# 一、 vue-cli 脚手架工具

我们后端使用maven来创建项目，有两个目的，一是通过maven的依赖机制，能够快速的管理依赖。二是通过maven来确定项目的结构，所谓项目的结构就是项目里有哪些文件和文件夹，文件夹是一个怎样的层级关系。

问题来了？ 一个vue项目里的项目结构应该是什么样的？ 我们能不能快速的获得这样的项目结构呢？

可以通过vue-cli这种脚手架工具来解决这样的问题。 vue-cli里存放了很多常用的项目骨架，直接拿来用就可以搭建出一个拥有比较成熟的项目结构的项目。

## 1.安装node.js

要想使用vue-cli,得先安装node.js

node.js是一个可以让前端运行在node.js提供的服务器上的一个工具，换句话说，就是提供了一个服务器。

官网下载即可：

<http://nodejs.cn/download/>

测试node.js是否安装成功： cmd中输入命令： node -v查看版本，如果看到版本，就表示安装成功。

## 2.使用node.js安装vue-cli

在cmd使用命令：

npm install vue-cli -g

npm: 是指明使用node.js的命令

install 安装

vue-cli 要安装的工具

-g 全局安装

如果使用npm官方镜像速度比较慢，可以安装淘宝镜像

npm install -g cnpm --registry=https://registry.npm.taobao.org

之后使用npm命令时就可以替换成cnpm

cnpm install vue-cli -g

cnpm install

cnpm run dev

## 3. 使用vue-cli下载项目骨架搭建我们的项目

使用以下命令可以列出当前官方提供的骨架，可以使用这些骨架来搭建我们的项目

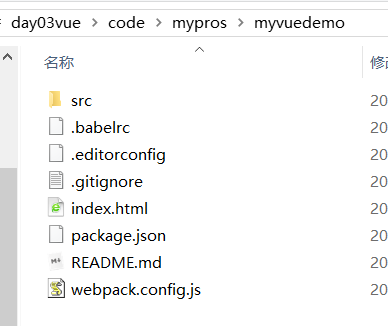
vue list

使用命令来搭建项目骨架。

vue init 骨架名 项目名

比如： vue init webpack-simple myvuedemo

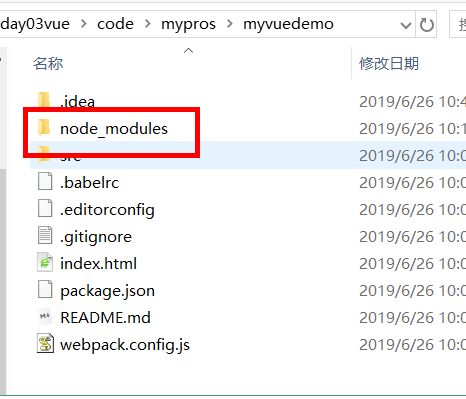
使用命令后当前路径下就会生成一个myvuedemo的文件夹，里面存放着webpack-simple骨架应该有的文件（下载过来的）



