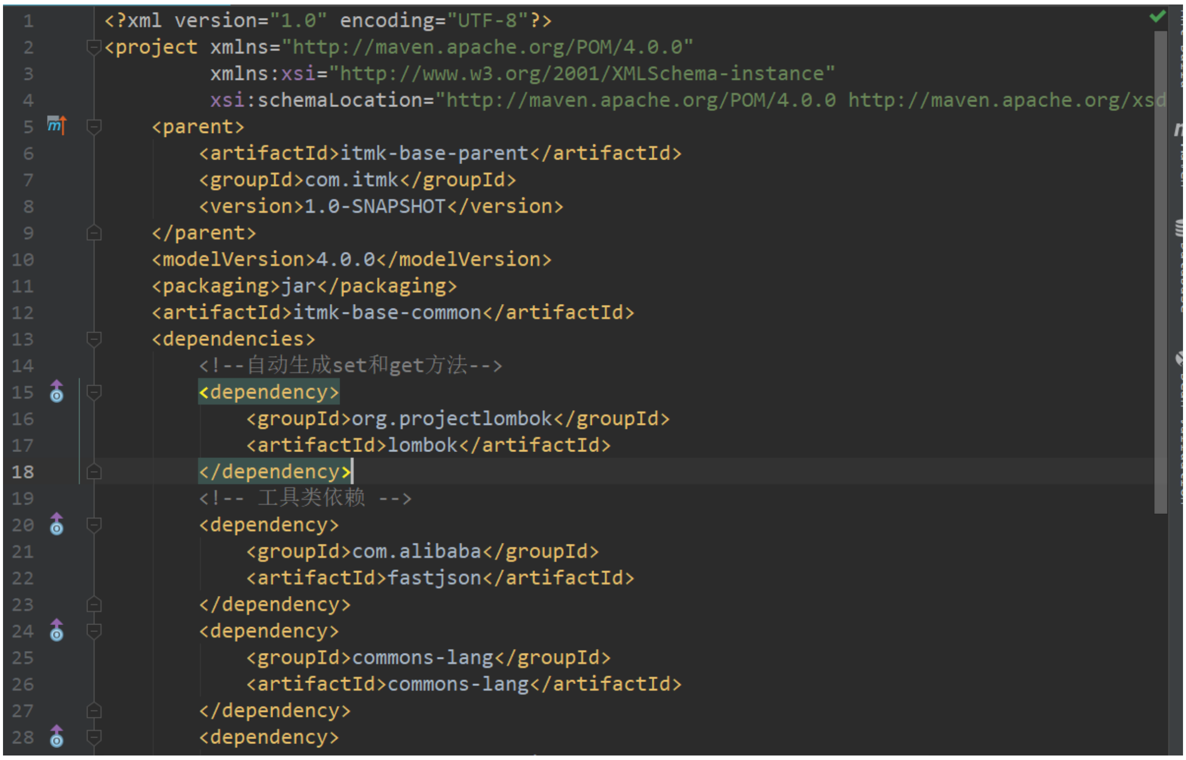


```xml
            <dependency>
                <groupId>commons-io</groupId>
                <artifactId>commons-io</artifactId>
                <version>${commons-io.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
</project>
```

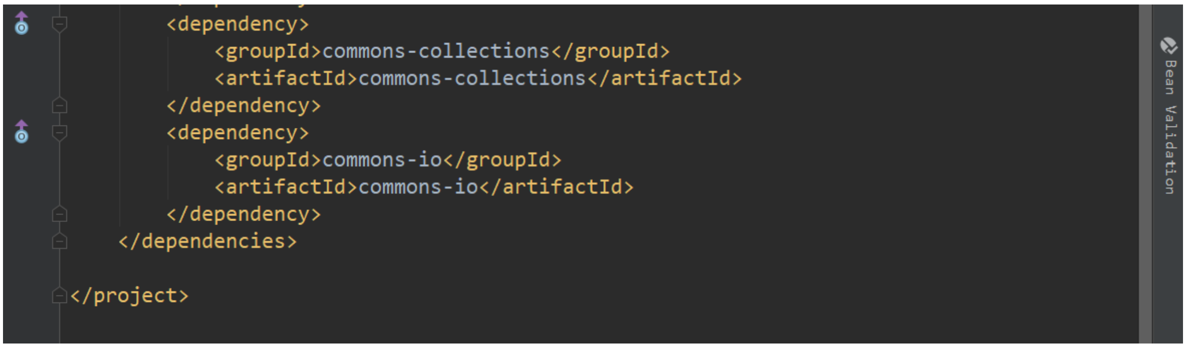
itmk-base-parent pom.xml文件

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <modules>
        <module>itmk-base-common</module>
        <module>itmk-base-web</module>
    </modules>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.itmk</groupId>
    <artifactId>itmk-base-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>
    <!-- 各依赖版本号 -->
    <properties>
        <java.version>1.8</java.version>
        <mybatis-plus.version>3.2.0</mybatis-plus.version>
        <druid.version>1.1.12</druid.version>
        <kaptcha.version>2.3.2</kaptcha.version>
        <fastjson.version>1.2.68</fastjson.version>
        <commons-lang.version>2.6</commons-lang.version>
        <commons-collections.version>3.2.2</commons-collections.version>
        <commons-io.version>2.6</commons-io.version>
    </properties>
    <dependencyManagement>
        <dependencies>
            <!--mybatis-plus依赖-->
            <dependency>
                <groupId>com.baomidou</groupId>
                <artifactId>mybatis-plus-boot-starter</artifactId>
                <version>${mybatis-plus.version}</version>
            </dependency>
            <!--druid连接池-->
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>${druid.version}</version>
            </dependency>
            <!-- kaptcha 图形验证码 -->
            <dependency>
                <groupId>com.github.penggle</groupId>
                <artifactId>kaptcha</artifactId>
                <version>${kaptcha.version}</version>
            </dependency>

            <!-- JSON转换工具类依赖 -->
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>fastjson</artifactId>
                <version>${fastjson.version}</version>
            </dependency>

            <dependency>
                <groupId>commons-lang</groupId>
                <artifactId>commons-lang</artifactId>
                <version>${commons-lang.version}</version>
            </dependency>
            <dependency>
                <groupId>commons-collections</groupId>
                <artifactId>commons-collections</artifactId>
                <version>${commons-collections.version}</version>
            </dependency>
            <dependency>
                <groupId>commons-io</groupId>
                <artifactId>commons-io</artifactId>
                <version>${commons-io.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
</project>
```

1.2、添加itmk-base-common 模块 pom.xml依赖



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
    <parent>
        <artifactId>itmk-base-parent</artifactId>
        <groupId>com.itmk</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>
    <artifactId>itmk-base-common</artifactId>
    <dependencies>
        <!--自动生成set和get方法-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <!-- 工具类依赖 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
        </dependency>
        <dependency>
```



```
        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
        </dependency>
    </dependencies>

</project>
```

```
itmk-base-common pom.xml文件
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>itmk-base-parent</artifactId>
        <groupId>com.itmk</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>
    <artifactId>itmk-base-common</artifactId>
    <dependencies>
        <!-- jwt-->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.0</version>
        </dependency>
        <!--自动生成set和get方法-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <!-- 工具类依赖 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
        </dependency>
        <!--采用redis来管理-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
    </dependencies>

</project>
```

1.3、添加itmk-base-web.xml依赖

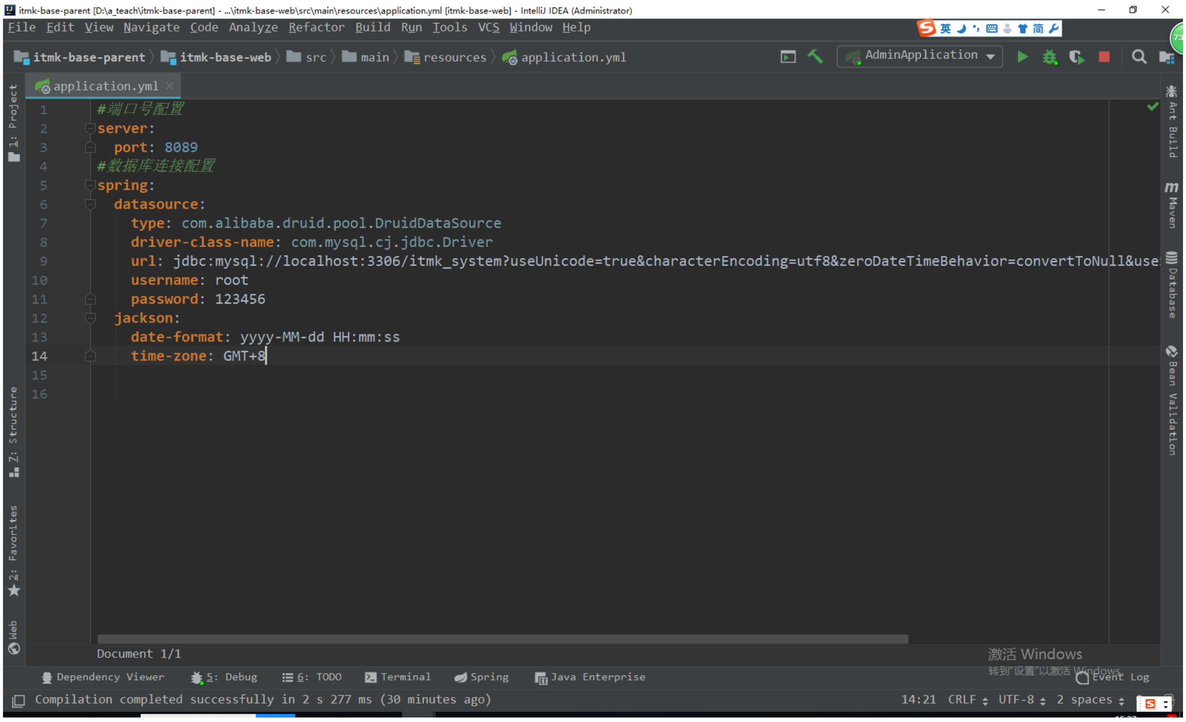
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>itmk-base-parent</artifactId>
        <groupId>com.itmk</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>
    <artifactId>itmk-base-web</artifactId>
    <dependencies>
        <dependency>
            <groupId>com.itmk</groupId>
            <artifactId>itmk-base-common</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
        <!--web启动器,对springmvc,serlvet等支持-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- spring security 启动器-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
        </dependency>


        <!--图片验证码-->
        <dependency>
            <groupId>com.github.penggle</groupId>
            <artifactId>kaptcha</artifactId>
        </dependency>

        <!--数据库依赖-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>

        <!--解决找不到 javax.annotation.meta.When.MAYBE-->
        <dependency>
            <groupId>com.google.code.findbugs</groupId>
            <artifactId>annotations</artifactId>
            <version>3.0.1</version>
        </dependency>
        <!--mybatis-plus启动器-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
        </dependency>

        <!--druid连接池-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
        </dependency>
        <!-- application.yml 配置处理器-->
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>

        <!-- springboot 单元测试 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
        </dependency>
        <!--热部署 ctrl+f9-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.1.4.RELEASE</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.19.1</version>
                <configuration>
                    <skipTests>true</skipTests>    <!--打包过程默认关掉单元测试 -->
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

第26讲 整合Mybatis Plus

1.1、新建application.yml文件

1、在itmk-base-web模块resources文件下新建 application.yml文件


itmk-base-parent [D:\a_teach\itmk-base-parent] - ...\itmk-base-web\src\main\resources\application.yml [itmk-base-web] - IntelliJ IDEA (Administrator)
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
itmk-base-parent 〉 itmk-base-web 〉 src 〉 main 〉 resources 〉 application.yml
AdminApplication
application.yml
Document 1/1
Dependency Viewer
5: Debug
6: TODO
Terminal
Spring
Java Enterprise
Event Log
14:21 CRLF UTF-8 2 spaces

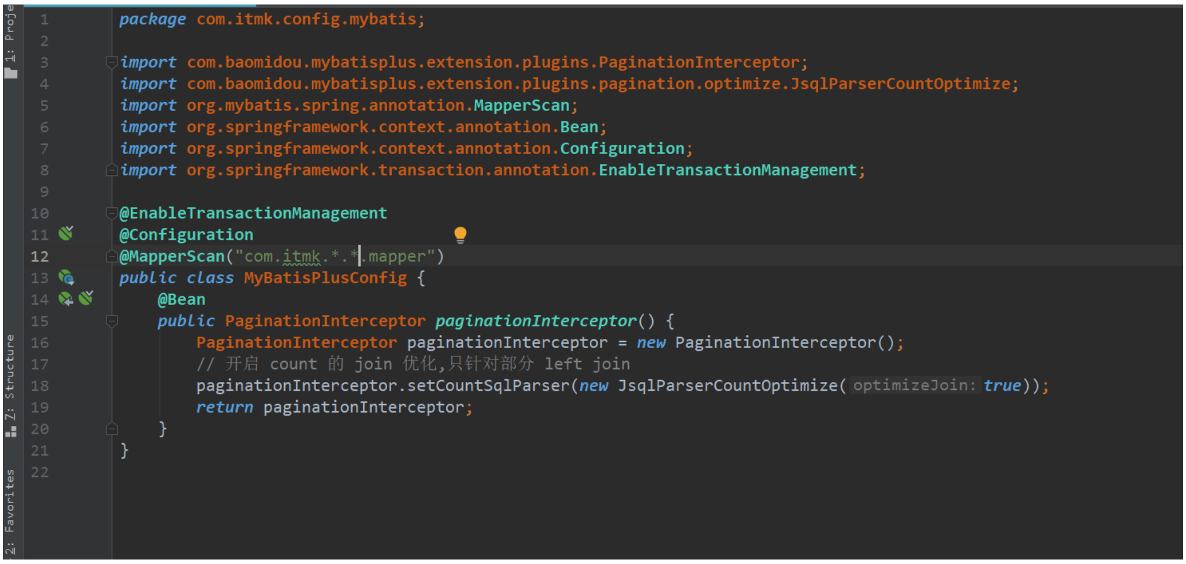
```yaml
#端口号配置
server:
  port: 8089
#数据库连接配置
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/itmk_system?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&use
    username: root
    password: 123456
  jackson:
    date-format: yyyy-MM-dd HH:mm:ss
    time-zone: GMT+8
```

Compilation completed successfully in 2 s 277 ms (30 minutes ago)

```yaml
#端口号配置
server:
  port: 8089
#数据库连接配置
spring:
  profiles:
    active: common
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/itmk-system?
useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&
serverTimezone=GMT%2B8
    username: root
    password: 123456
```

1.2、配置Mybatis PlusConfig

1.2.1、在itmk-base-web模块新建MyBatisPlusConfig配置类



```java
package com.itmk.config.mybatis;

import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;
import com.baomidou.mybatisplus.extension.plugins.pagination.optimize.JsqlParserCountOptimize;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@EnableTransactionManagement
@Configuration
@MapperScan("com.itmk.*.*.mapper")
public class MyBatisPlusConfig {
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        PaginationInterceptor paginationInterceptor = new PaginationInterceptor();
        // 开启 count 的 join 优化,只针对部分 left join
        paginationInterceptor.setCountSqlParser(new JsqlParserCountOptimize( optimizeJoin: true));
        return paginationInterceptor;
    }
}

```

1.2.2、在com.itmk下新建config->mybatis->MyBatisPlusConfig配置类

```
@EnableTransactionManagement
@Configuration
@MapperScan("com.itmk.*.*.mapper")
public class MyBatisPlusConfig {
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        PaginationInterceptor paginationInterceptor = new PaginationInterceptor();
        // 开启 count 的 join 优化,只针对部分 left join
        paginationInterceptor.setCountSqlParser(new JsqlParserCountOptimize(true));
        return paginationInterceptor;
    }
}
```

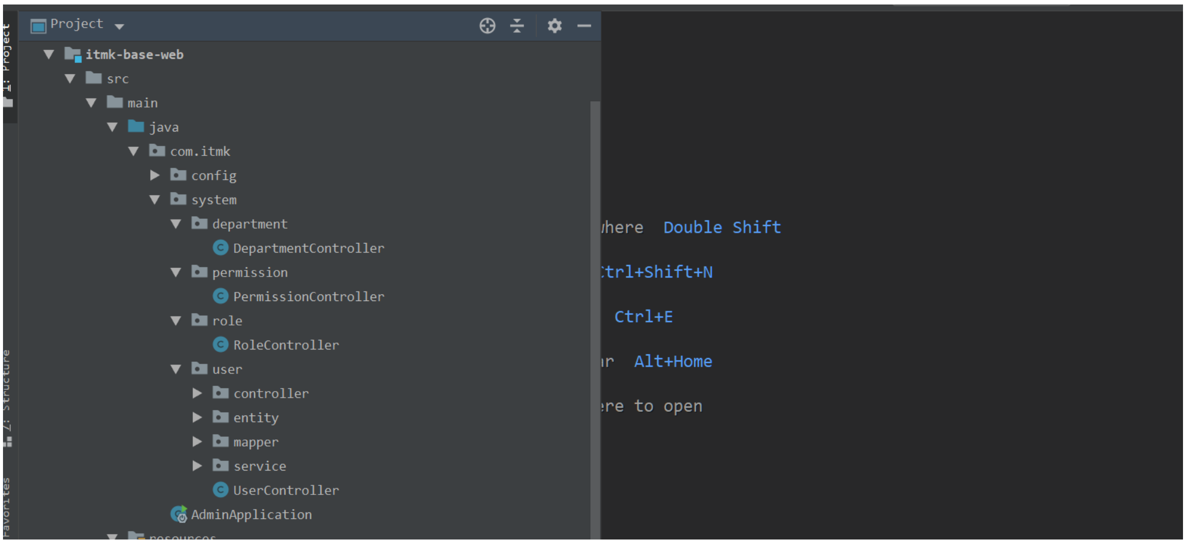
1.2.3、在config-> datasource下新建DruidConfig配置类

```
@Slf4j
@Configuration
public class DruidConfig {
    @ConfigurationProperties(prefix = "spring.datasource")
    @Bean
    public DataSource dataSource(){
       return new DruidDataSource();
   }
}
```

