Vue

前后端，Vue+python任意web框架

Vue采用ES6，对java语法.

Vue特点：

组件式开发。很容易改变组件内的数据。

专业：前端页面优化：减少DOM树的操作,vue操作虚拟DOM树，由vue管理真的DOM树.

vue需要node.js，下一步直接安装。

软件管理包, npm

Npm install cnpm -g

-g全局，在任意的目录路径下可以执行相关的指令,相当于设置了path

npm install vue -g

一般情况，希望像django把一个管理目录列表出来。在前端里这种技术叫脚手架

npm install vue-cli -g

利用脚手架可以帮助我们创建项目。

vue路由理解成地址和组件之间的关系。

*react* angular

产生项目脚手架后命令:

Vue init webpack vue项目名称（order\_food)

前后端分离：后端提供一个json接口。

前端后分离：前端有前端路由，不需要后端提供一个地址访问页面。要求后端给一个接口地址，能拿到json数据就可以。

前端必须有页面，然后才是逻辑。

后端有后端路由，不需要考虑为前端提供路由访问页面的事情。也就是后端考虑返回前端模板.

只要后端会做返回json数据的接口。主要注重逻辑.

vue中主要文件.

首先python支持vue

文件：

Package.json

安装的模块

**"dependencies"**: {  
 **"vue"**: **"^2.5.2"**,  
 **"vue-router"**: **"^3.0.1"**}

开发中的文件：

src

所有的组件放在components，这里可以去开发自定义组件

APP.Vue可能添加内容，不用修改

Main.js基本不用理。

route文件夹下建立路由，

理解vue组件式开发。

Vue中template有一个要求：template只能要求有一个标签。

Vue实用技术

Vue如何巧妙布局，有一些组件做一些修改可以使用。

组件使用HTML5+CSS3.

Vue结合于组件开发。

组件无数据，数据需要请求，前端与后端唯一请求数据的途径就是ajax，jquery只用ajax不是异步，同步的ajax，一堆ajax，这些ajax异步，这时异步ajax框架，axios专门做ajax请求。

数据改变，专门操作models.py。专门操作数据的模块:vuex

vue全家桶技术

mobx

Vue+vuex+axios+html5+css3

现在常用的Vue 布局组件模块 element-ui

Vue点食

Vue+elementUI+axios+vuex(结合于mvc来进行设计)

后端: tornado

Vue安装element-ui

Npm install elment-ui --save

--save表示在当前项目使用element-ui模块，不用--save模块不会写到package.json的dependices里面去。

在Vue中使用element-ui设置。

在main.js声明使用element-ui,语句如下

**import** ElementUI **from 'element-ui'  
import 'element-ui/lib/theme-chalk/index.css'**Vue.use(ElementUI)

ElementUI基础的24分栏

相当于ElementUI一行有24列

菜品列表需要动态加载加载，这个组件先用虚拟数据渲染表格，当表格的样式完成以后，（后端完成tornado的接口实现）通过ajax请求数据，把数据用到表格中去。

并发问题，前后分离的问题

列表总结

<el-table>里面包含<el-table-column>

el-table标签中:data就是绑定的data()中返回的数据

el-table-column中prop属性标志就是返回的数据的键的名字

label表格中这一列表示的意义,它会显示在表格的头上.

清楚element-ui的布局中

el-row行, 它会另起一行

el-col列

如果表格中需要显示图片,借用<template>标签模板,如果把标签模块template放在el-table-column当中,调用vue当中的技术,这个技术叫插槽.相当于调用插槽中的scope变量(作用域插槽),scope可以提取出当行这一行数据的值.

<**el-table-column  
prop="images"  
label="菜品图片"**>  
 <**template v-slot="**scope**"**>  
 <**img :src="**scope.row.**images" height="60"**/>  
 </**template**>  
</**el-table-column**>

如果想让el-table这个组件不能随着数据增加而使后面元素顶下去了,可以加height限制el-table的高度.

