

第12章 基于web编程前后端分离 程序设计 主要内容

- 前后端分离技术
- 技术栈及研发环境
- 服务程序设计与编写
- 客户端程序设计与编写

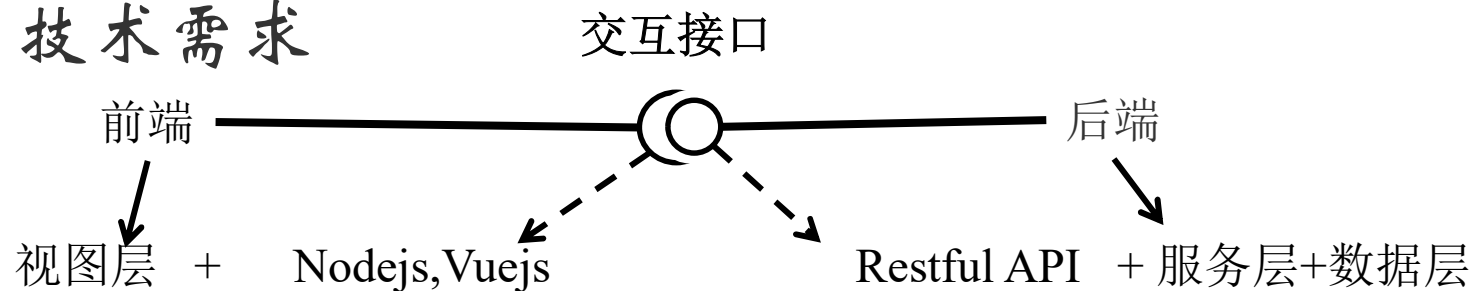
===前后端分离技术===

➤ 前后端分离技术

● 基本概念

- (1)前端：展示数据并与用户进行交互的程序模块(浏览器)
- (2)后端：为前端提供业务逻辑和数据准备程序模块
- (3)战略地位：开发模式与web应用的架构模式