1.3、测试MybatisPlus是否整合成功

1.3.1、导入sql文件

1.3.2新建各个包



```
Project
▼ itmk-base-web
  ▼ src
    ▼ main
      ▼ java
        ▼ com.itmk
          ▶ config
          ▼ system
            ▼ department
                DepartmentController
            ▼ permission
                PermissionController
            ▼ role
                RoleController
            ▼ user
              ▶ controller
              ▶ entity
              ▶ mapper
              ▶ service
                UserController
          AdminApplication
```

在itmk-base-web模块下新建department、permission、role、user等四个模块

1.3.3、新建user模块的mapper、service、controller

1.SysUser实体

```java
package com.itmk.system.user.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.io.Serializable;
import java.util.Collection;
import java.util.Date;

/**
 * 用户表
 */
@Data
@TableName("sys_user")
public class SysUser implements UserDetails, Serializable {
    //主键自动增长
    @TableId(type = IdType.AUTO)
    private Long id;
    //登录名
    private String username;
    //用户名
    private String loginName;
    //登录密码，密码需要加密
    private String password;
    //帐户是否过期(1 未过期，0已过期)
    private boolean isAccountNonExpired = true;
    //帐户是否被锁定(1 未锁定，0已锁定)
    private boolean isAccountNonLocked = true;
    //密码是否过期(1 未过期，0已过期)
    private boolean isCredentialsNonExpired = true;
    //帐户是否可用(1 可用，0 删除用户)
    private boolean isEnabled = true;
    //由于authorities不是数据库里面的自动，所以要排除他，不然mybatis-plus找不到该字段会报错
    @TableField(exist = false)
    Collection<? extends GrantedAuthority> authorities;
    //昵称
    private String nickName;
    //手机号
    private String mobile;
    //邮箱
    private String email;
    //部门id
    private Long deptId;
    //部门名称
    private String deptName;
    //创建时间
    @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private Date createTime;
    //更新时间
    @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private Date updateTime;
    //是否是管理员 1：是 0 ：不是
    private String isAdmin;
}
```

2.新建SysUserMapper接口

```
package com.itmk.system.user.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.itmk.system.user.entity.SysUser;

public interface SysUserMapper extends BaseMapper<SysUser> {
}
```

3.在itmk-base-web模块下resources下新建mapper目录，然后新建SysUserMapper.xml

注意 xml文件名和第二步mapper中接口SysUserMapper名保持一致

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.user.mapper.SysUserMapper">

</mapper>
```

3.创建UserServie接口

```
package com.itmk.system.user.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.user.entity.SysUser;

/**
 * 用户service层接口
 */
public interface UserService extends IService<SysUser> {

}
```

4.新建UserService实现类

```
package com.itmk.system.user.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.itmk.system.user.entity.SysUser;
import com.itmk.system.user.mapper.UserMapper;
import com.itmk.system.user.service.UserService;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl extends ServiceImpl<SysUserMapper,SysUser> implements
UserService {
}
```

5.新建UserController文件

```
package com.itmk.system.user.controller;

import com.itmk.result.ResultVo;
import com.itmk.system.user.entity.SysUser;
import com.itmk.system.user.service.UserService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@Slf4j
@RestController
@RequestMapping("/system/user")
public class UserController {

    @Autowired
    private UserService userService;

    /**
     * 获取用户信息列表
     * @return
     */
    @RequestMapping(value = "getUser",method = RequestMethod.GET)
    public ResultVo getUser(){
        ResultVo resultVo = new ResultVo();
        List<SysUser> list = userService.list();
        resultVo.setData(list);
        return resultVo;
    }
}
```

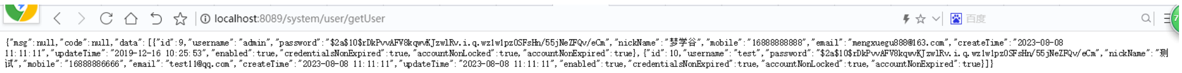
5.注释itmk-base-web pom.xml文件中的spring security启动jar

```
 <!-- spring security 启动器-->
        <!--<dependency>-->
            <!--<groupId>org.springframework.boot</groupId>-->
            <!--<artifactId>spring-boot-starter-security</artifactId>-->
        <!--</dependency>-->
```

6.配置启动类

```
@SpringBootApplication
public class AdminApplication {
    public static void main(String[] args) {
        SpringApplication.run(AdminApplication.class,args);
    }
}
```

7.浏览器访问 http://localhost:8089/system/user/getUser，返回json数据



```
{"msg":null,"code":null,"data":[{"id":9,"username":"admin","password":"$2a$10$rDkPvvAFV8kqwvKJzwlRv.i.q.wz1w1pz0SFsHn/55jNeZFQv/eCm","nickName":"梦学谷","mobile":"16888888888","email":"mengxuegu888@163.com","createTime":"2023-08-08
11:11:11","updateTime":"2019-12-16 10:25:53","enabled":true,"credentialsNonExpired":true,"accountNonLocked":true,"accountNonExpired":true},{"id":10,"username":"test","password":"$2a$10$rDkPvvAFV8kqwvKJzwlRv.i.q.wz1w1pz0SFsHn/55jNeZFQv/eCm","nickName":"测
试","mobile":"16888886666","email":"test11@qq.com","createTime":"2023-08-08 11:11:11","updateTime":"2023-08-08 11:11:11","enabled":true,"credentialsNonExpired":true,"accountNonLocked":true,"accountNonExpired":true}]}
```

第27讲 项目基础工具类讲解

1.1、返回数据类型封装

在common模块新建com.itmk.result包，新建返回数据类型类ResultVo

```
package com.itmk.result;

import lombok.AllArgsConstructor;
import lombok.Data;

/**
 * 返回实体数据
 * @param <T>
 */
@Data
@AllArgsConstructor
public class ResultVo<T> {
    /**
     * 返回提示信息
     */
    private String msg;
    /**
     * 返回状态码
     */
    private int code;
    /**
     * 返回数据
     */
    private T data;
}
```

1.2、返回分页数据实体封装 ResultPageVo

在common模块com.itmk.reslut包下新建 分页返回值类型类 ResultPageVo

```
import lombok.AllArgsConstructor;
import lombok.Data;

/**
 * 返回分页数据实体
 * @param <T>
 */
@Data
@AllArgsConstructor
public class ResultPageVo<T> {
    /**
     * 返回提示信息
     */
    private String msg;
    /**
     * 返回状态码
     */
    private Integer code;
    /**
     * 当前第几页
     */
    private Integer pageNum;
    /**
     * 每页条数
     */
    private Integer pageSize;
    /**
     * 总条数
     */
    private Integer total;
    /**
     * 返回数据
     */
    private T data;

}
```

1.3、数据返回工具类 ResultUtils

在common模块com.itmk.result包下新建ResultUtils类

```
package com.itmk.result;

import com.itmk.status.CodeStatus;

/**
 * 数据返回工具类
 */
public class ResultUtils {
    /**
     * 无参数返回
     * @return
     */
    public static ResultVo succcess() {
        return Vo(null, CodeStatus.SUCCESS_CODE, null);
    }
    public static ResultVo success(String msg){
        return Vo(msg,CodeStatus.SUCCESS_CODE,null);
    }
    /**
     * 返回带参数
     * @param msg
     * @param data
     * @return
     */
    public static ResultVo success(String msg,Object data){
        return Vo(msg,CodeStatus.SUCCESS_CODE,data);
    }
    public static ResultVo success(String msg,int code,Object data){
        return Vo(msg,code,data);
    }
    public static ResultVo Vo(String msg, int code, Object data) {
        return new ResultVo(msg, code, data);
    }

    /**
     * 错误返回
     * @return
     */
    public static ResultVo error(){
        return Vo(null,CodeStatus.ERROR_CODE,null);
    }
    public static ResultVo error(String msg){
        return Vo(msg,CodeStatus.ERROR_CODE,null);
    }
    public static ResultVo error(String msg,int code,Object data){
        return Vo(msg,code,data);
    }
    public static ResultVo error(String msg,int code){
        return Vo(msg,code,null);
    }
    public static ResultVo error(String msg,Object data){
        return Vo(msg,CodeStatus.ERROR_CODE,data);
    }
    public static ResultPageVo success(String msg,Integer pageNum,Integer
pageSize,Integer total,Object data){
        return new
ResultPageVo(null,CodeStatus.SUCCESS_CODE,pageNum,pageSize,total,data);
    }
}
```

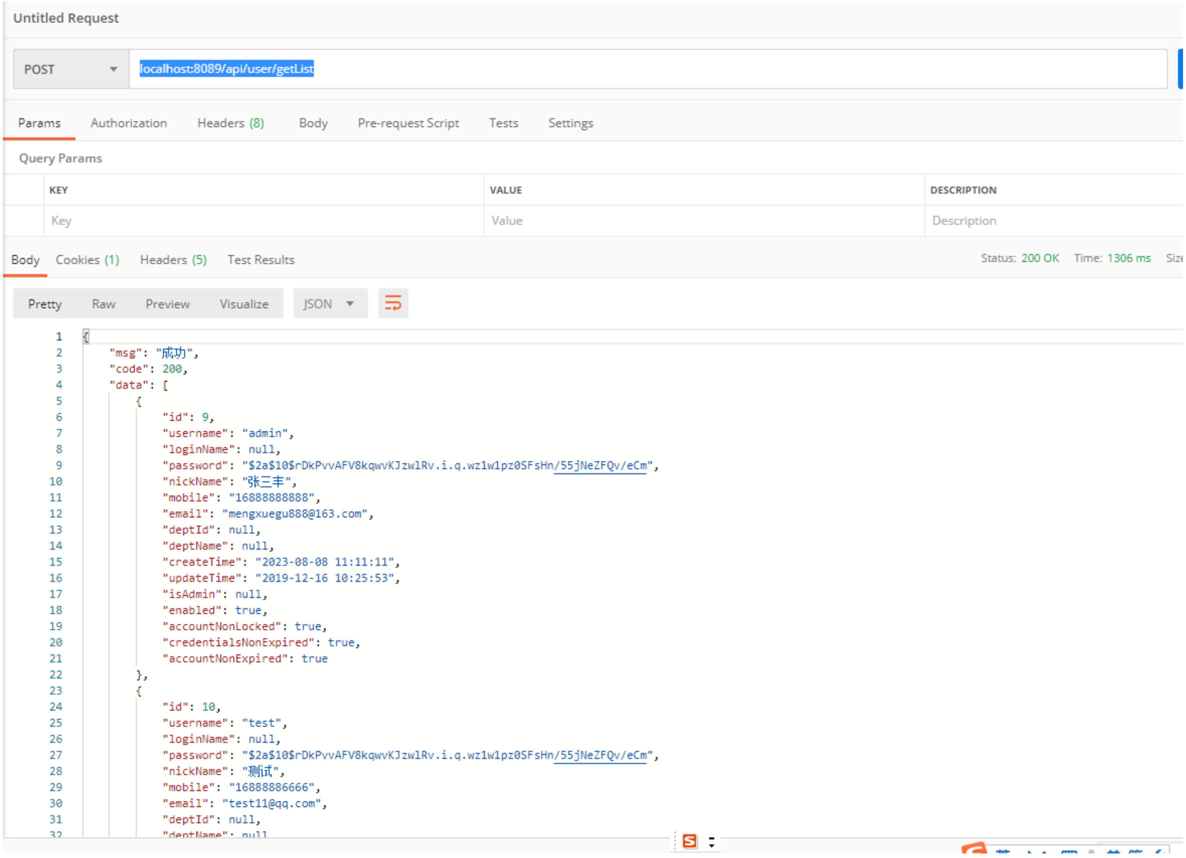
1.4、测试返回工具类

把UserController中getList返回值改为 ResultVo

```
package com.itmk.system.user.controller;

import com.itmk.result.ResultUtils;
import com.itmk.result.ResultVo;
import com.itmk.system.user.entity.SysUser;
import com.itmk.system.user.service.UserService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@Slf4j
@RestController
@RequestMapping("/api/user")
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value = "/getList",method = RequestMethod.POST)
    public ResultVo getList(){
        List<SysUser> list = userService.list();
       return ResultUtils.success("成功",list);
    }
}
```

1.5、postman请求

localhost:8089/api/user/getList



Untitled Request

POST localhost:8089/api/user/getList

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 1306 ms Siz

```
{
    "msg": "成功",
    "code": 200,
    "data": [
        {
            "id": 9,
            "username": "admin",
            "loginName": null,
            "password": "$2a$10$rDkPvvAFV8kqwvKJzwlRv.i.q.wz1w1pz0SFsHn/55jNeZFQv/eCm",
            "nickName": "张三丰",
            "mobile": "16888888888",
            "email": "mengxuegu888@163.com",
            "deptId": null,
            "deptName": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2019-12-16 10:25:53",
            "isAdmin": null,
            "enabled": true,
            "accountNonLocked": true,
            "credentialsNonExpired": true,
            "accountNonExpired": true
        },
        {
            "id": 10,
            "username": "test",
            "loginName": null,
            "password": "$2a$10$rDkPvvAFV8kqwvKJzwlRv.i.q.wz1w1pz0SFsHn/55jNeZFQv/eCm",
            "nickName": "测试",
            "mobile": "16888886666",
            "email": "test11@qq.com",
            "deptId": null,
            "deptName": null
```

第28讲 RBAC模型讲解

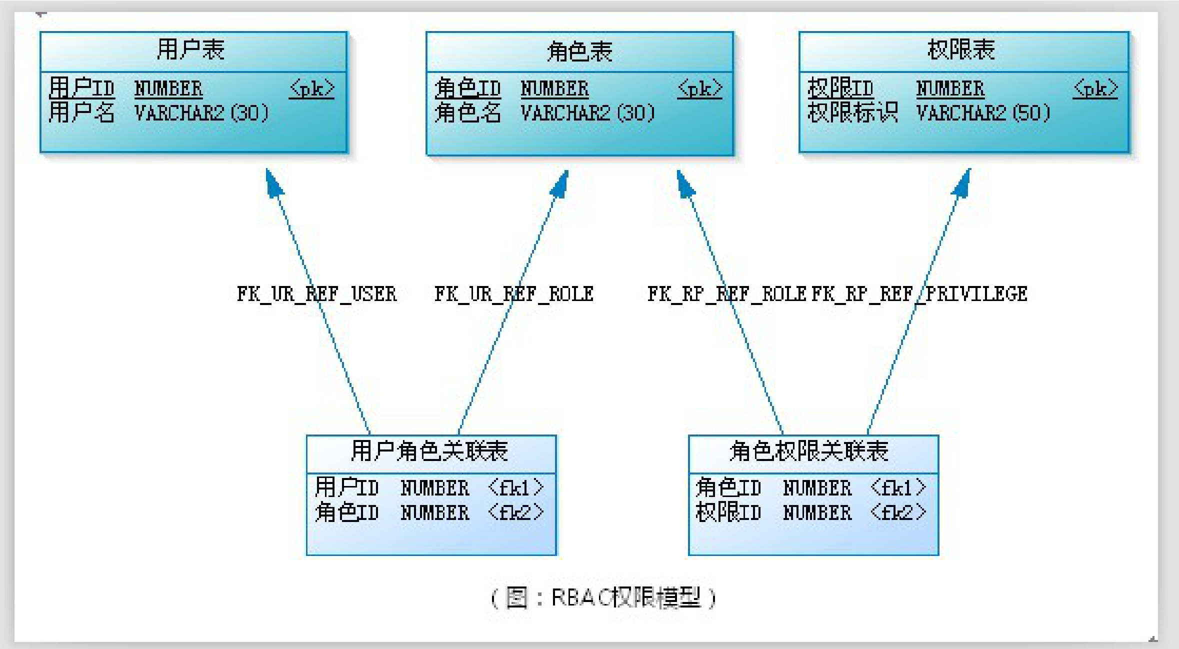
1.1、什么是RBAC模型

RBAC（Role-Based Access Control ）基于角色的访问控制

1.2、RBAC原理

RBAC（Role-Based Access Control，基于角色的访问控制），就是用户通过角色与权限进行关联。简单地说，一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成'用户-角色-权限'的授权模型。在这种模型中，用户与角色之间，角色与权限之间，一般是多对多的关系。

1.3、基本RBAC模型图



（图：RBAC权限模型）

1.4、RBAC理解

如一个公司，有部门，有用户，就有对应的角色（部门经理、项目经理、部门组长等），不同的角色，他们拥有不同的权限。

第29讲 用户认证和授权

1.1、什么是认证

认证简单说就是登录，查询数据库看看登录的用户在数据库存不存在

1.2、什么是授权

授权就是验证用户可以访问的菜单、按钮、数据等。

1.3、Spring Security认证原理

1.3.1、Spring Security登录认证主要涉及两个重要的接口 UserDetailService和UserDetail接口

1.3.2、UserDetailService接口主要定义了一个方法 loadUserByUsername(String username)用于完成用户信息的查询，其中username就是登录时的登录名称，登录认证时，需要自定义一个实现类实现UserDetailService接口，完成数据库查询，该接口返回UserDetail。

1.3.3、UserDetail主要用于封装认证成功时的用户信息，即UserDetailService返回的用户信息，可以用Spring自己的User对象，但是最好是实现UserDetail接口，自定义用户对象。

1.4、登录认证步骤

1.4.1、在UserService中添加中添加getUserByUserName(String username),用户查询用户是否存在

```
package com.itmk.system.user.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.user.entity.SysUser;

/**
```

```
 * 用户service层接口
 */
public interface UserService extends IService<SysUser> {
    /**
     * 根据用户名查询用户信息
     * @param username
     * @return
     */
    SysUser getUserByUserName(String username);
}
```

1.4.2、在UserServiceImpl中实现getUserByUserName方法

```
 @Override
 public SysUser getUserByUserName(String username) {
        QueryWrapper<SysUser> query = new QueryWrapper<>();
        query.lambda().eq(SysUser::getUsername,username);
        return this.baseMapper.selectOne(query);
    }
```

1.4.3、自定义认证类CustomerUserDetailsService

自定义 CustomerUserDetailsService 实现 UserDetailService， 并实现 loadUserByUsername 方法，loadUserByUsername方法主要作用是查询用户是否存在和设置用户权限信息

在itmk-base-web模块新建 com.itmk.security.detailservice包，新建CustomerUserDetailsService类，实现loadUserByUsername

```
package com.itmk.security.detailservice;



/**
 * 认证处理类
 * 查询数据库是否有用户
 */
@Slf4j
@Component("customerUserDetailService")
public class CustomerUserDetailsService implements UserDetailsService {

    @Autowired
    private UserService userService;