## 4.安装项目依赖--npm install

当使用命令下载完项目文件以后，也需要跟maven工程一样，去下载依赖的jar包。

vue需要通过npm install命令来下载依赖。



## 5.使用开发模式来运行项目

使用命令： npm run dev

就能在浏览器中：http://localhost:8080/ 看到当前项目的首页。

# 二、介绍webpack-simple模板项目

项目内的结构

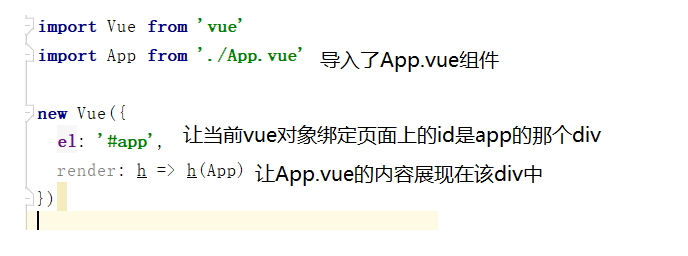
## 1. index.html

无论前端页面内容有多复杂，index.html都只有11行



实际的内容已经被打包进了 /dist/build.js

## 2. main.js文件是整个vue项目的入口js



## 3. App.vue

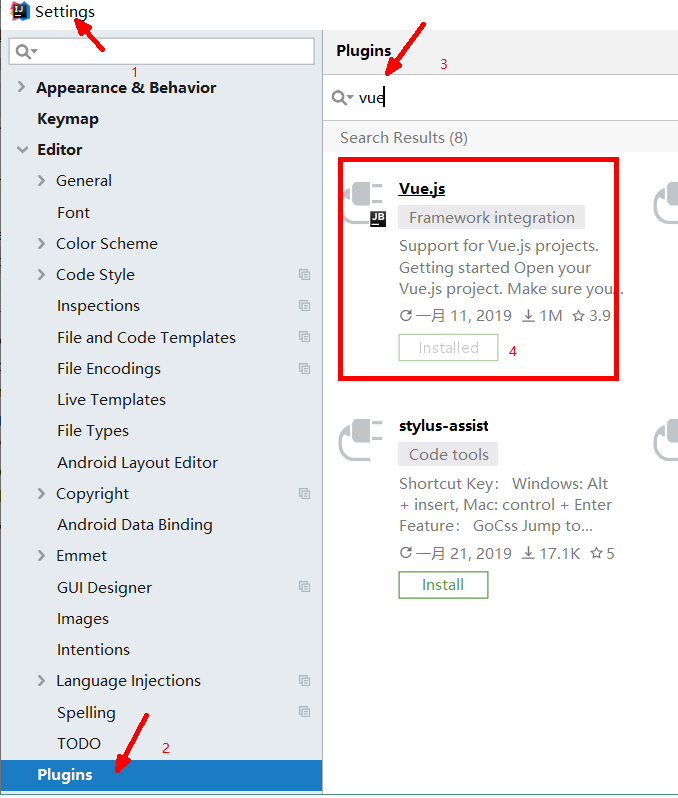
万事万物皆组件。

App.vue 这种以“.vue”为扩展名的内容（文件），实际上就是一个Vue对象。在这里这种文件，也称为Vue组件。

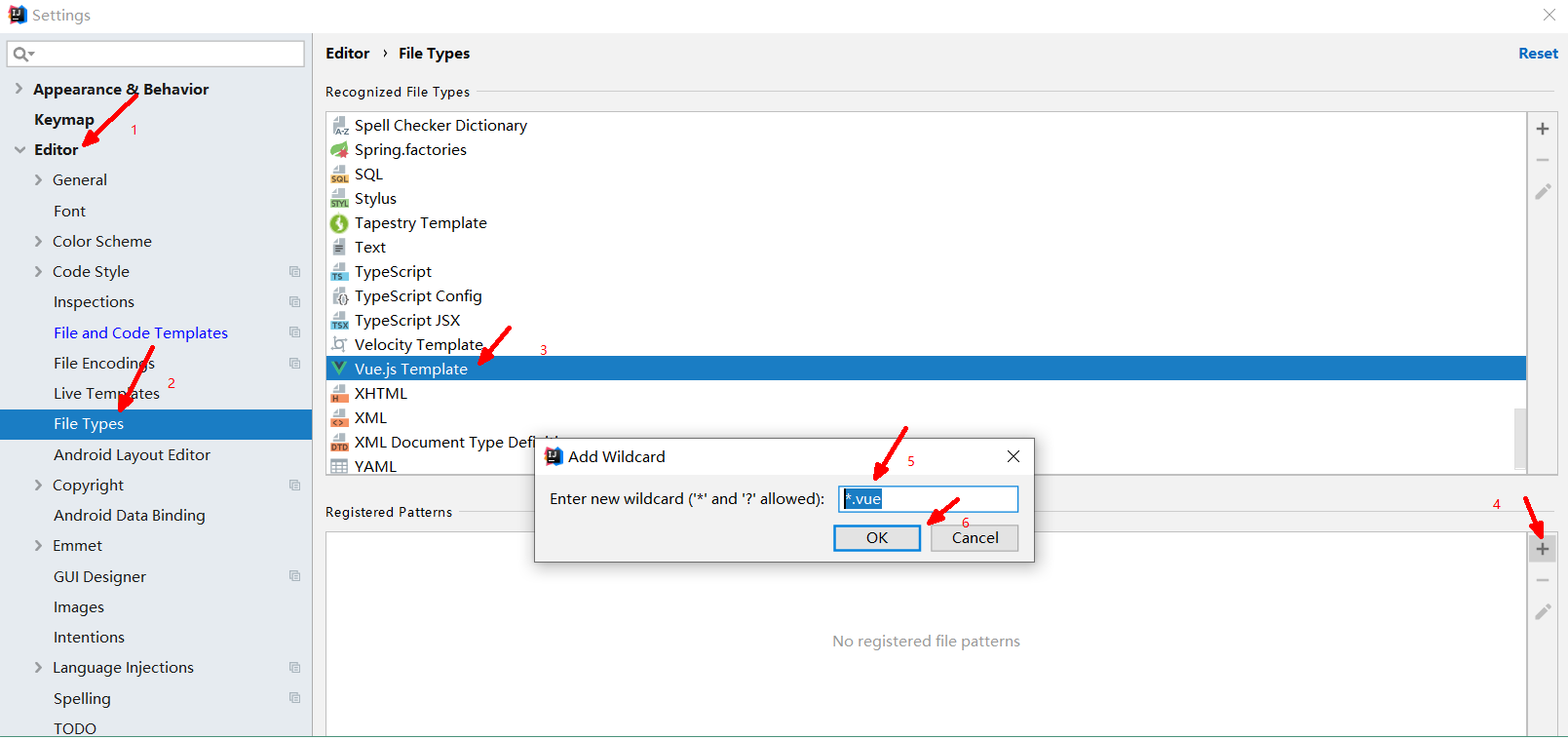
# 三、.vue文件（Vue组件）的组成部分

## 1.让idea支持Vue的插件

### 1）下载vue插件



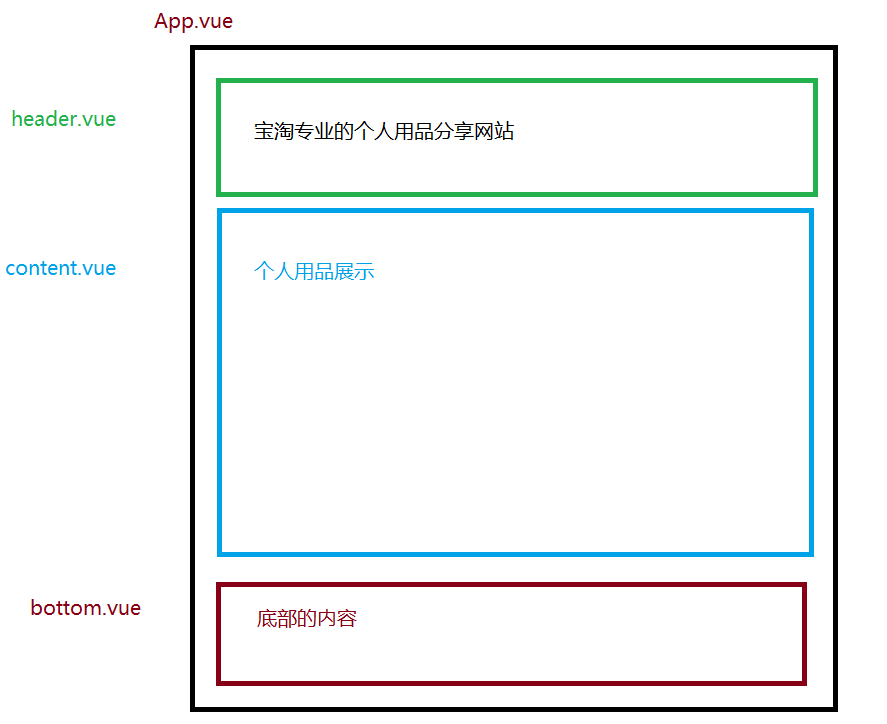