简单复习

vue做组件

头部<el-header>

左栏导航 <el-menu>里面嵌套<el-menu-item>

修改样式直接修改”.el-menu-item”

中间内容表格列表：常用组件。

<el-table>里面包含<el-table-column>

el-table标签中:data就是绑定的data()中返回的数据

el-table-column中prop属性标志就是返回的数据的键的名字

在prop中显示图片，把<el-table-column>中嵌套<template>，使用v-slot插槽技术，scope变量成为template模板的作用域变量，scope.row表示当前行的数据，$index表示行数（索引）。

尾部<el-footer>

利用一行24个格，使用<el-row> <el-col :span=”4”>进行布局的管理。

初始化项目； vue init webpack 项目名称

运行项目: npm run dev

如何使用vue来做方法

首先在元素上面用@click为元素添加点击事件。

需要控制菜品数量的变量，需要控制的变量内容可以放在data()当中return去初始化定义，注意这里变量和值之间用“：”来联系。

下面书写@click事件的方法

需要加入methods，在methods中定义方法plus\_goods()，后面直接大括号括起来代码。这里完成菜品数量的添加:

this.count\_food++

注意添加完的数据量一定显示在前台上，把{{}}输出在页面中相应位置上

vue中两个组件之间如何把数值传递

父传子或子传父

FoodMain里面的变量传给HelloWorld父

HelloWorld父里变量再传给Footer

父变量中有count\_food变量，如何把父变量放到FooterMain中完成加操作。

父传子还是子传父都在传值，

冒号是双向绑定的意思

HelloWorld总页面对FooterMain列表页面（购买商品的统计量count\_food)即做子传父，也做父传子

价格也是两个孤立组件传值

思路：通过<el-table-column>和<template>共同作用，利用作用域插槽技术，把当前行的价格传给父组件，因为每次点击加上１个，相当于把1个价格传给父组件，再通过父组件传给购物车统计的脚部组件。由此

父组件中还有一个价格变量sum\_price，在FoodMain餐饮列表组件中利用<template v-slot=”scope”＞作用域插槽把变量的值放在plug\_goods的参数中，参数中接收的就是scope.row.price,通过this.$emit方法传给父容器，最后把父容器值sum\_price再传给Footer。

后端传送给前端json菜谱数据

tornado框架

解决并发。

Self.write 写完等待再次写数据

Self.finish 写完结束

解决并发的问题，不能出现self.finish，代码中只能出现self.write

tornado技术实现并发:

1. 最基本的就是用tornado.gen.conrotine的装饰器，结合于yield来使用。
2. 注意，如果实现并发，self.finish是达不到，需要用到self.write

线程无限制开启，正常来讲，是要用一个线程池来维护并发的访问，executor=ThreadPoolExecutor(10000)，可以使用run\_on\_executor这样的装饰器对yield进行并发限制。

vue组件

没有数据，现在需要用tornado提供的接口数据被vue使用，异步ajax框架, axios

vue前端的表格数据一定从后端接口获取的。

Vue在当前项目下安装axios

axios.get(“地址”).then(function(res){

Console.log(res)

});

前端发来一个跨域请求。

不满足同域名，同端口，同协议，就是跨域。

前端和后端解决跨域，

axios格式

Axios.get(“地址”).then(箭头函数）

箭头函数写法:(参数)=>语句

本例中 (res)=>this.foodlist=res.data

Vue把数据拿出单独处理 models.py

Vuex

安装vuex

*/\*逻辑:dispatch一个指令  
\* dispatch(信号，值）\*/*

*/\*  
console.log("-----------------")  
console.log(datas)  
console.log("---------------")  
datas.myvalue=datas.myvalue+1  
this.$emit("transfer",datas.myvalue)  
this.$emit("totalprice",prices)  
\*/*

Vue+Vuex+Axios+elementUI 全家桶