    //这里需要注入PasswordEncoder，不然会报错的
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        //1.根据用户username查询数据库是否有用户
        SysUser user = userService.getUserByUserName(username);
        if(null == user){
            throw new UsernameNotFoundException("用户名或密码错误!");
        }
        //2.查询用户的权限
        //3.设置用户权限
        return user;
    }
}
```

1.4.4、在itmk-base-web模块新建 com.itmk.jwt包新建JwtUtils类

```
package com.itmk.jwt;

import com.itmk.system.user.entity.SysUser;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import lombok.Data;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

@Data
@ConfigurationProperties(prefix = "jwt")
@Component
public class JwtUtils {
    private String secret;

    // 过期时间 毫秒
    private Long expiration;

    private String header;

    /**
     * 从数据声明生成令牌
     *
     * @param claims 数据声明
     * @return 令牌
     */
    private String generateToken(Map<String, Object> claims) {
        Date expirationDate = new Date(System.currentTimeMillis() + expiration);
        return
Jwts.builder().setClaims(claims).setExpiration(expirationDate).signWith(SignatureAlgor
ithm.HS512, secret).compact();
    }

    /**
     * 从令牌中获取数据声明
     *
     * @param token 令牌
     * @return 数据声明
     */
    private Claims getClaimsFromToken(String token) {
        Claims claims;
        try {
            claims =
Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();
        } catch (Exception e) {
            claims = null;
        }
        return claims;
    }

    /**
     * 生成令牌
     *
     * @param userDetails 用户
     * @return 令牌
     */
    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>(2);
        claims.put(Claims.SUBJECT, userDetails.getUsername());
        claims.put(Claims.ISSUED_AT, new Date());
        claims.put("authorities", userDetails.getAuthorities());
        return generateToken(claims);
    }

    /**
     * 从令牌中获取用户名
     *
     * @param token 令牌
     * @return 用户名
     */
    public String getUsernameFromToken(String token) {
        String username;
        try {
            Claims claims = getClaimsFromToken(token);
            username = claims.getSubject();
        } catch (Exception e) {
            username = null;
        }
        return username;
    }

    /**
     * 判断令牌是否过期
     *
     * @param token 令牌
     * @return 是否过期
     */
    public Boolean isTokenExpired(String token) {
        Claims claims = getClaimsFromToken(token);
        if(claims == null){
            return false;
        }
        Date expiration = claims.getExpiration();
        return expiration.after(new Date());
    }

    /**
     * 刷新令牌
     *
     * @param token 原令牌
     * @return 新令牌
     */
    public String refreshToken(String token) {
        String refreshedToken;
        try {
            Claims claims = getClaimsFromToken(token);
            claims.put(Claims.ISSUED_AT, new Date());
            refreshedToken = generateToken(claims);
        } catch (Exception e) {
            refreshedToken = null;
        }
        return refreshedToken;
    }

    /**
     * 验证令牌
     *
     * @param token       令牌
     * @param userDetails 用户
     * @return 是否有效
     */
    public Boolean validateToken(String token, UserDetails userDetails) {
        SysUser user = (SysUser) userDetails;
        String username = getUsernameFromToken(token);
        return (username.equals(user.getUsername()) && isTokenExpired(token));
    }
}
```

1.4.5、在itmk-base-web下新建com.itmk.security.handler包，并在包下新建如下类

1.4.5.1、新建认证成功处理器 LoginSuccessHandler，由于项目采用前后端分离项目，登录认证成功需要返回JSON数据，该类主要用于处理认证成功返回JSON数据和生成token，返回用户权限菜单等

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.itmk.KeyCode.KeyCode;
import com.itmk.config.redis.CacheService;
import com.itmk.jwt.JwtUtils;
import com.itmk.result.ResultUtils;
import com.itmk.status.CodeStatus;
import com.itmk.system.permission.Vo.MenuVo;
import com.itmk.system.permission.entity.Permission;
import com.itmk.system.permission.service.PermissionService;
import com.itmk.system.user.entity.SysUser;
import com.itmk.system.user.service.UserService;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;
import java.util.stream.Collectors;

/**
 * 登录成功处理器
 *
 * 登录成功要返回json和token
 */
@Component("loginSuccessHandler")
public class LoginSuccessHandler implements AuthenticationSuccessHandler {
    @Autowired
    private JwtUtils jwtUtils;

    @Autowired
    private UserService userService;

    @Override
    public void onAuthenticationSuccess(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Authentication authentication) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream out = httpServletResponse.getOutputStream();
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        String token = jwtUtils.generateToken(userDetails);
        MenuVo vo = new MenuVo();
        vo.setToken(token);
        String username = ((UserDetails) authentication.getPrincipal()).getUsername();

        if (StringUtils.isEmpty(username)) {
            ResultUtils.success("用户信息过期", CodeStatus.NO_AUTN, null);
        }
        //获取用户
       SysUser user = (SysUser)authentication.getPrincipal();
        if(user == null){
            ResultUtils.success("用户信息过期", CodeStatus.NO_AUTN, null);
        }
        vo.setUserId(user.getId());
        //查询用户菜单
        String str = JSONObject.toJSONString(ResultUtils.success("认证成
功",vo),SerializerFeature.DisableCircularReferenceDetect);
        out.write(str.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
}
```

1.4.6、新建MenuVo类

在itmk-base-web下新建com.itmk.system.permission.Vo包，并新建MenuVo类，用于返回认证成功用户信息

```
package com.itmk.system.permission.Vo;

import com.itmk.system.permission.entity.Permission;
import lombok.Data;

import java.io.Serializable;
import java.util.List;

/**
 * 菜单返回实体
 */
@Data
public class MenuVo implements Serializable {
   private List<Permission> menuList;
```

```
    private List<String> authList;
    private List<Permission> routerList;
    private String token;
    private Long userId;
}
```

1.4.7、在com.itmk.security.handler包中新建登录认证失败处理器

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.itmk.result.ResultUtils;
import com.itmk.security.image_code.ImageCodeException;
import org.springframework.security.authentication.*;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.authentication.AuthenticationFailureHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * 登录失败返回处理
 */
@Component("loginFailureHandler")
public class LoginFailureHandler implements AuthenticationFailureHandler {
    @Override
    public void onAuthenticationFailure(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, AuthenticationException e) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream out = httpServletResponse.getOutputStream();
        String str = null;
        if (e instanceof AccountExpiredException) {
            //账号过期
            str = "账户过期，登录失败!";
        } else if (e instanceof BadCredentialsException) {
            //密码错误
            str = "用户名或密码输入错误，登录失败!";
        } else if (e instanceof CredentialsExpiredException) {
            //密码过期
            str = "密码过期，登录失败!";
        } else if (e instanceof DisabledException) {
            //账号不可用
            str = "账户被禁用，登录失败!";
        } else if (e instanceof LockedException) {
            //账号锁定
            str = "账户被锁定，登录失败!";
        } else if (e instanceof InternalAuthenticationServiceException) {
            //用户不存在
            str = "用户不存在";
        }else if(e instanceof ImageCodeException){
            //验证码异常
            str = e.getMessage();
        }
        else{
            //其他错误
            str = "登录失败!";
        }
        String rstr = JSONObject.toJSONString(ResultUtils.error(str),
SerializerFeature.DisableCircularReferenceDetect);
        out.write(rstr.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
}
```

1.4.8、在com.itmk.security.handler包中新建匿名用户和认证用户访问失败处理器

1、匿名用户访问失败处理器

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSON;
import com.itmk.result.ResultUtils;
import com.itmk.status.CodeStatus;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 *  匿名用户访问无权限资源时的异常
 */
@Component("customizeAuthenticationEntryPoint")
public class CustomizeAuthenticationEntryPoint implements AuthenticationEntryPoint {
    @Override
    public void commence(HttpServletRequest httpServletRequest, HttpServletResponse
httpServletResponse, AuthenticationException e) throws IOException, ServletException {
        httpServletResponse.setContentType("text/json;charset=utf-8");
        httpServletResponse.getWriter().write(JSON.toJSONString(ResultUtils.error("用
户未登录", CodeStatus.NO_AUTN)));
    }
}
```

2、认证用户无权限访问处理器

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSONObject;
import com.itmk.result.ResultUtils;
import com.itmk.status.CodeStatus;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.web.access.AccessDeniedHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
@Component("customAccessDeineHandler")
public class CustomAccessDeineHandler implements AccessDeniedHandler {

    @Override
    public void handle(HttpServletRequest request, HttpServletResponse response,
                       AccessDeniedException accessDeniedException) throws
IOException, ServletException {
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/javascript;charset=utf-8");
        response.getWriter().print(JSONObject.toJSONString(ResultUtils.error("没有访问
权限!", CodeStatus.NO_AUTN)));
    }

}
```

1.4.9、Spring Security核心配置：WebSecurityConfig配置

WebSecurityConfig主要完成自定义认证处理器、登录成功处理器、登录失败处理器、登录请求URL、会话管理等的配置

新建com.itmk.config.security_config包，并新建SpeingSecurityConfig配置类

```
package com.itmk.config.security_config;

import com.itmk.security.detailservice.CustomerUserDetailsService;
import com.itmk.security.handler.CustomAccessDeineHandler;
import com.itmk.security.handler.CustomizeAuthenticationEntryPoint;
import com.itmk.security.handler.LoginFailureHandler;
import com.itmk.security.handler.LoginSuccessHandler;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity  //启用Spring Security
public class SpeingSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private CustomerUserDetailsService customerUserDetailsService;
    @Autowired
    private LoginSuccessHandler loginSuccessHandler;
    @Autowired
    private LoginFailureHandler loginFailureHandler;
    @Autowired
    private CustomizeAuthenticationEntryPoint customizeAuthenticationEntryPoint;
    @Autowired
    private CustomAccessDeineHandler customAccessDeineHandler;
    @Bean
    public PasswordEncoder passwordEncoder() {
        // 明文+随机盐值》加密存储
        return new BCryptPasswordEncoder();
    }
    /**
     * 配置权限资源
     * @param http
     * @throws Exception
     */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin()
                .loginProcessingUrl("/api/user/login")
                //  自定义的登录验证成功或失败后的去向

.successHandler(loginSuccessHandler).failureHandler(loginFailureHandler)
                // 禁用csrf防御机制(跨域请求伪造)，这么做在测试和开发会比较方便。
                .and().csrf().disable()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .authorizeRequests()
                .antMatchers("/api/user/login").permitAll()
                .anyRequest().authenticated()
                .and()
                .exceptionHandling()
                .authenticationEntryPoint(customizeAuthenticationEntryPoint)
                .accessDeniedHandler(customAccessDeineHandler);
    }

    /**
     * 配置认证处理器
     * 自定义的UserDetailsService
     * @param auth
     * @throws Exception
     */
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(customerUserDetailsService);
    }
}
```

1.4.10、测试登录认证

1.4.11、登录认证步骤总结：

1.1、自定义UserDetails

当实体对象字段不满足时需要自定义UserDetails，一般都要自定义UserDetails

1.2、自定义UserDetailsService

主要用于从数据库查询用户信息

1.3、创建登录认证成功处理器

认证成功需要返回JSON数据，菜单权限等

1.4、创建登录认证失败处理器

认证失败需要返回JSON数据，给前端判断

1.5、创建匿名用户访问无权限资源时处理器

匿名用户访问时，需要提示JSON

1.6、创建认证过的用户访问无权限资源时的处理器

无权限访问时，需要提示JSON

1.7、配置Spring Security配置类

把上面自定义的处理器交给Spring Security

第30讲 前端登录和后端api接口对接

1.1、安装axios

npm install axios --save

1.2、引入axios

1.2.1、在main.js中引入axios

import axios from 'axios';

1.2.2、在main.js中把axios设为全局变量

Vue.prototype.$http = axios;

1.3、在登录页面Login.vue页面使用axios

```
let parm = {
            username:this.loginForm.username,
            password:this.loginForm.password
          }
let { data: res} = await _this.$http.post("/api/user/login",parm);
```

1.4、配置跨域请求

在项目根目录新建vue.config.js文件

```
module.exports = {
    devServer:{
        open:true,
        port:8082,
        hotOnly:false,
        proxy:{
            '/api':{
                target: "http://127.0.0.1:8089/api",
                changeOrigin:true,
                pathRewrite:{
                    '^/api':''
                }
            }
        }
    }
}
```

1.5、解决后端UsernamePasswordAuthenticationFilter接收不到用户名和密码的问题

由于Spring Security采用form形式接收参数，我们axios提交数据获取不到，那么在axios请求之前做处理

1.5.1、修改main.js

```
axios.interceptors.request.use(config => {
  //解决spring security 不能获取到用户名和密码，验证码的问题
   if(config.url.indexOf('/api/user/login') != -1){
     config.headers['Content-Type'] = 'multipart/form-data';
   }else{
     config.headers['Content-Type'] = 'application/json';
   }
   return config
 })
```

1.5.2、Login.vue修改如下

```
 let datafor = new FormData();
 datafor.append("username", _that.loginForm.username);
 datafor.append("password", _that.loginForm.password);
 datafor.append("code", _that.loginForm.code);
 const { data: res } = await _that.$http.post("/api/user/login", datafor);
```

第31讲 认证成功获取用户权限

当用户登录认证成功之后，需要把用户的权限设置到Spring Security中

1.0、新建Permission实体

```
package com.itmk.system.permission.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;
import lombok.extern.slf4j.Slf4j;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Data
@Slf4j
@TableName(value = "sys_permission")
public class Permission implements Serializable {
    @TableId(type = IdType.AUTO)
    private Long id;
    private Long parentId;
    private String parentName;
    private String label;
    private String code;
    private String path;
    private String name;
    private String url;
    private Integer orderNum;
    private String type;
    private String icon;
    private String remark;
    private Date createTime;
    private Date updateTime;
    private Integer isHome;
    //不是数据库的字段需要排除
    @TableField(exist = false)
    private List<Permission> children = new ArrayList<>();
}
```

在用户实体添加如下字段

```
 //用户权限列表,不属于用户表字段，需要排除
    @TableField(exist = false)
    List<Permission> permissionList;
```

1.1、新建permission的papper接口 PermissionMapper

在PermissionMapper接口新建两个方法，用于查询用户权限；根据用户ID查询权限和根据角色ID查询权限

新建com.itmk.system.permission.mapper.PermissionMapper接口

```
package com.itmk.system.permission.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.itmk.system.permission.entity.Permission;
import org.apache.ibatis.annotations.Param;

import java.util.List;

/**
 * 权限管理
 */
public interface PermissionMapper extends BaseMapper<Permission> {
    /**
     * 根据用户Id查询所有的权限
     * @param userId
     * @return
     */
    List<Permission> selectPermissionByUserId(@Param("userId") Long userId);

    /**
     * 根据角色id查询所有的权限
     * @param roleId
     * @return
     */
    List<Permission> findByRoleId(@Param("roleId") Long roleId);
}
```

1.2、新建PermissionMapper.xml

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.permission.mapper.PermissionMapper">

    <select parameterType="long" id="selectPermissionByUserId"
resultType="com.itmk.system.permission.entity.Permission">
      SELECT DISTINCT
            p.id,
            p.parent_id,
            p. name,
            p. code,
            p.url,
            p.type,
            p.icon,
            p.remark,
            p.create_time,
            p.update_time,
            p.label,
            p.path,
            p.is_home,
            p.order_num
        FROM
            sys_user AS u
        LEFT JOIN sys_user_role AS ur ON u.id = ur.user_id
        LEFT JOIN sys_role AS r ON ur.role_id = r.id
        LEFT JOIN sys_role_permission AS rp ON rp.role_id = r.id
        LEFT JOIN sys_permission AS p ON rp.permission_id = p.id
        WHERE
            u.id = #{userId}
        ORDER BY p.order_num ASC
    </select>

    <select id="findByRoleId"
resultType="com.itmk.system.permission.entity.Permission">
        SELECT
            DISTINCT p.*
        FROM
            sys_permission p
        JOIN sys_role_permission rp ON p.id = rp.permission_id
        JOIN sys_role sr ON rp.role_id = sr.id
        WHERE rp.role_id = #{roleId}
        ORDER BY p.id
    </select>

</mapper>
```

1.3、新建PermissionService

com.itmk.system.permission.service.PermissionService

```
package com.itmk.system.permission.service;
```

```
import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.permission.entity.Permission;

import java.util.List;

public interface PermissionService extends IService<Permission> {
    /**
     * 根据用户Id查询所有的权限
     * @param userId
     * @return
     */
    List<Permission> selectPermissionByUserId(Long userId);

    /**
     * 根据角色id查询所有的权限
     * @param roleId
     * @return
     */
    List<Permission> findByRoleId(Long roleId);
}
```

1.4、新建service实现类 PermissionServiceImpl

```
package com.itmk.system.permission.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.itmk.system.permission.entity.Permission;
import com.itmk.system.permission.mapper.PermissionMapper;
import com.itmk.system.permission.service.PermissionService;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class PermissionServiceImpl extends ServiceImpl<PermissionMapper, Permission>
implements PermissionService {
    @Override
//    @Cacheable(value = "permissions",key = "#userId")
    public List<Permission> selectPermissionByUserId(Long userId) {
        return this.baseMapper.selectPermissionByUserId(userId);
    }

    @Override
//    @Cacheable(value = "permissions",key = "#roleId")
    public List<Permission> findByRoleId(Long roleId) {
        return this.baseMapper.findByRoleId(roleId);
    }
}
```

1.5、新建PermissionController

com.itmk.system.permission.controller.PermissionController

```
/**
 * 权限管理控制器
 */
@Slf4j
@RestController
@RequestMapping("/api/permission")
public class PermissionController {

}
```

1.6、修改CustomerUserDetailsService

```
package com.itmk.security.detailservice;

import com.itmk.system.permission.entity.Permission;
import com.itmk.system.permission.service.PermissionService;
import com.itmk.system.user.entity.SysUser;
import com.itmk.system.user.service.UserService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.stream.Collectors;

@Slf4j
@Component("customerUserDetailsService")
public class CustomerUserDetailsService implements UserDetailsService {
    //注入UserService
    @Autowired
    private UserService userService;
    @Autowired
    private PermissionService permissionService;
    //此处需要注入PasswordEncoder 不然会报错
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        //1.查询用户信息
        SysUser user = userService.getUserByUserName(username);
        //2.用户不存在抛出异常
        if(null == user){
            throw new UsernameNotFoundException("用户名或密码错误!");
        }
        //3.查询用户权限，设置到SysUser 的 authorities 中
        List<Permission> permissions =
permissionService.getPermissionListByUserId(user.getId());
        //4.获取code字段
        List<String> collect = permissions.stream().filter(item -> item !=
null).map(item -> item.getCode()).collect(Collectors.toList());
        //5.转成数组
        String[] codes  = collect.toArray(new String[collect.size()]);
        //6.把codes转成List<GrantedAuthority>
        List<GrantedAuthority> authorityList =
AuthorityUtils.createAuthorityList(codes);
        //7.设置权限
        user.setAuthorities(authorityList);
        //8.设置用户所有菜单
        user.setPermissionList(permissions);
        return user;
    }
}
```