### 2）在idea中配置，能够操作.vue文件



## 2.Vue文件的组成部分



## 3.在vue组件中使用多个vue组件搭建一个页面



### 1）全局注册

是在main.js中，通过import 和 Vue.component配合，来讲一个.vue文件注册成为一个标签。该标签就可以在整个项目中使用。

|  |
| --- |
| **import** Vue **from 'vue' import** App **from './App.vue' import** Header **from './components/Header.vue' import** Content **from './components/Content' import** bottom **from './components/bottom'** *//全局注册了三个组件==》 可以把组件拿来当标签一样的使用* Vue.component(**'MyHeader'**,Header); Vue.component(**'MyContent'**,Content); Vue.component(**'MyBottom'**,bottom);  **new** Vue({  **el**: **'#app'**,  render: h => h(App) }) |

### 2）本地注册

在组件的内部去注册另外一个组件，成为一个标签，这个标签，只在该组件的内部使用，而不能在其他组件内使用。

|  |
| --- |
| <**template**>  <**div id="app"**>  <**MyHeader**></**MyHeader**>  <**MyContent**></**MyContent**>  <**MyBottom**></**MyBottom**>  <**MHeader**></**MHeader**>  </**div**> </**template**>  <**script**>   **import** MHeader **from './components/Header'   export default** {  **name**: **'app'**,  **components**:{  **"MHeader"**:MHeader  }  } </**script**> <**style**>  </**style**> |

## 4.各组件之间的参数传递

### 1）父传子



通过子组件的props部分，来指明可以接收的参数，父组件通过在标签中写明参数的键值对来传递参数。

props是表示一个组件的参数部分，那么props的写法有两种：

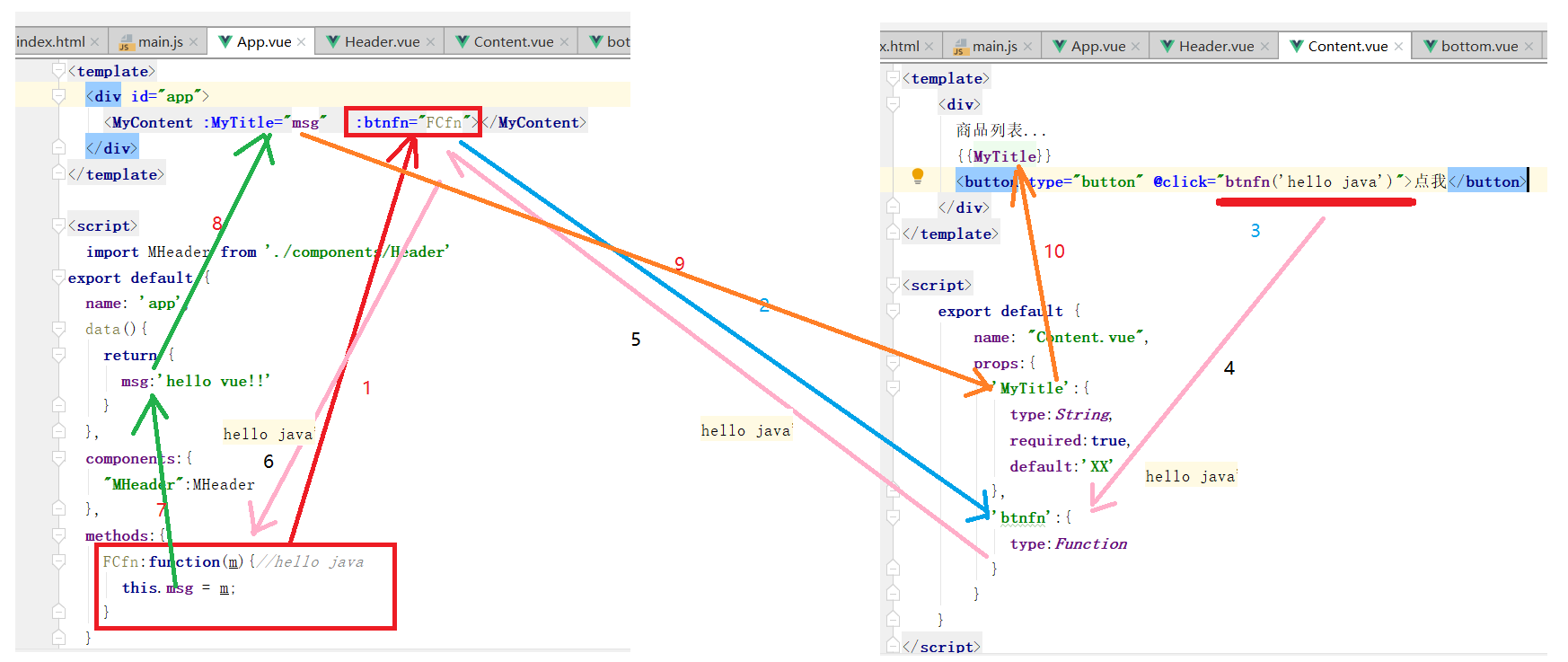
1）props:[参数列表]