1.7、修改登录认证成功处理器LoginSuccessHandler

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.itmk.jwt.JwtUtils;
import com.itmk.result.ResultUtils;
import com.itmk.system.permission.Vo.MenuVo;
import com.itmk.system.permission.entity.Permission;
import com.itmk.system.user.entity.SysUser;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;
import java.util.stream.Collectors;

/**
 * 登录认证成功处理器
 */
@Component("loginSuccessHandler")
public class LoginSuccessHandler implements AuthenticationSuccessHandler {
    @Autowired
    private JwtUtils jwtUtils;
    @Override
    public void onAuthenticationSuccess(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Authentication authentication) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream out = httpServletResponse.getOutputStream();
        MenuVo vo = new MenuVo();
        //1.获取用户信息
        SysUser user = (SysUser)authentication.getPrincipal();
        //2.生成token
        String token = jwtUtils.generateToken(user);
        vo.setToken(token);
        vo.setUserId(user.getId());
        //3.查询用户菜单权限
        List<Permission> permissionList = user.getPermissionList();
        if(permissionList.size() > 0){
            //设置用户拥有的权限字段
            List<String> auth = permissionList.stream().filter(item -> item !=
null).map(item -> item.getCode()).collect(Collectors.toList());
            vo.setAuthList(auth);
            //获取除按钮以外的菜单
            List<Permission> collect = permissionList.stream().filter(item -> item !=
null && !item.getType().equals("2")).collect(Collectors.toList());
            //生成菜单树数据
            List<Permission> listMenu = makeTree(collect, 0L);
            vo.setMenuList(listMenu);
            //获取路由数据
            List<Permission> routerList = permissionList.stream().filter(item -> item
!= null && item.getType().equals("1")).collect(Collectors.toList());
            vo.setRouterList(routerList);

        }
        String str = JSONObject.toJSONString(ResultUtils.success("认证成功",vo),
SerializerFeature.DisableCircularReferenceDetect);
        out.write(str.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
    /**
     * 组装树
     *
     * @param menuList
     * @param pId
     * @return
     */
    private static List<Permission> makeTree(List<Permission> menuList, Long pId) {

        //子类
        List<Permission> children = menuList.stream().filter(x -> x.getParentId() ==
pId).collect(Collectors.toList());

        //后辈中的非子类
        List<Permission> successor = menuList.stream().filter(x -> x.getParentId() !=
pId).collect(Collectors.toList());
        if (children.size() > 0) {
            children.forEach(x ->
                    {
                        if(successor.size() > 0){
                            makeTree(successor, x.getId()).forEach(
                                    y -> x.getChildren().add(y)
                            );
                        }
                    }
            );
        }
        return children;

    }
}
```

1.8、前端登录设置权限保存

```
if(res.code != 200){
           _this.$message.error(res.msg)
           return;
 }
 let menuList = res.data.menuList;
 let routerList = res.data.routerList;
 let auths = res.data.authList;
```

第32讲 验证码验证讲解

1、验证码实现思路：

在Spring Security认证之前做验证，如果验证码验证失败，直接不做Spring Security认证

2、实现方式

定义一个过滤器，继承OncePerRequestFilter重写doFilterInternal方法，如果验证码错误，直接抛出AuthenticationException类型的异常

3、配置验证码

com.itmk.img_code.ImageCodeConfig

配置验证码配置类如下

```
package com.itmk.img_code;

import com.google.code.kaptcha.Constants;
import com.google.code.kaptcha.impl.DefaultKaptcha;
import com.google.code.kaptcha.util.Config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.Properties;

@Configuration
public class ImageCodeConfig {
    @Bean
    public DefaultKaptcha getDefaultKaptcha(){
        DefaultKaptcha defaultKaptcha = new DefaultKaptcha();
        Properties properties = new Properties();
        //验证码是否有边框
        properties.setProperty(Constants.KAPTCHA_BORDER, "yes");
        //边框颜色
        properties.setProperty(Constants.KAPTCHA_BORDER_COLOR, "192,192,192");
        //验证码图片宽度
        properties.setProperty(Constants.KAPTCHA_IMAGE_WIDTH, "110");
        //验证码图片高度
        properties.setProperty(Constants.KAPTCHA_IMAGE_HEIGHT, "36");
        //字体颜色
        properties.setProperty(Constants.KAPTCHA_TEXTPRODUCER_FONT_COLOR, "blue");
        //字体大小
        properties.setProperty(Constants.KAPTCHA_TEXTPRODUCER_FONT_SIZE, "28");
        //字体样式
        properties.setProperty(Constants.KAPTCHA_TEXTPRODUCER_FONT_NAMES, "宋体");
        //验证码位数
        properties.setProperty(Constants.KAPTCHA_TEXTPRODUCER_CHAR_LENGTH, "4");
        // 图片效果
        properties.setProperty(Constants.KAPTCHA_OBSCURIFICATOR_IMPL,
"com.google.code.kaptcha.impl.ShadowGimpy");
        Config config = new Config(properties);
        defaultKaptcha.setConfig(config);
        return defaultKaptcha;
    }
}
```

4、生成验证码控制器

com.itmk.system.user.controller.LoginController

```
package com.itmk.system.user.controller;

import com.google.code.kaptcha.impl.DefaultKaptcha;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.imageio.ImageIO;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.awt.image.BufferedImage;
import java.io.IOException;
@Slf4j
@RestController
@RequestMapping(value = "/api/login")
public class LoginController {
    public static final String SESSION_KEY = "IMAGE_CODE";
    @Autowired
    private DefaultKaptcha defaultKaptcha;
    /**
     * 获取图形验证码
     */
    @RequestMapping("/image")
    public void imageCode(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        //设置以图片的形式响应
        response.setHeader("Cache-Control", "no-store, no-cache");
        //设置页面缓存方式    不缓存，不存储
        response.setContentType("image/jpeg");
        // 1. 获取验证码字符串
        String code = defaultKaptcha.createText();
        log.info("生成的图形验证码是：" + code);
        // 2. 字符串把它放到session中
        request.getSession().setAttribute(SESSION_KEY , code);
        // 3. 获取验证码图片
        BufferedImage image = defaultKaptcha.createImage(code);
        // 4. 将验证码图片把它写出去
        ServletOutputStream out = response.getOutputStream();
        ImageIO.write(image, "jpg", out);
        if (out != null) {
            out.close();
        }
    }
}
```

5、自定义验证码验证失败异常

该类继承AuthenticationException

com.itmk.security.image_code.ImageCodeException

```
package com.itmk.security.image_code;

import org.springframework.security.core.AuthenticationException;

/**
 * 验证码验证失败异常类
 */
public class ImageCodeException extends AuthenticationException {

    public ImageCodeException(String msg) {
        super(msg);
    }
}
```

6、自定义定义验证码过滤器

com.itmk.security.filte.CheckTokenFilter 该类继承OncePerRequestFilter，

```
package com.itmk.security.filter;

import com.itmk.jwt.JwtUtils;
import com.itmk.security.detailservice.CustomerUserDetailsService;
import com.itmk.security.handler.LoginFailureHandler;
import com.itmk.security.image_code.ImageCodeException;
import com.itmk.system.user.controller.LoginController;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;


@Slf4j
@Component("checkTokenFilter")
public class CheckTokenFilter extends OncePerRequestFilter {
    @Value("${itmk.loginUrl}")
    private String loginUrl;
    @Autowired
    private CustomerUserDetailsService customerUserDetailsService;
    @Autowired
    private LoginFailureHandler loginFailureHandler;
    @Autowired
    private JwtUtils jwtUtils;
    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {
        String url = request.getRequestURI();
        log.info(url);
        if(url.equals(loginUrl)){//如果是登录，则做验证码验证
            try {
                // 校验验证码合法性
                validate(request);
            }catch (AuthenticationException e) {
                // 交给失败处理器进行处理异常
                loginFailureHandler.onAuthenticationFailure(request, response, e);
                // 一定要记得结束
                return;
            }
        }
        // 放行请求
        filterChain.doFilter(request, response);
    }
    private void validate(HttpServletRequest request) {
        // 获取用户输入的验证码
        String inpuCode = request.getParameter("code");
        // 先获取seesion中的验证码
        String sessionCode =

(String)request.getSession().getAttribute(LoginController.SESSION_KEY);
        if(StringUtils.isBlank(inpuCode)) {
            throw new ImageCodeException("验证码不能为空");
        }

        if(!inpuCode.equalsIgnoreCase(sessionCode)) {
            throw new ImageCodeException("验证码输入错误");
        }
    }
}
```

application.yml

```
itmk:
  loginUrl: /api/user/login
```

7、在登录认证失败处理器添加如下代码

```
else if(e instanceof ImageCodeException){
            //验证码异常
            str = e.getMessage();
}
```

8、配置自定义验证码过滤器

在SpeingSecurityConfig中添加如下：

```
http.addFilterBefore(checkTokenFilter, UsernamePasswordAuthenticationFilter.class)
```

9、放开验证码请求

```
.antMatchers("/api/user/login","/api/user/image").permitAll()
```

10、修改前端测试

```
 <img @click="getImage" :src="imgSrc" class="codeImg"/>

 css样式
 .codeImg{
     width: 100%;
     cursor: pointer;
   }


 //获取验证码
   getImage(){
     let res =
       "http://localhost:8082/api/user/image?t=" + new Date().getTime();
     this.imgSrc = res;
   },
```

第33讲 菜单管理接口开发

1.1、获取菜单列表

在PermissionController中添加getMenuList()方法，用户查询菜单列表

```
/**
     * 获取菜单列表
     *
     * @param
     * @return
     */
   @RequestMapping(value = "/getMenuList", method = RequestMethod.POST)
   public ResultVo getMenuList() {
       QueryWrapper<Permission> query = new QueryWrapper<>();
       query.lambda().orderByAsc(Permission::getOrderNum);
       List<Permission> list = permissionService.list(query);
       List<Permission> menuList = null;
       if(!list.isEmpty()){
          menuList = makeTree(list, 0L);
       }
       return ResultUtils.success("成功", CodeStatus.SUCCESS_CODE, menuList);
   }

   /**
     * 组装树
     *
     * @param menuList
     * @param pId
     * @return
     */
   private static List<Permission> makeTree(List<Permission> menuList, Long pId) {
        //子类
        List<Permission> children = menuList.stream().filter(x -> x.getParentId() ==
pId).collect(Collectors.toList());

        //后辈中的非子类
        List<Permission> successor = menuList.stream().filter(x -> x.getParentId() !=
pId).collect(Collectors.toList());
        if (children.size() > 0) {
            children.forEach(x ->
                    {
                        if(successor.size() > 0){
                            makeTree(successor, x.getId()).forEach(
                                    y -> x.getChildren().add(y)
                            );
                        }
                    }
            );
        }
        return children;

    }
```

1.2、新增权限

```
 /**
     * 新增权限
     */
    @RequestMapping(value = "/addPermission",method = RequestMethod.POST)
    public ResultVo addPermission(@RequestBody Permission permission){
        permissionService.save(permission);
        return ResultUtils.success("新增成功");
    }
```

1.3、获取权限上级树

1.3.1、查询permission表type为0和1的数据

1.3.2、新建封装树的实体

1.3.3、组装查询出来的数据为ztree所需的数据

新建树实体com.itmk.system.permission.Vo.TreeVo

```
package com.itmk.system.permission.Vo;

import lombok.Data;

@Data
public class TreeVo {
    //树的id
    private Long id;
    //树的父id
    private Long pid;
    //树的名称
    private String name;
    //是否展开
    private Boolean open;
    //是否选中
    private Boolean checked;
}
```

```
 /**
     * 新增权限，上级菜单树
     * @return
     */
    @RequestMapping(value = "/getParentTree",method = RequestMethod.POST)
    public ResultVo getParentTree(){
        QueryWrapper<Permission> query = new QueryWrapper<>();
        query.lambda().eq(Permission::getType,"0").or().eq(Permission::getType,"1");
        List<Permission> list = permissionService.list(query);
        List<TreeVo> listTree = new ArrayList<>();
        TreeVo parentTree = new TreeVo();
        parentTree.setId(0L);
        parentTree.setPid(-1L);
        parentTree.setName("顶级菜单");
        parentTree.setOpen(true);
        parentTree.setChecked(false);
        listTree.add(parentTree);
        if(list.size() > 0){
            for(Permission p : list){
                if(p != null){
                    TreeVo tree = new TreeVo();
                    tree.setId(p.getId());
                    tree.setPid(p.getParentId());
                    tree.setName(p.getLabel());
                    tree.setOpen(true);
                    tree.setChecked(false);
                    listTree.add(tree);
                }
            }
        }

        return ResultUtils.success("成功",listTree);
    }
```

1.4、编辑权限

1.4.1、根据id查询要编辑的数据

```
  /**
     * 根据id查询菜单
     * @param permission
     * @return
     */
    @RequestMapping(value = "getMenuById",method = RequestMethod.POST)
    public ResultVo getMenuById(@RequestBody Permission permission){
        Permission menu = permissionService.getById(permission.getId());
        return ResultUtils.success("成功",menu);
    }
```

1.4.2、编辑权限保存

```
/**
```

```
     * 根据id更新权限
     * @param permission
     * @return
     */
    @RequestMapping(value = "/editSave",method = RequestMethod.POST)
    public ResultVo editSave(@RequestBody Permission permission){
        permission.setCreateTime(new Date());
        boolean res = permissionService.updateById(permission);
        if(res){
            return ResultUtils.success("更新成功");
        }else{
            return ResultUtils.error("更新失败");
        }

    }
```

1.5、删除权限

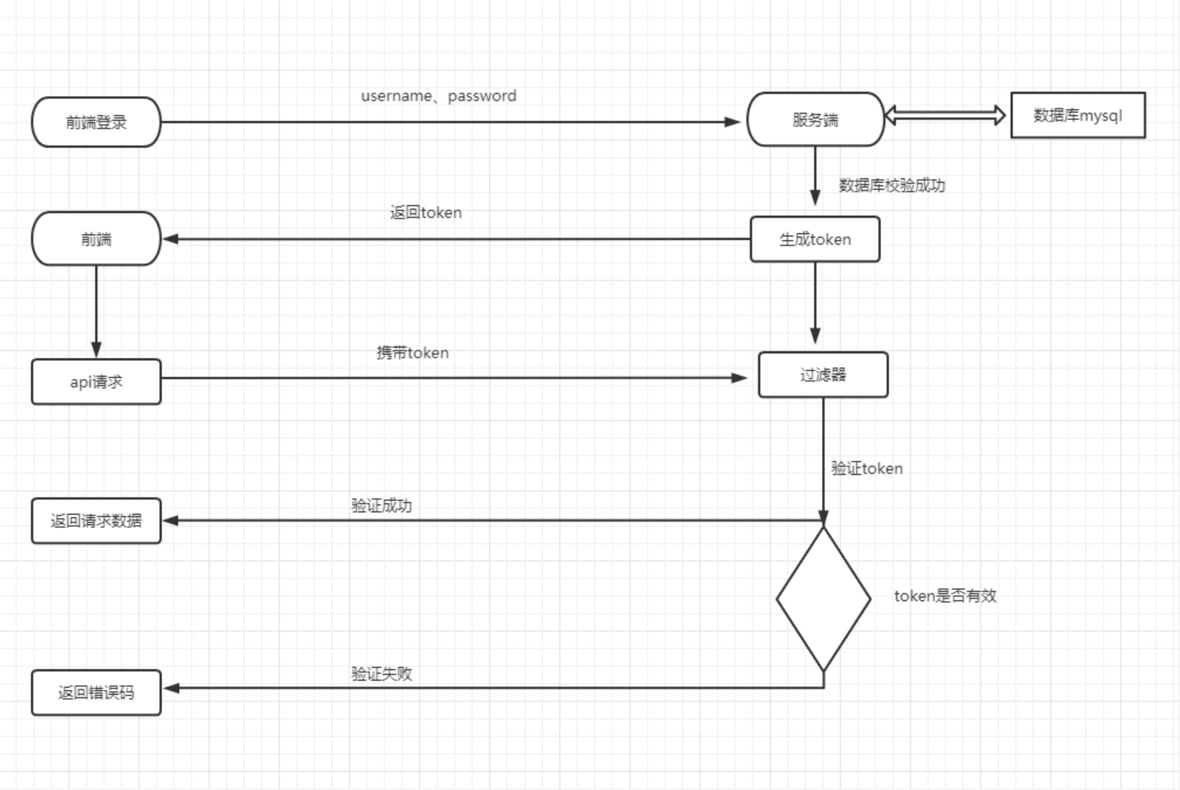
```
/**
 * 删除权限
 * @return
 */
@RequestMapping(value = "/deleteEntity",method = RequestMethod.POST)
public ResultVo deleteEntity(@RequestBody Permission permission){

    boolean b = permissionService.removeById(permission.getId());
    if(b){
        return ResultUtils.success("删除成功!");
    }else{
        return ResultUtils.error("删除失败!");
    }

}
```

第34讲 token验证处理

1.1、什么是token

Token是服务端生成的一串字符串，以作客户端进行请求的一个令牌，当第一次登录后，服务器生成一个Token便将此Token返回给客户端，以后客户端只需带上这个Token前来请求数据即可，无需再次带上用户名和密码。

1.2、token认证流程图



1.3、在CheckTokenFilter中定义token验证

```
package com.itmk.security.filte;

import com.itmk.jwt.JwtUtils;
import com.itmk.security.detailservice.CustomerUserDetailsService;
import com.itmk.security.handler.LoginFailureHandler;
import com.itmk.security.image_code.ImageCodeException;
import com.itmk.system.user.controller.UserController;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
@Slf4j
@Component("checkTokenFilter")
public class CheckTokenFilter extends OncePerRequestFilter {
    @Value("${itmk.loginUrl}")
    private String loginUrl;
    @Value("${itmk.imgUrl}")
    private String imgUrl;
    @Autowired
    private LoginFailureHandler loginFailureHandler;
    @Autowired
    private JwtUtils jwtUtils;
    @Autowired
    private CustomerUserDetailsService customerUserDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {
        String url = request.getRequestURI();
        if(url.equals(loginUrl)){
            try{
                validate(request);
            }catch (AuthenticationException e){
                loginFailureHandler.onAuthenticationFailure(request,response,e);
                return;
            }
        }else {
            //验证token,验证码请求不需要验证token
            String imgurl = request.getRequestURI();
            if(!imgurl.equals(imgUrl)){
                try{
                    validateToken(request);
                }catch (AuthenticationException e){
                    loginFailureHandler.onAuthenticationFailure(request,response,e);
                    return;
                }
            }
        }
        filterChain.doFilter(request,response);
    }
    //验证token
    private void validateToken(HttpServletRequest request){
        //获取前端传来的token
        String token = request.getHeader("token");
        //解析token，获取用户名
        String username = jwtUtils.getUsernameFromToken(token);
        //如果token或者用户名为空的话，不能通过认证
        if(StringUtils.isBlank(token) || StringUtils.isBlank(username)){
            throw new ImageCodeException("token验证失败!");
        }
        UserDetails userDetails =
customerUserDetailsService.loadUserByUsername(username);
        if(userDetails == null){
            throw new ImageCodeException("token验证失败!");
        }
        UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());
        authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
        //设置为已登录
        SecurityContextHolder.getContext().setAuthentication(authentication);

    }
    //验证验证码
    private void validate(HttpServletRequest request){
        //1.获取登录请求的验证码
        String inputCode = request.getParameter("code");
        //2.获取Session中的验证码
        String  code =
(String)request.getSession().getAttribute(UserController.SESSION_KEY);
        //3.判断验证码是否为空
        if(StringUtils.isBlank(inputCode)){
            throw new ImageCodeException("验证码不能为空!");
        }
        //4.判断验证码是否相等
        if(!inputCode.equalsIgnoreCase(code)){
            throw new ImageCodeException("验证码输入错误!");
        }
    }
}
```

1.4、前端项目设置携带token

在main.js中添加添加token

```
axios.interceptors.request.use(config => {
 //解决spring security 不能获取到用户名和密码，验证码的问题
  if(config.url.indexOf('/api/user/login') != -1){
    config.headers['Content-Type'] = 'multipart/form-data';
  }else{
    config.headers['Content-Type'] = 'application/json';
  }
  // 为请求头添加token字段
  config.headers.token = sessionStorage.getItem('token')
  return config
})
```

第35讲 菜单管理之新增数据接口对接

1.1、树形列表对接

1.1.1、在methods里面添加如下代码

```
 //获取权限列表
   async getMenuList() {
     let { data: res } = await this.$http.post("/api/permission/getMenuList");
     this.tableTreeDdata = res.data;
   },
```

1.1.2、在created()方法中添加如下代码

```
 this.getMenuList();
```

1.2、新增权限

1.2.1、为新增按钮添加点击事件

```
@click="addPermission()"
```

```
 //打开新增权限对话框
    addPermission() {
      this.editTag = "0";
      this.boxTitle = "新增权限";
      this.dialogFormVisible = !this.dialogFormVisible;
      //新增是清空数据
      this.resetForm("addMenu");
    },
```

```
 //解决重置表单时报 'resetFields' of undefined的错
    resetForm(formName) {
      if (this.$refs[formName]) {
        this.$refs[formName].resetFields();
      }
    },
```

1.2.2、获取上级菜单数据

```
// 获取上级菜单树数据
    async getParentTree() {
      let { data: res } = await this.$http.post("api/permission/getParentTree");
      console.log(res);
      this.nodes = res.data;
    }
```

在created()方法中添加getParentTree()方法

```
this.getParentTree();
```

1.3、上级菜单树弹框布局与配置

1.3.1、引入ztree   import tree from "vue-giant-tree";

1.3.2、注册ztree树组件

```
 components: {
    tree
  },
```

1.3.3、配置树

```
ztreeObj: null,
      setting: {
        // check: {
        //   enable: true
        // },
        data: {
          simpleData: {
            enable: true,
            idKey: "id",
            pIdKey: "pid",
            rootPId: "0"
          }
        },
        callback: {
          onClick: this.ztreeOnClick
```

```
    }
   },
```

```
 // 菜单树点击事件
  ztreeOnClick: function(evt, treeId, treeNode) {
   this.permissions.parentName = treeNode.name;
   this.permissions.parentId = treeNode.id;
  },
```

```
 //加载树时执行
  handleCreated: function(ztreeObj) {
   console.log("加载树");
   this.ztreeObj = ztreeObj;

   console.log(this.ztreeObj);
   //  let firstTree = ztreeObj.getNodes()[0];
   //默认选中第一个
   // ztreeObj.selectNode(firstTree);
   //设置节点全部展开
   ztreeObj.expandAll(true);
   //加载完自动点击第一个，加载右边表格
   // this.setting.callback.onClick(null, firstTree.id, firstTree);
  },
```

1.3.4、点击树弹框

```
 <!-- 选择上级菜单树弹框 -->
  <el-dialog width="25%" title="上级菜单" :visible.sync="innerVisible" append-to-body>
   <tree :nodes="nodes" :setting="setting" @onCreated="handleCreated" />
   <div slot="footer" class="dialog-footer">
    <el-button @click="innerVisible = false">取 消</el-button>
    <el-button type="primary" @click="getCheckedNodes">确 定</el-button>
   </div>
  </el-dialog>
```

1.3.5、树弹框确认事件

```
 //上级树确认事件
  getCheckedNodes() {
   this.innerVisible = false;
  },
```

1.4、新增必填字段验证

注意 要添加 prop=""属性

```
addMenuValdate: {
    label: [
     { required: true, trigger: "change", message: "请填写权限名称" }
    ],
    parentName: [
     { required: true, trigger: "change", message: "请选择上级菜单" }
    ],
    name: [
     { required: true, trigger: "change", message: "请填写路由名称" }
    ],
    path: [
     { required: true, trigger: "change", message: "请填写路由地址" }
    ],
    url: [{ required: true, trigger: "change", message: "请填写组件路径" }],
    code: [{ required: true, trigger: "change", message: "请填写权限标识" }]
  },
```

1.5、提交新增

```
//提交新增权限
    async addMenuBtn() {
      let _this = this;
      _this.$refs.addMenu.validate(async valid => {
        if (valid) {
          let url = "";
          if (_this.editTag == "0") {
            //新增
            url = "/api/permission/addPermission";
          } else {
            url = "/api/permission/editSave"; //编辑
          }
          let parm = _this.permissions;
          let { data: res } = await _this.$http.post(url, parm);
          if (res.code == 200) {
            //关闭弹框
            _this.dialogFormVisible = false;
            _this.getMenuList();
            _this.getParentTree();
          }
          _this.$message({
            message: res.msg,
            type: "success"
          });
        }
      });
    },
```

第36讲 token验证失败处理

######

1.0.1、新建token异常处理类com.itmk.security.exception.TokenException

```
package com.itmk.security.exception;

import org.springframework.security.core.AuthenticationException;

/**
 * token异常处理类
 */
public class TokenException extends AuthenticationException {
    public TokenException(String msg) {
        super(msg);
    }
}
```

1.0.2、修改CheckTokenFilter过滤器 抛出异常为 TokenException

```
  //如果token或者用户名为空的话，不能通过认证
        if(StringUtils.isBlank(token) || StringUtils.isBlank(username)){
            throw new TokenException("token验证失败!");
        }
        UserDetails userDetails =
customerUserDetailsService.loadUserByUsername(username);
        if(userDetails == null){
            throw new TokenException("token验证失败!");
        }
```

1.0.3、修改LoginFailureHandler认证失败处理器

```
 int code = 500;
 else if(e instanceof TokenException){
            //token异常
            code = 600;
            str = e.getMessage();
 }
```

1.0.4、修改ResultUtils

```
public static ResultVo error(String msg,int code,Object data){
        return vo(msg, code,data);
    }

      public static ResultVo success(String msg,int code,Object data){
        return vo(msg, code,data);
    }
```

1.0.5、修改main.js

```
// 接口数据返回时，如果后台返回token过期，那么需要重新登录
// 响应拦截器
axios.interceptors.response.use(
  response => {
    console.log(response);
    // 如果返回的状态码为200，说明接口请求成功，可以正常拿到数据
    // 否则的话抛出错误
    if (response.status === 200) {
      if (response.data.code == 600) {
        sessionStorage.clear();
        window.location.href = '/login';
        return response;
      } else {
        return Promise.resolve(response);
      }

    } else {
      return Promise.reject(response);
    }
  },
  // 服务器状态码不是2开头的的情况
  // 这里可以跟你们的后台开发人员协商好统一的错误状态码
  // 然后根据返回的状态码进行一些操作，例如登录过期提示，错误提示等等
  // 下面列举几个常见的操作，其他需求可自行扩展
  error => {
    if (error.response.status) {
      switch (error.response.status) {
        // 401: 未登录
        // 未登录则跳转登录页面，并携带当前页面的路径
        // 在登录成功后返回当前页面，这一步需要在登录页操作。
        case 401:
          router.replace({
            path: '/login',
            query: {
              redirect: router.currentRoute.fullPath
            }
          });
          break;

        // 403 token过期
        // 登录过期对用户进行提示
        // 清除本地token和清空vuex中token对象
        // 跳转登录页面
        case 405:
          ElementUI.Message({
            message: '请求方式错误',
            type: 'error'
          });
          break;
        // 404请求不存在
        case 404:
          ElementUI.Message({
            message: '网络请求不存在',
            type: 'error'
          });
          break;
        // 其他错误，直接抛出错误提示
        default:
          ElementUI.Message({
            message: error.response.data.msg,
            type: 'error'
          });
          if (error.response.data.code == 600) {
            sessionStorage.clear();
            window.location.href = '/login';
          }
      }
      return Promise.reject(error.response);
    }
  }
);
```

第37讲 菜单管理之编辑、删除接口对接

1.1、编辑按钮点击事件

```
@click="editMenu(scope.row)"
```

1.2、获取编辑数据

添加 editTag 标准

```
    editMenu(item) {
      this.editTag = "1";
      this.boxTitle = "编辑权限";
      this.dialogFormVisible = true;
      this.resetForm("addMenu");

      let row = item;
      this.getMenuById(row.id);
      console.log(row);
    },
    async getMenuById(editId) {
      let { data: res } = await this.$http.post("/api/permission/getMenuById", {
        id: editId
      });
      if (res.code == 200) {
        console.log(res.data.label);
        this.permissions.id = res.data.id;
        this.permissions.code = res.data.code;
        this.permissions.icon = res.data.icon;
        this.permissions.label = res.data.label;
        this.permissions.name = res.data.name;
        this.permissions.orderNum = res.data.orderNum;
        this.permissions.parentId = res.data.parentId;
        this.permissions.parentName = res.data.parentName;
        this.permissions.path = res.data.path;
        this.permissions.type = res.data.type;
        this.permissions.url = res.data.url;
      }
    },
```

1.3、编辑保存

```
 //提交新增权限
    async addMenuBtn() {
      let _this = this;
      _this.$refs.addMenu.validate(async valid => {
        if (valid) {
          let url = "";
          if (_this.editTag == "0") {
            //新增
            url = "/api/permission/addPermission";
          } else {
            url = "/api/permission/editSave"; //编辑
          }
          let parm = _this.permissions;
          let { data: res } = await _this.$http.post(url, parm);
          if (res.code == 200) {
            //关闭弹框
```

```
              _this.dialogFormVisible = false;
              _this.getMenuList();
              _this.getParentTree();
            }
            _this.$message({
              message: res.msg,
              type: "success"
            });
          }
        });
      },
```

1.4、删除权限

1.4.1、添加删除事件

```
@click="handleDelete(scope.row)
```

1.4.2、删除提交

```
  //删除权限
    handleDelete(row) {
      let _this = this;
      this.$confirm("确定删除吗 ?", "系统提示", {
        confirmButtonText: "确定",
        cancelButtonText: "取消",
        type: "warning"
      }).then(async () => {
        let parm = {
          id: row.id
        };
        let { data: res } = await this.$http.post(
          "/api/permission/deleteEntity",
          parm
        );
        if (res.code == 200) {
          _this.getMenuList();
          _this.getParentTree();
        }
        _this.$message({
          message: res.msg,
          type: "success"
        });
      });
    },
```

第38讲 角色管理接口开发讲解

1.1、新增角色

1.1.1、新建SysRole实体,在system目录下建role目录,在role目录下建entity目录，然后新建SysRole实体com.itmk.system.role.entity.SysRole

```
package com.itmk.system.role.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;

import java.io.Serializable;
import java.util.Date;


@Data
@TableName(value = "sys_role")
public class SysRole implements Serializable {
    //主键
    @TableId(type = IdType.AUTO)
    private Long id;
    //角色名称
    private String name;
    //角色说明
    private String remark;
    //创建时间
    private Date createTime;
    //更新时间
    private Date updateTime;
}
```

1.1.2、在role目录下新建mapper目录,新建数据访问层 RoleMapper接口

com.itmk.system.role.mapper.RoleMapper

```
/**
 * 角色mapper
 */
public interface RoleMapper extends BaseMapper<SysRole> {
}
```

1.1.3、在resources目录下mapper目录新建RoleMapper映射文件RoleMapper.xml文件

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.role.mapper.RoleMapper">

</mapper>
```

1.1.4、在role目录新建service目录，新建service服务层RoleService接口

com.itmk.system.role.service.RoleService

```
package com.itmk.system.role.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.role.entity.SysRole;

/**
 * 角色管理service
 */
public interface RoleService extends IService<SysRole> {
}
```

1.1.5、在service目录新建Impl目录，新建RoleService实现类com.itmk.system.role.service.Impl.RoleServiceImpl

```
package com.itmk.system.role.service.Impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.itmk.system.role.entity.SysRole;
import com.itmk.system.role.mapper.RoleMapper;
import org.springframework.stereotype.Service;

@Service
public class RoleServiceImpl extends ServiceImpl<RoleMapper, SysRole> {
}
```

1.1.6、在role目录新建controller目录,新建RoleController控制器，并编写新增代码

com.itmk.system.role.controller.RoleController

```
package com.itmk.system.role.controller;

import com.itmk.result.ResultUtils;
import com.itmk.result.ResultVo;
import com.itmk.system.role.entity.SysRole;
import com.itmk.system.role.service.RoleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/api/role")
public class RoleController {
    @Autowired
    private RoleService roleService;

    /**
     * 新增角色
     * @param role
     * @return
     */
    @RequestMapping(value = "addRole",method = RequestMethod.POST)
    public ResultVo addRole(@RequestBody SysRole role){
        boolean b = roleService.save(role);
        if(b){
            return ResultUtils.success("新增成功!");
        }else{
            return ResultUtils.error("新增失败！ ");
        }
    }
}

```

1.2、根据id查询角色

```
  /**
     * 根据id查询角色
     * @return
     */
    @RequestMapping(value = "/getRoleById",method = RequestMethod.POST)
    public ResultVo getRoleById(@RequestBody SysRole sysRole){
        SysRole role = roleService.getById(sysRole.getId());
        return ResultUtils.success("成功",role);

    }
```

1.3、编辑角色

```
/**
     * 编辑角色
     * @return
     */
    @RequestMapping(value = "/updateRole",method = RequestMethod.POST)
    public ResultVo updateRole(@RequestBody SysRole sysRole){
        boolean b = roleService.updateById(sysRole);
        if(b){
            return ResultUtils.success("编辑角色成功!");
        }else{
            return ResultUtils.error("编辑角色失败!");
        }
    }
```

1.4、删除角色

```
 /**
     * 删除角色
     * @return
     */
    @RequestMapping(value = "/deleteRole",method = RequestMethod.POST)
    public ResultVo deleteRole(@RequestBody SysRole sysRole){
        boolean b = roleService.removeById(sysRole.getId());
        if(b){
            return ResultUtils.success("删除角色成功!");
        }else{
            return ResultUtils.error("删除角色失败!");
        }

    }
```

第39讲 角色管理前端接口对接

1.1、角色列表

在data中添加 currentPage 和pageSize变量，用于接收当前页和页容量

```
currentPage: 1, //当前页
pageSize:10, //页容量
```

在methods中添加如下方法

```
 //查询table列表
    async getRoleList(){
      let parm = {
        currentPage:this.currentPage,
        pageSize:this.pageSize
      }
      let {data:res} = await this.$http.post("/api/role/getRoleList",parm);
      if(res.code == 200){
        this.currentPage = res.data.current;
        this.pageSize = res.data.size;
        this.tableData =  res.data.records;
      }
    },
```

1.2、添加角色

```
 //确认新增或编辑
    confirmBtn() {
      let _this = this;
      _this.$refs.addRole.validate(async valid => {
        if (valid) {
          let {data:res} = await
_this.$http.post("/api/role/addRole",_this.addRoleForm);
          if(res.code == 200){
            //信息提示
            _this.$message({
              message:res.msg,
              type:'success'
            })
            //刷新数据
            _this.getRoleList();
            //关闭弹框
             _this.visible = false;

          }else{
             //信息提示
            _this.$message({
              message:res.msg,
              type:'error'
            })
          }

        }
      });
    },
```

1.3、编辑角色

```
  //编辑角色事件
   editRole(row) {
     this.resetForm("addRole");
     this.dialogTitle = "新增角色";
     this.visible = true;
     //查询编辑的数据
     this.getRoleById(row.id);
   },
```

```
//根据id查询编辑的数据
   async getRoleById(id){
     let parm = {
       id:id
     }
     let {data:res} = await this.$http.post("/api/role/getRoleById",parm);
     if(res.code == 200){
       this.addRoleForm.name = res.data.name;
       this.addRoleForm.remark = res.data.remark;
     }
   },
```

```
//确认新增或编辑
   confirmBtn() {
     let _this = this;
     _this.$refs.addRole.validate(async valid => {
       if (valid) {
         let url = "";
         if(_this.editTag == "0"){
           url = "/api/role/addRole";
         }else{
            url = "/api/role/updateById";
         }
         let {data:res} = await _this.$http.post(url,_this.addRoleForm);
         if(res.code == 200){
           //信息提示
           _this.$message({
             message:res.msg,
             type:'success'
           })
           //刷新数据
           _this.getRoleList();
           //关闭弹框
            _this.visible = false;

         }else{
            //信息提示
           _this.$message({
             message:res.msg,
             type:'error'
           })
         }

       }
     });
   },
```