比如： props:['MyProp1','MyProp2',...]

2）props:{参数名1:{type:String,required:true,default:'XX'},参数名2:{...}}

### 2）子传父

一个复杂的过程！

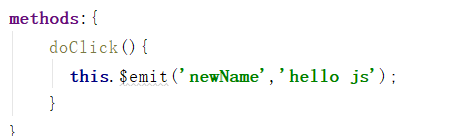


是以事件发射的方式来实现子传父的效果。

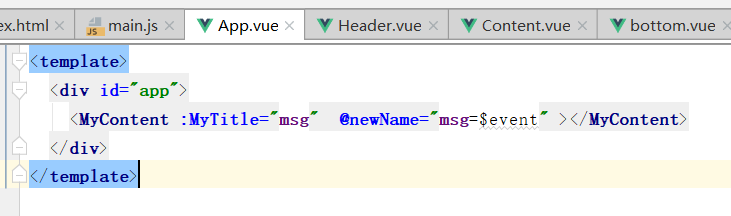
在子组件中，使用 this.$emit("键","值")

在父组件中，子组件的标签中 使用 @键=" msg=$event" 其中$event就能得到值，msg是父组件中vue属性。

子组件



父组件



# 四、Vue中发送Ajax请求

## 1. 安装axios模块

npm install --save axios vue-axios

## 2. 安装依赖

npm install

## 3. 在main.js中使用axios

|  |
| --- |
| **import** Vue **from 'vue' import** App **from './App.vue' import** Header **from './components/Header.vue' import** Content **from './components/Content' import** bottom **from './components/bottom' import** axios **from 'axios' import** VueAxios **from 'vue-axios'** *//全局注册了三个组件==》 可以把组件拿来当标签一样的使用* Vue.component(**'MyHeader'**,Header); Vue.component(**'MyContent'**,Content); Vue.component(**'MyBottom'**,bottom);  Vue.use(VueAxios, axios)  **new** Vue({  **el**: **'#app'**,  render: h => h(App) }) |

## 4.发送Ajax请求：

### 1）发送get请求

|  |
| --- |
| **this**.axios({  **method**:**'get'**,  **url**:**'http://localhost:8090/regist?mail='**+**this**.**mail**+**'&password='**+**this**.**password** //注意 get请求的参数一定是在url之后  }).then(**function** (response) {  ***console***.log(response.**data**) //返回的结果在response里  }); |

### 2）发送post请求

|  |
| --- |
| **this**.axios({  **method**:'**post'**,  **url**:**'http://localhost:8090/regist**,  data:{  mail:this.mail,  password:this.password  }  }).then(**function** (response) {  ***console***.log(response.**data**)  }); |
|  |

## 5.解决跨域问题：

在spring-mvc.xml中加入如下代码

|  |
| --- |
| <**mvc:cors**>  <**mvc:mapping path="/\*\*"  allowed-origins="\*"  allowed-methods="POST, GET, OPTIONS, DELETE, PUT,PATCH"  allowed-headers="Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"  allow-credentials="true"** />  </**mvc:cors**> |