1.4、删除角色

```
 //删除角色
    deleteRole(row) {
      let _this = this;
      this.$confirm("确认删除吗？","系统提示",{
        confirmButtonText:'确定',
        cancelButtonText:'取消',
        type:'waring'
      }).then(async () =>{
        let parm = {
          id:row.id
        }
        let {data:res} = await _this.$http.post("/api/role/deleteRole",parm);
        if(res.code == 200){
          _this.getRoleList();
        }
        _this.$message({
          message:res.msg,
          type:'success'
        })
      })
    },
```

第40讲 部门管理接口开发讲解

1、新增部门

1.1、在system下新建department目录，然后新建Department实体

```
package com.itmk.system.department.entity;

import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;

import java.io.Serializable;

@Data
@TableName(value = "sys_dept")
public class Department implements Serializable {
    //主键
    @TableId
    private String id;
    //上级部门id
    private String pid;
    //上级部门id集合
    private String likeId;
    //上级部门名称
    private String parentName;
    //部门经理
    private String manager;
    //部门名称
    private String name;
    //部门编码
    private String deptCode;
    //部门地址
    private String deptAddress;
    //部门电话
    private String deptPhone;
    //序号
    private Integer orderNum;
}
```

1.2、新建mapper目录，新建DepartmentMapper

```
package com.itmk.system.department.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.itmk.system.department.entity.Department;

/**
 * 部门管理mapper接口
 */
public interface DepartmentMapper extends BaseMapper<Department> {
}
```

1.3、新建DepartmentMapper.xml映射文件

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.department.mapper.DepartmentMapper">

</mapper>
```

1.4、在department目录下新建service目录，新建DepartmentService接口

```
package com.itmk.system.department.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.department.entity.Department;

/**
 * 部门service服务接口
 */
public interface DepartmnetService extends IService<Department> {
}
```

1.5、在service下新建Impl目录,新建DepartmentService实现类

```
package com.itmk.system.department.service.Impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.itmk.system.department.entity.Department;
import com.itmk.system.department.mapper.DepartmentMapper;
import com.itmk.system.department.service.DepartmnetService;
import org.springframework.stereotype.Service;

@Service
public class DepartmnetServiceImpl extends ServiceImpl<DepartmentMapper, Department>
implements DepartmnetService {
}
```

1.6、department目录新建DepartmentController控制器

```
package com.itmk.system.department.controller;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.itmk.result.ResultUtils;
import com.itmk.result.ResultVo;
import com.itmk.system.department.entity.Department;
import com.itmk.system.department.service.DepartmnetService;
import com.itmk.system.department.vo.DepartmentVo;
import com.itmk.uuid.UUIDUtil;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@Slf4j
@RestController
@RequestMapping(value = "/api/department")
public class DepartmentController {
    @Autowired
    private DepartmnetService departmnetService;

    /**
     * 新增部门
     * @param department
     * @return
     */
    @RequestMapping(value = "/addDepartment",method = RequestMethod.POST)
    public ResultVo addDepartment(@RequestBody Department department){
        String id = UUIDUtil.getUniqueIdByUUId();
        department.setId(id);
        boolean b = departmnetService.save(department);
        if(b){
            return ResultUtils.success("新增部门成功!");
        }else{
            return ResultUtils.error("新增部门失败!");
        }

    }
}
```

```
package com.itmk.utils;


import java.util.UUID;

public class UUIDUtil {

    private static final int SHORT_LENGTH = 8;

    public static String uuid() {
        String str = UUID.randomUUID().toString();
        String temp = str.replace("-","");
        return temp;
    }

    public static String getUniqueIdByUUId() {
        //最大支持1-9个集群机器部署
        int machineId = 1;
        int hashCodeV = UUID.randomUUID().toString().hashCode();
        if(hashCodeV < 0) {
            hashCodeV = - hashCodeV;
        }
        // 0 代表前面补充0
        // 4 代表长度为4
        // d 代表参数为正数型
        return machineId + String.format("%015d", hashCodeV);
    }
    public static void main(String[] args) {
        System.out.println(getUniqueIdByUUId());
        System.out.println(uuid());
    }



    public static String[] chars = new String[] { "a", "b", "c", "d", "e", "f",
            "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",
            "t", "u", "v", "w", "x", "y", "z", "0", "1", "2", "jqGrid-4.4.3", "4",
"5",
            "6", "7", "8", "9", "A", "B", "C", "D", "E", "F", "G", "H", "I",
            "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
            "W", "X", "Y", "Z" };


    public static String generateShortUuid() {
        StringBuffer shortBuffer = new StringBuffer();
        String uuid = UUID.randomUUID().toString().replace("-", "");
        for (int i = 0; i < SHORT_LENGTH; i++) {
            String str = uuid.substring(i * 4, i * 4 + 4);
            int x = Integer.parseInt(str, 16);
            shortBuffer.append(chars[x % 0x3E]);
        }
        return shortBuffer.toString();

    }



}
```

2、查询部门列表

```
/**
     * 获取部门列表
     * @param departmentVo
     * @return
     */
    @RequestMapping(value = "/getDepartmentList")
    public ResultVo getDepartmentList(@RequestBody DepartmentVo departmentVo){
        //1.生成sql条件构造器
        QueryWrapper<Department> query = new QueryWrapper<>();

query.lambda().like(Department::getName,departmentVo.getName()).eq(Department::getPid,
departmentVo.getDeptId());
        //2.设置分页数据
        IPage<Department> page = new Page<>();
        page.setCurrent(departmentVo.getCurrentPage());
        page.setSize(departmentVo.getPageSize());
        IPage<Department> respage = departmnetService.page(page, query);
        return ResultUtils.success("查询成功",respage);
    }
```

3、编辑部门

3.1、查询编辑的数据

```
  /**
     * 根据id查询部门数据
     * @param department
     * @return
     */
    @RequestMapping(value = "/getDepartmentById",method = RequestMethod.POST)
    public ResultVo getDepartmentById(@RequestBody Department department){
        Department res = departmnetService.getById(department.getId());
        return ResultUtils.success("查询成功",res);
    }
```

```
  /**
     * 编辑部门保存
     * @param department
     * @return
     */
    @RequestMapping(value = "/updateDepartmentById",method = RequestMethod.POST)
    public ResultVo updateDepartmentById(@RequestBody Department department){
        boolean b = departmnetService.save(department);
        if(b){
            return ResultUtils.success("编辑成功!");
        }else{
            return ResultUtils.error("编辑失败!");
        }
    }
```

4、删除部门

```
  /**
     * 根据id删除部门
     * @return
     */
    @RequestMapping(value = "deleteDepartmentById",method = RequestMethod.POST)
    public ResultVo deleteDepartmentById(@RequestBody Department department){
        boolean b = departmnetService.removeById(department.getId());
        if(b){
            return ResultUtils.success("删除部门成功!");
        }else{
            return ResultUtils.error("删除部门失败!");
        }

    }
```

5、查询左侧部门树

```
/**
     * 获取左侧部门树
     * @return
     */
    @RequestMapping(value = "/getDeptTree",method = RequestMethod.POST)
    public ResultVo getDeptTree(){
        List<SysDept> list = sysDeptServie.list();
        return  ResultUtils.success("成功",list);
    }
```

6、新增部门获取上级部门树

```
/**
     * 新增部门获取上级部门树
     * @return
     */
    @RequestMapping(value = "/getParentTree",method = RequestMethod.POST)
    public ResultVo getParentTree(){
        //获取列表
        List<SysDept> list = sysDeptServie.list();
        SysDept sysDept = new SysDept();
        sysDept.setId("0");
        sysDept.setPid("-1");
        sysDept.setName("顶级部门");
        sysDept.setLikeId("0,");
        list.add(0,sysDept);
        return ResultUtils.success("成功",list);
    }
```

第41讲 前端部门管理接口对接

1、左侧部门树对接

当左侧树加载完成时，默认选中第一个节点，根据选中的第一个节点的id查询右边的列表数据

```
 //树创建成功之后调用
    handleCreated(treeObj) {
      this.ztreeObj = treeObj;
      treeObj.expandAll(true);
      let firstTree = this.ztreeObj.getNodes()[0];
      //默认选中第一个
      this.ztreeObj.selectNode(firstTree);
      //加载完自动点击第一个，加载右边表格
      if (firstTree) {
        //此处需要判断，否则会报错
        this.setting.callback.onClick(null, firstTree.id, firstTree);
      }
```

```
 //获取左侧部门树
    async getLeftTree() {
      let { data: res } = await this.$http.post(
        "/api/department/getDepartmentTree"
      );
      if (res.code == 200) {
        this.nodes = res.data;
        console.log(res);
      }
    },
```

```
created(){
this.getLeftTree();
}
```

```
 //树的点击事件
    ztreeOnClick(evt, treeId, treeNode) {
      this.deptId = treeNode.id;
      this.getDepartmentList();
      console.log(evt);
      console.log(treeId);
      console.log(treeNode);
      //
    },
```

2、列表接口对接

2.1、定义分页参数

```
  //当前页数
      currentPage: 1,
      pageSize:10,
```

2.1、查询列表数据

```
 //获取table列表数据
    async getDepartmentList(){
      let _this = this;
      let parm ={
        deptId:_this.deptId,
        currentPage:_this.currentPage,
        pageSize:_this.pageSize
      }
      let {data:res} = await
_this.$http.post("/api/department/getDepartmentList",parm);
      if(res.code == 200){
        console.log('111111111')
        console.log(res);
        _this.tableData = res.data.records;
      }
    },
```

3、新增部门

新增完部门需要刷新左侧部门树和右侧列表

3.0、加载上级部门树

```
//加载上级部门树
    async getParentTree() {
      let { data: res } = await this.$http.post(
        "/api/sysDept/getParentTree",
        null
      );
      if (res.code == 200) {
        this.parentNodes = res.data;
      }
    },
```

```
 //上级部门树点击事件
    ztreeParentOnClick(evt, treeId, treeNode) {
      console.log(treeNode);
      this.deptForm.pid = treeNode.id;
      this.deptForm.deptId = treeNode.likeId;
      this.deptForm.parentName = treeNode.name;
    },
```

3.1、新增弹框事件

```
  //新增部门打开弹框
    addDept() {
      this.editTag = "0";
      //清空表单数据
      this.resetForm("deptForm");
      //设置表单标题
      this.dialogTitle = "新增部门";
      //打开弹框
      this.dialogVisible = true;
    },
```

3.2、新增保存

```
//新增部门
    async addDeptSave() {
      let _this = this;
      //1.验证表单
      _this.$refs.deptForm.validate(async valid => {
        if (valid) {
          let url = "";
          if (_this.editTag == "0") {
            url = "/api/sysDept/addDept";
          }
          let parm = _this.deptForm;
          let { data: res } = await _this.$http.post(url, parm);
          if (res.code == 200) {
            //刷新左侧树数据
            _this.getLeftDeptTree();
            //刷新新增部门上级部门树
            _this.getParentTree();
            //刷新表格数据
            _this.getDeptListByLikeId(_this.deptForm.pid);
            //关闭弹框
            _this.dialogVisible = false;
            _this.$message({
              message: res.msg,
              type: "success"
            });
          } else {
            _this.$message({
              message: res.msg,
              type: "error"
            });
          }
        }
      });
    },
```

4、编辑部门

4.1、打开弹框，需要查询要编辑的数据

```
//编辑机构
    handleTableEdit(index, row) {
      //显示表单
      this.dialogVisible = !this.dialogVisible;
      //1.设置标志为编辑 0 新增 1 编辑
      this.editTag = "1";
      //2.清空表单
      this.resetForm("deptForm");
      //3.设置弹框标题
      this.dialogTitle = "编辑部门";
      //4.根据id查询需要编辑的数据
      this.getDeptById(row.id);
    },
```

4.2、查询需要编辑的数据

```
//根据id查询编辑部门的数据
  async getDeptById(editId) {
    let parm = {
      id: editId
    };
    let { data: res } = await this.$http.post(
      "/api/sysDept/getDeptById",
      parm
    );
    if (res.code == 200) {
      this.deptForm = res.data;
    }
  },
```

4.3、设置编辑之前上级选中数据

```
//注意，编辑设置默认选中节点时，需要在created方法里面设置，不能在input
  //点击事件那里
  handleParentCreated: function(parentZtreeObj) {
    this.parentZtreeObj = parentZtreeObj;
    //根据原来选中的id来找到要选中的节点
    var node = this.parentZtreeObj.getNodeByParam("id", this.deptForm.pid);
    //把找到的节点设为选中状态
    this.parentZtreeObj.selectNode(node);
    //设置节点全部展开
    parentZtreeObj.expandAll(true);
  },
```

4.4、编辑保存

```
//新增部门
  async addDeptSave() {
    let _this = this;
    //1.验证表单
    _this.$refs.deptForm.validate(async valid => {
      if (valid) {
        let url = "";
        if (_this.editTag == "0") {
          url = "/api/sysDept/addDept";
        } else {
          url = "/api/sysDept/updateDept";
        }
        let parm = _this.deptForm;
        let { data: res } = await _this.$http.post(url, parm);
        if (res.code == 200) {
          //刷新左侧树数据
          _this.getLeftDeptTree();
          //刷新新增部门上级部门树
          _this.getParentTree();
          //刷新表格数据
          _this.getDeptListByLikeId(_this.deptForm.pid);
          //关闭弹框
          _this.dialogVisible = false;
          _this.$message({
            message: res.msg,
            type: "success"
          });
        } else {
          _this.$message({
            message: res.msg,
            type: "error"
          });
        }
      }
    });
  },
```

5、删除部门

```
 //删除机构
    async handleTableDelete(index, row) {
      let _this = this;
      _this.$confirm("确定删除吗 ?", "系统提示", {
        confirmButtonText: "确定",
        cancelButtonText: "取消",
        type: "warning"
      }).then(async () => {
        let parm = {
          id: row.id
        };
        let { data: res } = await _this.$http.post(
          "/api/sysDept/deleteDept",
          parm
        );
        if (res.code == 200) {
          //刷新表格数据
          //刷新左侧树数据
          _this.getLeftDeptTree();
          //刷新新增部门上级部门树
          _this.getParentTree();
          //刷新表格数据
          _this.getDeptListByLikeId(_this.deptForm.pid);
          _this.$message({
            message: res.msg,
            type: "success"
          });
        } else {
          _this.$message({
            message: res.msg,
            type: "error"
          });
        }
      });
    },
```

第42讲 用户管理后台接口开发讲

1、新增用户接口

新增用户需要判断用户名是否存在，存在不能重复添加

```
  /**
     * 新增用户
     * @return
     */
    @RequestMapping(value = "addUser",method = RequestMethod.POST)
    public ResultVo addUser(@RequestBody SysUser user){
        QueryWrapper<SysUser> query = new QueryWrapper<>();
        query.lambda().eq(SysUser::getUsername,user.getUsername());
        //查询用户是否存在
        SysUser one = userService.getOne(query);
        if(one != null){
            return ResultUtils.error("用户名已经存在!");
        }
        //加密用户密码
        String pwd = passwordEncoder.encode(user.getPassword());
        user.setPassword(pwd);
        boolean b = userService.save(user);
        if(b){
            return ResultUtils.success("新增用户成功");
        }else{
            return ResultUtils.error("新增用户失败");
        }
    }
```

2、编辑用户

编辑用户先查询编辑的用户信息回显，再编辑

```
  /**
     * 编辑用户保存
     * @return
     */
    @RequestMapping(value = "updateSaveUser",method = RequestMethod.POST)
    public ResultVo updateSaveUser(@RequestBody SysUser user){
        //判断用户是否存在
        QueryWrapper<SysUser> query = new QueryWrapper<>();
        query.lambda().eq(SysUser::getUsername,user.getUsername());
        SysUser one = userService.getOne(query);
        Long id = one.getId();//查询出来的id
        Long editId = user.getId();//编辑的用户id
        if(!id.equals(editId)){
            return ResultUtils.error("用户名已经存在!");
        }
        boolean b = userService.updateById(user);
        if(b){
            return ResultUtils.success("编辑成功");
        }else{
            return ResultUtils.error("编辑失败");
        }
    }
```

3、删除用户

```

    /**
     * 根据用户id删除
     * @return
     */
    @RequestMapping(value = "deleteUserById",method = RequestMethod.POST)
    public ResultVo deleteUserById(@RequestBody SysUser user){
        boolean b = userService.removeById(user.getId());
        if(b){
            return ResultUtils.success("删除用户成功");
        }else{
            return ResultUtils.error("删除用户失败");
        }
    }
```

4、根据id查询用户

```
  /**
     * 根据用户id查询用户端
     * @return
     */
    @RequestMapping(value = "getUserById",method = RequestMethod.POST)
    public ResultVo getUserById(@RequestBody SysUser user){
        SysUser sysUser = userService.getById(user.getId());
        return ResultUtils.success("查询成功",sysUser);
    }
```

5、查询用户列表

```
 //查询用户列表
    @RequestMapping(value = "/getUserList",method = RequestMethod.POST)
    public ResultVo getUserList(@RequestBody UserParm parm){
        QueryWrapper<SysUser> query  =new QueryWrapper<>();
        if(StringUtils.isNotBlank(parm.getLoginName())){
            query.lambda().eq(SysUser::getLoginName,parm.getLoginName());
        }
        if(StringUtils.isNotBlank(parm.getMobile())){
            query.lambda().eq(SysUser::getMobile,parm.getMobile());
        }
        query.lambda().eq(SysUser::getDeptId,parm.getDeptId());
        IPage<SysUser> page = new Page<>();
        page.setCurrent(parm.getCurrentPage());
        page.setSize(parm.getPageSize());
        IPage<SysUser> userIPage = userService.page(page, query);
        return ResultUtils.success("查询成功",userIPage);
    }
```

第43讲 用户管理前端接口对接

1、左侧部门树对接

树加载完，需要点击第一个节点，查询该节点下的用户列表

1.1、加载左侧部门树

```
//获取左侧组织树
  async getLeftTree() {
    let _this = this;
    let { data: res } = await _this.$http.post("/api/department/getLeftTree");
    if (res.code == 200) {
      _this.nodes = res.data;
    }
  },
```

1.2、加载完设置第一个节点选中

```
handleCreated: function(ztreeObj) {
     this.ztreeObj = ztreeObj;
     let firstTree = ztreeObj.getNodes()[0];
     //默认选中第一个
     ztreeObj.selectNode(firstTree);
     //设置节点全部展开
     ztreeObj.expandAll(true);
     //加载完自动点击第一个，加载右边表格
     if (firstTree) {
       this.setting.callback.onClick(null, firstTree.id, firstTree);
     }
   },
```

1.3、点击节点事件

```
ztreeOnClick: function(evt, treeId, treeNode) {
     this.leftDeptId = treeNode.id;
     console.log("调用点击事件");
     console.log(treeNode);
     //根据部门id查询部门下的用户

     this.getUserByDeptId(treeNode.id);
   },
```

2、获取右侧用户列表

2.1、根据树的选中树的id查询用户列表

```
async getUserByDeptId(deptId) {
     let parm = {
       deptId: deptId,
       pageSize: this.pageSize,
       currentPage: this.currentPage
     };
     let { data: res } = await this.$http.get("/api/user/getUserList", parm);
     if (res.code == 200) {
       this.tableData = res.data.records;
       this.currentPage = res.data.current;
       this.total = res.data.total;
     }
   },
```

3、新增用户

3.1、获取新增用户部门树

```
//获取新增弹框组织树
   async getSelectDeptTree() {
     let _this = this;
     let { data: res } = await _this.$http.post("/api/department/getDeptTree");
     if (res.code == 200) {
       _this.selectNodes = res.data;
     }
   },
```

3.2、新增按钮点击

```
//打开新增页面
   addUI() {
     this.editTag = "0";
     this.dialogTitle = '新增用户';
     this.dialogFormVisible = true;
     //清空表单数据
     this.resetForm("userDialog");
   },
```

3.3、上级部门点击事件

```
//新增用户选择部门点击树事件
    selectZtreeOnClick(evt, treeId, treeNode) {
      this.userInfo.deptName = treeNode.name;
      this.userInfo.deptId = treeNode.id;
      this.clickDeptId = treeNode.id;
    },
```

3.4、新增用户确认事件

```
//保存新增用户信息
   async addUser() {
     let _this = this;
     let parm = _this.userInfo;
     let url = "";
     if (_this.editTag == "0") {
       url = "/api/user/addUser";
     }
     let { data: res } = await _this.$http.post(url, parm);
     if (res.code == 200) {
       //关闭窗口
       _this.dialogFormVisible = false;
        //取消全部选中
       _this.ztreeObj.checkAllNodes(false);
       _this.ztreeObj.cancelSelectedNode();
       //设置添加时选中的节点
       var node = this.ztreeObj.getNodeByParam("id", _this.clickDeptId);
       if (node) {
         _this.ztreeObj.selectNode(node, true);
         _this.setting.callback.onClick(null, node.id, node);
       }
       _this.$message({
         message: res.msg,
         type: "success"
       });
     } else {
       _this.$message({
         message: res.msg,
         type: "error"
       });
     }
   },
```

4、编辑用户

获取要编辑的用户信息，用于回显

4.0、编辑点击事件

```
 //编辑用户弹框
    editUserUI(index, row) {
      let _this = this;
      _this.editTag = "1";
      //显示弹框
      _this.dialogFormVisible = true;
      //清空表单数据
      _this.resetForm("userDialog");
      //查询要编辑的用户信息
      _this.getUserById(row.id);
    },
```

4.1、获取要编辑的用户信息

```
//根据id查询用户信息
    async getUserById(userId) {
      let _this = this;
      let parm = {
        id: userId
      };
      let { data: res } = await _this.$http.post("/api/user/getUserById", parm);
      if (res.code == 200) {
        _this.userInfo = res.data;
        _this.clickDeptId = res.data.deptId;
      }
    },
```

4.2、选择上级部门树时，选中原理选中的节点

```
  //上级部门树创建成功调用
    createdParent(obj) {
     this.parentZtreeObj = obj;
      obj.expandAll(true);
      //根据原来选中的id来找到要选中的节点
      var node = this.parentZtreeObj.getNodeByParam("id", this.clickDeptId);
      //把找到的节点设为选中状态
      this.parentZtreeObj.selectNode(node);
    },
```

4.3、编辑保存用户

```
 //保存新增用户信息
    async addUser() {
      let _this = this;
```

```
     let parm = _this.userInfo;
     let url = "";
     if (_this.editTag == "0") {
       url = "/api/user/addUser";
     } else {
       url = "/api/user/updateUser";
     }
     let { data: res } = await _this.$http.post(url, parm);
     if (res.code == 200) {
       //关闭窗口
       _this.dialogFormVisible = false;
       //取消全部选中
       _this.ztreeObj.checkAllNodes(false);
       _this.ztreeObj.cancelSelectedNode();
       //设置添加时选中的节点
       var node = this.ztreeObj.getNodeByParam("id", _this.clickDeptId);
       if (node) {
         _this.ztreeObj.selectNode(node, true);
         _this.setting.callback.onClick(null, node.id, node);
       }
       _this.$message({
         message: res.msg,
         type: "success"
       });
     } else {
       _this.$message({
         message: res.msg,
         type: "error"
       });
     }
   },
```

5、删除用户

```
//删除用户
   async deleteUser(index, row) {
     let parm = {
       id: row.id
     };
     let { data: res } = await this.$http.post("/api/user/deleteUser", parm);
     if (res.code == 200) {
       this.$message({
         message: res.msg,
         type: "success"
       });
        this.getUserByDeptId(this.leftDeptId);
     } else {
       this.$message({
         message: res.msg,
         type: "error"
       });
     }
   },
```

第44讲 用户分配角色接口讲解

1、查询所有的角色列表

查询所有的角色，用户选择用户的角色；在RoleController中添加如下方法

```
 /**
     * 分配角色时查询角色列表
     * @return
     */
    @RequestMapping(value = "getRoleListForUser",method = RequestMethod.POST)
    public ResultVo getRoleListForUser(){
        List<SysRole> list = roleService.list();
        return ResultUtils.success("成功",list);
    }
```

2、根据用户id查询用户的角色

用于在分配角色时，如果用户已经分配过角色，需要表格显示出当前用户已经分配的角色

2.1、在后台system目录下新建user_role目录，再新建entity和mapper目录，

2.2、在entity目录下新建UserRole实体用于映射数据库sys_user_role表

```
package com.itmk.system.user_role.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;

@Data
@TableName(value = "sys_user_role")
public class UserRole {
    @TableId(type = IdType.AUTO)
    private Long id;
    private Long userId;
    private Long roleId;
}
```

2.3、在mapper目录下新建SysUserRoleMapper接口

```
package com.itmk.system.user_role.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.itmk.system.user_role.entity.UserRole;
import org.apache.ibatis.annotations.Param;

public interface SysUserRoleMapper extends BaseMapper<UserRole> {
    UserRole getRoleIdByUserId(@Param("userId") Long userId);
}
```

2.4、在resources目录下新建SysUserRoleMapper.xml

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.user_role.mapper.SysUserRoleMapper">

    <select id="getRoleIdByUserId" parameterType="long"
resultType="com.itmk.system.user_role.entity.UserRole">
        select * from sys_user_role
        where user_id = #{userId}
    </select>
</mapper>
```

2.5、在RoleService接口新增方法

```
  /**
     * 根据用户id查询角色id
     *
     * @return
     */
    UserRole getRouleIdByUser(UserRole userRole);

    /**
     * 分配权限
     *
     * @param userRole
     * @return
     */
    void assignRole(UserRole userRole);
```

2.6、在RoleController新增方法

```
 /**
     * 分配角色时查询角色列表
     * @return
     */
    @RequestMapping(value = "getRoleListForUser",method = RequestMethod.POST)
    public ResultVo getRoleListForUser(){
        List<SysRole> list = roleService.list();
        return ResultUtils.success("成功",list);
    }
      /**
     * 根据用户id查询角色id
     * @param userRole
     * @return
     */
    @RequestMapping(value = "/getRouleIdByUser",method = RequestMethod.POST)
    public ResultVo getRouleIdByUser(@RequestBody UserRole userRole){
        UserRole id = roleService.getRouleIdByUser(userRole);
        return ResultUtils.success("成功",id);
    }

    /**
     * 分配用户角色
     * @param userRole
     * @return
     */
    @RequestMapping(value = "/assignRole",method = RequestMethod.POST)
    public ResultVo assignRole(@RequestBody UserRole userRole){
        roleService.assignRole(userRole);
        return ResultUtils.success("分配成功!");
    }
```

第45讲 用户分配角色前端接口对接讲解

1、分配角色按钮点击事件

setUserId:'',//当前分配的用户id

setRoleShow:false,//弹框的显示和影藏

currentRow: "", //分配角色表格当前选中行

table事件:

@current-change="selectRoleRow"点击行触发事件

```
//取消
   setCurrent(row) {
     this.setRoleShow = false;
     this.$refs.roleTable.setCurrentRow(row);
   },
   //分配角色表格选中行
   selectRoleRow(row) {
     this.currentRow = row;
   },
```

```
//分配角色弹框显示
   async assignRole(row) {
     let _this = this;
     _this.setUserId = row.id;
     //加await会等到请求返回才执行下面的语句
     //根据用户id查询角色id,用于回显
     let role = await _this.getRouleIdByUser(row.id);
     _this.setRoleShow = true;
     _this.$nextTick(function() {
       //查询出当前用户角色id，和角色列表比较，相等的设为选中
       for (let i = 0; i < _this.setRoleData.length; i++) {
         if (_this.roleId == _this.setRoleData[i].id) {
           //设为选中
           _this.$refs.roleTable.setCurrentRow(_this.setRoleData[i]);
           //保存当前选中的角色数据
           this.currentRow = _this.setRoleData[i];
         }
       }
     });
   },
```

```
//查询当前用户的角色
   async getRouleIdByUser(userId) {
     let parm = {
       userId: userId
     };
     let { data: res } = await this.$http.post(
       "/api/role/getRouleIdByUser",
       parm
     );
     if (res.code == 200 && res.data) {
       this.roleId = res.data.roleId;
     } else {
       this.roleId = "";
     }
   },
```

2、分配角色确定按钮

```
  //分配角色确认按钮
    async confirmSetRole() {
      let _this = this;
      if(!_this.currentRow.id){
        _this.$message({
          message:'请选择角色',
          type:'warning'
        })
        return;
      }
      let parm = {
        userId: _this.setUserId,
        roleId: _this.currentRow.id
      };
      let { data: res } = await _this.$http.post("/api/role/assignRole", parm);
      if (res.code == 200) {
        _this.$refs.roleTable.setCurrentRow();
        _this.setRoleShow = false;
        _this.$message({
          message: res.msg,
          type: "success"
        });
      } else {
        _this.$message({
          message: res.msg,
          type: "error"
        });
      }
    },
```

3、选中高亮样式

在分配角色el-dialog 添加class="roleClass"

```
.roleClass /deep/ .el-table__body tr.current-row > td {
  background: #409eff !important;
  color: #fff;
}
```

```
.el-dialog__wrapper /deep/ .el-dialog__body{
  padding-top:5px!important;
}
```

第46讲 角色分配权限后台接口讲解

1、新建role-permission层

在system中新建role_permission目录，并在该目录下新建mapper、entity、service目录

2、新建RolePermissionMapper接口

2.1、在mapper目录新建RolePermissionMapper接口

```
package com.itmk.system.RolePermission.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.itmk.system.RolePermission.entity.RolePermission;
import org.apache.ibatis.annotations.Param;

import java.util.List;

public interface RolePermissionMapper extends BaseMapper<RolePermission> {
    //批量新增权限
    boolean saveRolePermissions(@Param("roleId") Long roleId, @Param("perIds")
List<Long> perIds);
}
```

2.2、在resources目录新建RolePermissionMapper.xml文件

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.itmk.system.RolePermission.mapper.RolePermissionMapper">

    <insert id="saveRolePermissions" >
      insert  into sys_role_permission(role_id,permission_id) values
        <foreach collection="perIds" item="item" index="index" separator=",">
            (#{roleId},#{item})
        </foreach>
    </insert>
</mapper>
```

2.3、新建RolePermission实体

```
package com.itmk.system.RolePermission.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;

import java.io.Serializable;

@Data
@TableName(value = "sys_role_permission")
public class RolePermission implements Serializable {
    @TableId(type= IdType.AUTO)
    private Long id;
    private Long roleId;
    private Long permissionId;
}
```

2.4、新建RolePermissionService

```
package com.itmk.system.RolePermission.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.itmk.system.RolePermission.entity.RolePermission;
import com.itmk.system.permission.Vo.TreeVo;

import java.util.List;

public interface RolePermissionService extends IService<RolePermission> {
    /**
     * 分配权限保存
     * @param
     */
    void saveAssignRole(Long roleId,List<Long> collect);
}
```

```
package com.itmk.system.RolePermission.service.Impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.itmk.system.RolePermission.entity.RolePermission;
import com.itmk.system.RolePermission.mapper.RolePermissionMapper;
import com.itmk.system.RolePermission.service.RolePermissionService;
import com.itmk.system.permission.Vo.TreeVo;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class RolePermissionServiceImpl extends ServiceImpl<RolePermissionMapper,
RolePermission> implements RolePermissionService {

    @Override
    @Transactional
    public void saveAssignRole(Long roleId,List<Long> ids) {
        //1.删除原来角色的权限
        QueryWrapper<RolePermission> query = new QueryWrapper<>();
        query.lambda().eq(RolePermission::getRoleId,roleId);
        this.baseMapper.delete(query);
        //2.插入新权限
        this.baseMapper.saveRolePermissions(roleId,ids);
    }
}
```

2.5、新建PermissionRoleParmVo

```
package com.itmk.system.RolePermission.vo;

import com.itmk.system.permission.Vo.TreeVo;
import lombok.Data;

import java.util.List;

@Data
public class PermissionRoleParmVo {
    private List<TreeVo> list;
    private Long roleId;
}
```

3、控制器层方法

3.1、查询权限树

```
  /**
     * 分配权限树查询
     * @return
     */
    @RequestMapping(value = "/permissonTree",method = RequestMethod.POST)
    public ResultVo permissonTree(@RequestBody PerVo perVo){
        SysUser sysUser = userService.getById(perVo.getUserId());

        //1.查询当前用户的所有权限
        Long userId = perVo.getUserId();
        List<Permission> permissions = null;
        if(StringUtils.isNotEmpty(sysUser.getIsAdmin()) &&
sysUser.getIsAdmin().equals("1")){
            permissions = permissionService.list();
        }else {
            permissions = permissionService.selectPermissionByUserId(userId);
        }
        //2.根据要分配角色id查询角色的权限
        List<Permission> byRoleId = permissionService.findByRoleId(perVo.getRoleId());
        //3.把2中的数据设为选中
        List<TreeVo> listTree = new ArrayList<>();
        for(int i = 0;i<permissions.size();i++){
            if(permissions.get(i) != null){
                TreeVo tree = new TreeVo();
                tree.setId(permissions.get(i).getId());
                tree.setName(permissions.get(i).getLabel());
                tree.setPid(permissions.get(i).getParentId());
                if(byRoleId.size() > 0){
                    for(int j = 0; j < byRoleId.size();j++){
                        if(permissions.get(i).getId().equals(byRoleId.get(j).getId()))
{
                            tree.setChecked(true);
                            break;
                        }
                    }
                }
                listTree.add(tree);
            }
        }
        return ResultUtils.success("成功",listTree);
    }
```

3.2、分配权限保存

```
  //保存权限
    @RequestMapping(value = "/saveAssignRole",method = RequestMethod.POST)
    public ResultVo saveAssignRole(@RequestBody PermissionRoleParmVo parmVo){
        if(parmVo != null && !parmVo.getList().isEmpty()){
            List<TreeVo> list = parmVo.getList();
            Long roleId = parmVo.getRoleId();
            List<Long> ids = list.stream().filter(item -> item != null).map(item ->
item.getId()).collect(Collectors.toList());
            rolePermissionService.saveAssignRole(roleId,ids);
            return ResultUtils.success("分配成功!");
        }else{
            return ResultUtils.error("请选择权限!");
        }
    }
```

第47讲 角色分配权限前端接口对接

1、分配权限点击事件

```
 //查询权限树
    async assignRole(row) {
      this.roldId = row.id;
      this.dialogTitle = '为【'+row.name+'】分配权限';
      let parm = {
        userId: sessionStorage.getItem("userId"),
        roleId: row.id
      };
      let { data: res } = await this.$http.post(
        "/api/permission/permissonTree",
        parm
      );
      if (res.code == 200) {
        this.treeDatas = res.data;
      }
      this.innerVisible = true;
    },
```

2、树点击选择事件

```
  ztreeOnCheck() {
      let checked = this.ztreeObj.getCheckedNodes(true);
      this.checkPermissions = checked;
      console.log(checked);
    },
```

3、保存分配的权限

```
async saveAssign() {
      if (this.checkPermissions.length < 1) {
        this.$message({
          message: "请勾选权限!",
          type: "success"
        });
        return;
      }
      let parms = {
        list:this.checkPermissions,
        roleId:this.roldId
      }
      let { data: res } = await
this.$http.post("/api/permission/saveAssignRole",parms);
      if(res.code ==200){
         this.innerVisible = false;
        this.$message({
          message:res.msg,
          type:'success'
        })
      }else{
        this.$message({
          message:res.msg,
          type:'error'
        })
      }
    },
```

第48讲 全局异常处理器

全局异常处理器：拦截运行时异常，给前端一个友好的提示

```
@Slf4j
@ControllerAdvice
public class GlobalExceptionHandler {
    /**
     * 未知的运行时异常拦截
     */
    @ExceptionHandler(RuntimeException.class)
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    @ResponseBody
    public ResultVo notFount(RuntimeException e) {
        log.error("运行时异常:", e);
        return ResultUtils.error("服务器错误");
    }
}
```

第49讲 退出登录讲解

1、前端页面

在home.vue添加点击事件

```
 <el-dropdown-item @click.native="logout">退出</el-dropdown-item>
 async logout() {
      let { data: res } = await this.$http.post("/api/user/loginOut");
      console.log(res);
      if(res.code == 200){
        sessionStorage.clear();
         window.location.href = "/login";
      }
    },
```

2、后台接口

2.1、编写自定义登录退出处理器

```
package com.itmk.security.handler;

import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.itmk.result.ResultUtils;
import com.itmk.status.StatusCode;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.logout.LogoutSuccessHandler;
import
org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * 退出登录处理器
 */
@Component
public class CustomerLogoutSuccessHandler implements LogoutSuccessHandler {
    @Override
    public void onLogoutSuccess(HttpServletRequest request, HttpServletResponse
response, Authentication authentication) throws IOException, ServletException {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if(auth != null){
            new SecurityContextLogoutHandler().logout(request, response, auth);
        }
        response.setContentType("application/json;charset=UTF-8");
        ServletOutputStream out = response.getOutputStream();
        String res = JSONObject.toJSONString(ResultUtils.success("退出登录成功！"),
SerializerFeature.DisableCircularReferenceDetect);
        out.write(res.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
}
```

2.2、配置自定义退出处理器

```
 @Autowired
    private CustomerLogoutSuccessHandler customerLogoutSuccessHandler;
```

```
.and()

.logout().logoutUrl("/api/user/loginOut").logoutSuccessHandler(customerLogoutSuccessHandler);
```

3、顶部时间显示

```
 showTime() {
      var week = new Array(
        "星期日",
        "星期一",
        "星期二",
        "星期三",
        "星期四",
        "星期五",
        "星期六"
      );
      var date = new Date();
      var year = date.getFullYear();
      var month = date.getMonth() + 1;
      var day = date.getDate();
      var hour = date.getHours();
      var minutes = date.getMinutes();
      var second = date.getSeconds();
      this.date =
        year +
        "." +
        (month < 10 ? "0" + month : month) +
        "." +
        day +
        "" +
        " " +
        hour +
        ":" +
        minutes +
        ":" +
        (second < 10 ? "0" + second : second) +
        " " +
        (week[date.getDay()] || "");
    }
```

```

  data(){
    return {
       date: "",
    }
  },
mounted() {
    $vueIndex = this;
    this.showTime();
    setInterval(function() {
      $vueIndex.showTime();
    }, 1000);
  },
```

第50讲 按钮权限判断

1.1、在src目录下新建permissions目录，并新建index.js

```
/**
 * 判断是否有权限
 * @param  perms
 */
export default function hasPermission (perms) {
    let hasPermission = false
    let permissions = JSON.parse(sessionStorage.getItem("authList"));
    for(let i=0, len=permissions.length; i<len; i++) {
        if(permissions[i] === perms) {
            hasPermission = true;
            break
        }
    }
    return hasPermission
}
```

1.2、在main.js中引入

```
import permissions from './permissions/index'
```

1.3、挂载到vue.js上

```
Vue.prototype.hasPerm = permissions;
```

1.4、使用

```
v-if='hasPerm("sys:addDepartment")'
```

第51讲 Redis缓存的使用讲解

1、引入redis

```
 <!--采用redis来管理-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
```

2、配置redis

2.1、在itmk-base-common模块新建config目录，并建redis目录

新建redis配置类RedisConfig

```
package com.itmk.config.redis;
import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.cache.RedisCacheConfiguration;
import org.springframework.data.redis.cache.RedisCacheManager;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.*;

import java.time.Duration;

@Configuration
public class RedisConfig {
    @Value("${spring.redis.expire}")
    private Long expire;

    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory)
{
        RedisTemplate<String, Object> template = new RedisTemplate<String, Object>();
        template.setConnectionFactory(factory);
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer(Object.class);
        //解决查询缓存转换异常的问题
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        om.activateDefaultTyping(om.getPolymorphicTypeValidator(),
ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);
        StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
        // key采用String的序列化方式
        template.setKeySerializer(stringRedisSerializer);
        // hash的key也采用String的序列化方式
        template.setHashKeySerializer(stringRedisSerializer);
        // value序列化方式采用jackson
        template.setValueSerializer(jackson2JsonRedisSerializer);
        // hash的value序列化方式采用jackson
        template.setHashValueSerializer(jackson2JsonRedisSerializer);
        template.afterPropertiesSet();
        return template;
    }

    //@Cacheable注解字符集编码配置
    @Bean
    public RedisCacheManager cacheManager(RedisConnectionFactory factory) {
        RedisCacheConfiguration config = RedisCacheConfiguration.defaultCacheConfig();
        config.entryTtl(Duration.ofMinutes(expire));//缓存过期时间
        RedisCacheConfiguration cacheConfiguration = config

.serializeKeysWith(RedisSerializationContext.SerializationPair.fromSerializer(RedisSerializer.string()))
.serializeValuesWith(RedisSerializationContext.SerializationPair.fromSerializer(RedisS
erializer.json()));

        return RedisCacheManager
                .builder(factory)
                .cacheDefaults(cacheConfiguration)
                .build();
    }
}
```

修改配置文件

```
spring:
  redis:
    expire: 60000
    database: 0 # Redis使用的库
    host: localhost
    port: 6379 #端口号
    password: huazuoban123456  #redis密码
#    lettuce:
#      pool:
#        max-active: 8     # 连接池最大连接数（使用负值表示没有限制）
#        max-wait: 10000   # 连接池最大阻塞等待时间（使用负值表示没有限制）
#        max-idle: 8       # 连接池中的最大空闲连接
#        min-idle: 1       # 连接池中的最小空闲连接
    timeout: 10000        #  连接超时时间（毫秒）
  cache:
    type: redis           #使用redis做缓存
#  mybatis-plus
mybatis-plus:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
```

新建redis常用工具类RedisService

```
package com.itmk.config.redis;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.TimeUnit;

@Component
public class RedisService {

    @Autowired
    private RedisTemplate<String, Object> redisTemplate;
    /**
     * 实现命令：TTL key，以秒为单位，返回给定 key的剩余生存时间(TTL, time to live)。
```

```
 *
 * @param key
 * @return
 */
public long ttl(String key) {
    return redisTemplate.getExpire(key);
}

/**
 * 实现命令：expire 设置过期时间，单位秒
 *
 * @param key
 * @return
 */
public void expire(String key, long timeout) {
    redisTemplate.expire(key, timeout, TimeUnit.SECONDS);
}

/**
 * 实现命令：INCR key，增加key一次
 *
 * @param key
 * @return
 */
public long incr(String key, long delta) {
    return redisTemplate.opsForValue().increment(key, delta);
}

/**
 * 实现命令：KEYS pattern，查找所有符合给定模式 pattern的 key
 */
public Set<String> keys(String pattern) {
    return redisTemplate.keys(pattern);
}

/**
 * 实现命令：DEL key，删除一个key
 *
 * @param key
 */
public void del(String key) {
    redisTemplate.delete(key);
}

// String（字符串）

/**
 * 实现命令：SET key value，设置一个key-value（将字符串值 value关联到 key）
 *
 * @param key
 * @param value
 */
public void set(String key, String value) {
    redisTemplate.opsForValue().set(key, value);
}

/**
 * 实现命令：SET key value EX seconds，设置key-value和超时时间（秒）
 *
 * @param key
 * @param value
```

```
     * @param timeout （以秒为单位）
     */
    public void set(String key, String value, long timeout) {
        redisTemplate.opsForValue().set(key, value, timeout, TimeUnit.SECONDS);
    }

    /**
     * 实现命令：GET key，返回 key所关联的字符串值。
     *
     * @param key
     * @return value
     */
    public String get(String key) {
        return (String) redisTemplate.opsForValue().get(key);
    }

    // Hash（哈希表）

    /**
     * 实现命令：HSET key field value，将哈希表 key中的域 field的值设为 value
     *
     * @param key
     * @param field
     * @param value
     */
    public void hset(String key, String field, Object value) {
        redisTemplate.opsForHash().put(key, field, value);
    }

    /**
     * 实现命令：HGET key field，返回哈希表 key中给定域 field的值
     *
     * @param key
     * @param field
     * @return
     */
    public String hget(String key, String field) {
        return (String) redisTemplate.opsForHash().get(key, field);
    }

    /**
     * 实现命令：HDEL key field [field ...]，删除哈希表 key 中的一个或多个指定域，不存在的域将被忽略。
     *
     * @param key
     * @param fields
     */
    public void hdel(String key, Object... fields) {
        redisTemplate.opsForHash().delete(key, fields);
    }

    /**
     * 实现命令：HGETALL key，返回哈希表 key中，所有的域和值。
     *
     * @param key
     * @return
     */
    public Map<Object, Object> hgetall(String key) {
        return redisTemplate.opsForHash().entries(key);
    }
    // List（列表）

    /**
     * 实现命令：LPUSH key value，将一个值 value插入到列表 key的表头
     *
     * @param key
     * @param value
     * @return 执行 LPUSH命令后，列表的长度。
     */
    public long lpush(String key, String value) {
        return redisTemplate.opsForList().leftPush(key, value);
    }

    /**
     * 实现命令：LPOP key，移除并返回列表 key的头元素。
     *
     * @param key
     * @return 列表key的头元素。
     */
    public String lpop(String key) {
        return (String) redisTemplate.opsForList().leftPop(key);
    }

    /**
     * 实现命令：RPUSH key value，将一个值 value插入到列表 key的表尾(最右边)。
     *
     * @param key
     * @param value
     * @return 执行 LPUSH命令后，列表的长度。
     */
    public long rpush(String key, String value) {
        return redisTemplate.opsForList().rightPush(key, value);
    }

    public Long setList(String key, List<Object> list){
        return redisTemplate.opsForList().rightPushAll(key,list);
    }
    /**
     * 查询key是否存在
     * @param key
     * @return
     */
    @SuppressWarnings("unchecked")
    public boolean exists(String key) {
        return redisTemplate.hasKey(key);
    }
}
```

启用redis，在项目启动类上添加注解启用redis

```
@EnableCaching
```

3、缓存使用原理

查询数据的时候，先查看缓存，如果缓存存在数据，直接返回缓存的数据；如果不存在，则查询数据库，把查到的数据放到缓存，并返回数据。

4、redis常用缓存注解

4.1、查询缓存

```
@Cacheable(value = "sys_role",key = "#roleId")
SysRole findById(int roleId);
```

4.2、新增或修改缓存

```
新增缓存

    @CachePut(value = "sys_role",key = "#role.id")
    public SysRole addRole(SysRole role) {
        this.baseMapper.insert(role);
        return role;
    }
```

```
编辑缓存

    @CachePut(value = "sys_role",key = "#role.id")
    public SysRole updateRole(SysRole role) {
        this.baseMapper.updateById(role);
        return this.baseMapper.selectById(role.getId());
    }
```

注意事项：新增和查询缓存的返回值必须要和查询缓存的返回值一样

4.3、删除缓存

```
删除单个
@CacheEvict(value = "sys_role",key = "#role.id")

删除value下的全部
@CacheEvict(value = "sys_role",allEntries = true)
```

5、测试缓存

spring boot 2.2.x junit使用 https://docs.spring.io/spring-boot/docs/2.2.x/reference/html/spring-boot-features.html#boot-features-testing

```
<!-- springboot 单元测试 -->
       <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-test</artifactId>
           <scope>test</scope>
           <exclusions>
               <exclusion>
                   <groupId>org.junit.vintage</groupId>
                   <artifactId>junit-vintage-engine</artifactId>
               </exclusion>
           </exclusions>
       </dependency>
```

```
@Slf4j
@SpringBootTest(classes = AdminApplication.class)
public class TestCache {
    @Autowired
    private RoleCacheService roleCacheService;
    @Test
    public void getRoleById(){
        Long roleId = 9L;
        SysRole role = roleCacheService.getRoleById(roleId);
        log.info(role.toString());
    }
}
```

第52讲 自定义缓存

1.1、新建 com.itmk.config.redis.RedisService

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.TimeUnit;

@Component
public class RedisService {

    @Autowired
    private RedisTemplate<String, Object> redisTemplate;
    /**
     * 实现命令：TTL key，以秒为单位，返回给定 key的剩余生存时间(TTL, time to live)。
     *
     * @param key
     * @return
     */
    public long ttl(String key) {
        return redisTemplate.getExpire(key);
    }

    /**
     * 实现命令：expire 设置过期时间，单位秒
     *
     * @param key
     * @return
     */
    public void expire(String key, long timeout) {
        redisTemplate.expire(key, timeout, TimeUnit.SECONDS);
    }

    /**
     * 实现命令：INCR key，增加key一次
     *
```

```
 * @param key
 * @return
 */
public long incr(String key, long delta) {
    return redisTemplate.opsForValue().increment(key, delta);
}

/**
 * 实现命令：KEYS pattern，查找所有符合给定模式 pattern的 key
 */
public Set<String> keys(String pattern) {
    return redisTemplate.keys(pattern);
}

/**
 * 实现命令：DEL key，删除一个key
 *
 * @param key
 */
public void del(String key) {
    redisTemplate.delete(key);
}

// String（字符串）

/**
 * 实现命令：SET key value，设置一个key-value（将字符串值 value关联到 key）
 *
 * @param key
 * @param value
 */
public void set(String key, String value) {
    redisTemplate.opsForValue().set(key, value);
}

/**
 * 实现命令：SET key value EX seconds，设置key-value和超时时间（秒）
 *
 * @param key
 * @param value
 * @param timeout （以秒为单位）
 */
public void set(String key, String value, long timeout) {
    redisTemplate.opsForValue().set(key, value, timeout, TimeUnit.SECONDS);
}

/**
 * 实现命令：GET key，返回 key所关联的字符串值。
 *
 * @param key
 * @return value
 */
public String get(String key) {
    return (String) redisTemplate.opsForValue().get(key);
}

// Hash（哈希表）

/**
 * 实现命令：HSET key field value，将哈希表 key中的域 field的值设为 value
 *
```

```
     * @param key
     * @param field
     * @param value
     */
    public void hset(String key, String field, Object value) {
        redisTemplate.opsForHash().put(key, field, value);
    }

    /**
     * 实现命令：HGET key field，返回哈希表 key中给定域 field的值
     *
     * @param key
     * @param field
     * @return
     */
    public String hget(String key, String field) {
        return (String) redisTemplate.opsForHash().get(key, field);
    }

    /**
     * 实现命令：HDEL key field [field ...]，删除哈希表 key 中的一个或多个指定域，不存在的域将被忽略。
     *
     * @param key
     * @param fields
     */
    public void hdel(String key, Object... fields) {
        redisTemplate.opsForHash().delete(key, fields);
    }

    /**
     * 实现命令：HGETALL key，返回哈希表 key中，所有的域和值。
     *
     * @param key
     * @return
     */
    public Map<Object, Object> hgetall(String key) {
        return redisTemplate.opsForHash().entries(key);
    }

    // List（列表）

    /**
     * 实现命令：LPUSH key value，将一个值 value插入到列表 key的表头
     *
     * @param key
     * @param value
     * @return 执行 LPUSH命令后，列表的长度。
     */
    public long lpush(String key, String value) {
        return redisTemplate.opsForList().leftPush(key, value);
    }

    /**
     * 实现命令：LPOP key，移除并返回列表 key的头元素。
     *
     * @param key
     * @return 列表key的头元素。
     */
    public String lpop(String key) {
        return (String) redisTemplate.opsForList().leftPop(key);
    }

    /**
     * 实现命令：RPUSH key value，将一个值 value插入到列表 key的表尾(最右边)。
     *
     * @param key
     * @param value
     * @return 执行 LPUSH命令后，列表的长度。
     */
    public long rpush(String key, String value) {
        return redisTemplate.opsForList().rightPush(key, value);
    }

    public Long setList(String key, List<Object> list){
        return redisTemplate.opsForList().rightPushAll(key,list);
    }
    /**
     * 查询key是否存在
     * @param key
     * @return
     */
    @SuppressWarnings("unchecked")
    public boolean exists(String key) {
        return redisTemplate.hasKey(key);
    }
}
```

1.2、新建查询单个实体的函数式接口

```
package com.itmk.config.redis;
@FunctionalInterface
public interface FunctionEntityCache<T> {
    T getCache();
}
```

1.3、新建查询List的函数式接口

```
package com.itmk.config.redis;

import java.util.List;

@FunctionalInterface
public interface FunctionListCache <T>{
    List<T> getCache();
}
```

1.4、从缓存中获取实体对象和List对象

```
package com.itmk.config.redis;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.util.List;
```

```
@Component
public class CacheService {
    @Autowired
    private RedisService redisService;

    //从缓存中获取实体对象
    public <T> T getEntityCache(String key, Long timeout, Class<T> cla,
FunctionEntityCache<T> function) {
        //返回的实体
        T obj = null;
        //1.根据key取出缓存中的key
        //2.判断缓存中的数据是否存在，存在则返回，不存在则查询数据库
        String value = redisService.get(key);
        if (StringUtils.isEmpty(value)) { //缓存中数据不存在，查询数据库
            obj = function.getCache();
            if (obj != null) {
                String seach = JSONObject.toJSONString(obj);
                redisService.set(key, seach, timeout);
            } else {
                redisService.set(key, null, 60000);
            }

        } else { //缓存中存在，直接返回
            obj = JSON.parseObject(value, cla);
        }
        return obj;
    }

    //从缓存中获取list
    public <T> List<T> getListCache(String key, Long timeout, Class<T> cla,
FunctionListCache<T> function) {
        List<T> list = null;
        String value = redisService.get(key);
        if (StringUtils.isEmpty(value)) {
            list = function.getCache();
            if (list.isEmpty()) {
                redisService.set(key, null, 60000);
            } else {
                String val = JSONObject.toJSONString(list);
                redisService.set(key, val, timeout);
            }

        } else {
            list = JSON.parseArray(value, cla);
        }
        return list;
    }
}
```

1.5、使用自定义缓存

1.5.1、自定义缓存key的定义方式为　public　static String USER_KEY = "user::";这样的好处是配合 spring 缓存注解 @CacheEvict(value = "sys_role",allEntries = true)可以方便的清除我们的自定义缓存

1.5.2、使用自定义缓存

```
 String key = KeyCode.USER_KEY+username;
        SysUser user = cacheService.getEntityCache(key, 60L, SysUser.class, () ->
userService.getUserByUserName(username));
```

```
 //2.查询用户的权限
        String pkey = KeyCode.PERMISSION_KEY+user.getId();
        List<Permission> permissionList = cacheService.getListCache(pkey, 60L,
Permission.class, () -> permissionService.selectPermissionByUserId(user.getId()));
```

第 53讲 课程总结