● 技术要求



页面表现，速度流畅，兼容性，用户体验等等。

三高（高并发，高可用，高性能），安全，存储，业务等等

html5, css3, jquery, vuejs, webpack, nodejs, Google V8引擎，模块化，面向切面编程，设计模式，性能优化等

设计模式，spring+springmvc原理及源码，数据库，事务隔离与锁机制，分布式架构，弹性计算架构，微服务架构，性能优化等

云化、微服化、多终端化

===前后端分离技术===

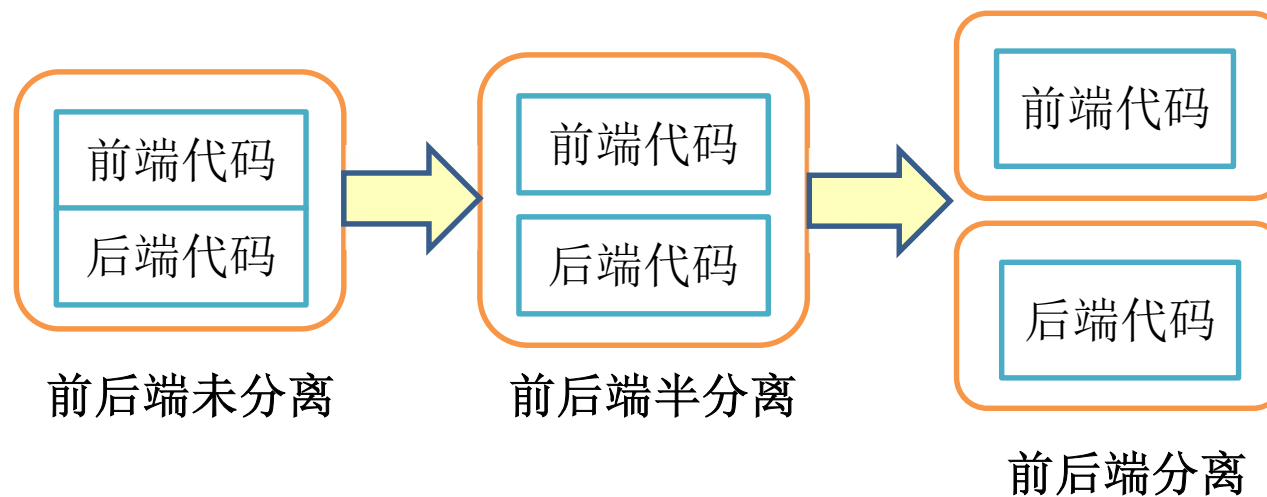
➤ 前后端分离技术

● 新旧技术对比

(1) 交互形式



(2) 代码组织方式

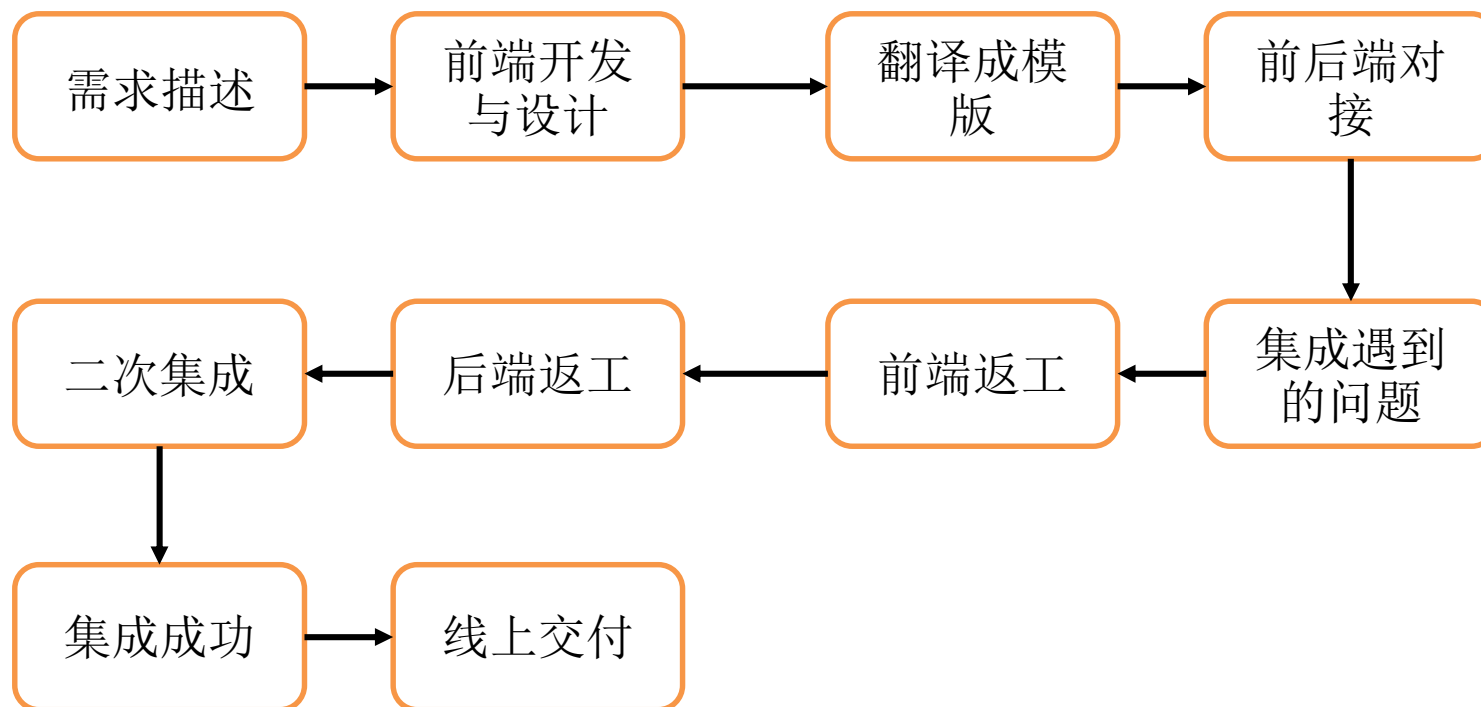


===前后端分离技术===

➤ 前后端分离技术

● 新旧技术对比

(3) 开发模式—传统

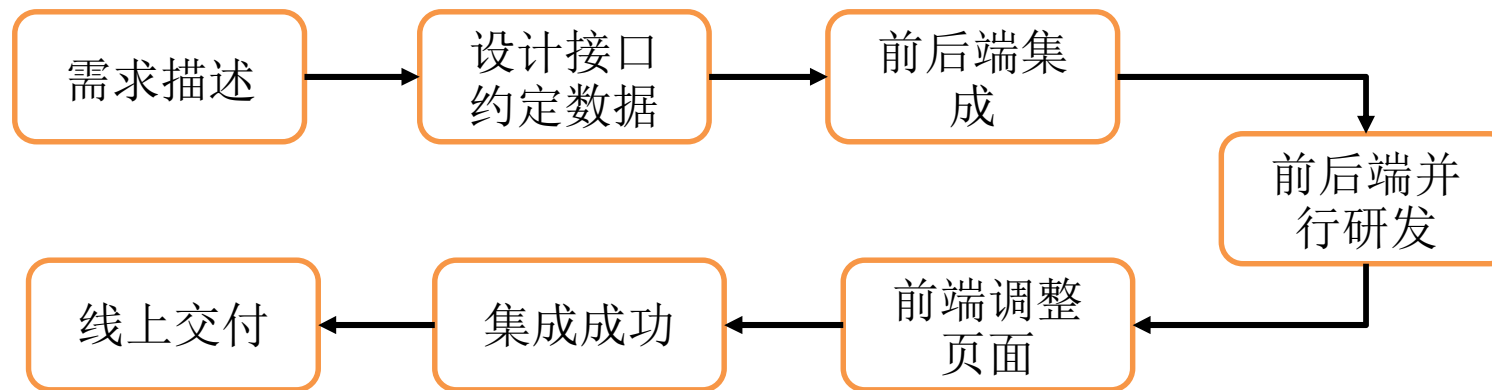


===前后端分离技术===

➤ 前后端分离技术

● 新旧技术对比

(3) 开发模式—新



(4) 前后端分离优势

为优质产品打造精益团队

提升开效率

完美应对复杂多变的前端需求

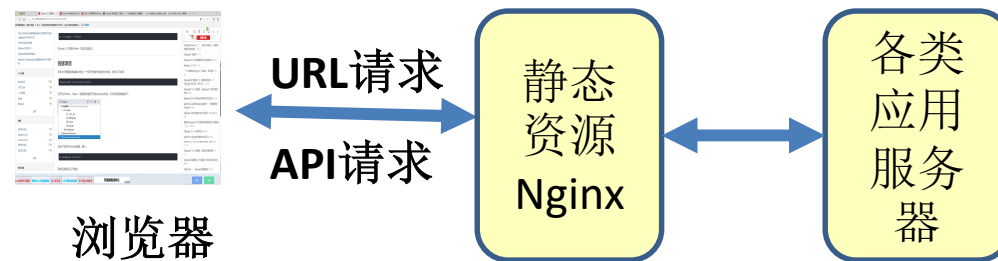
增强代码可维护性

===前后端分离技术===

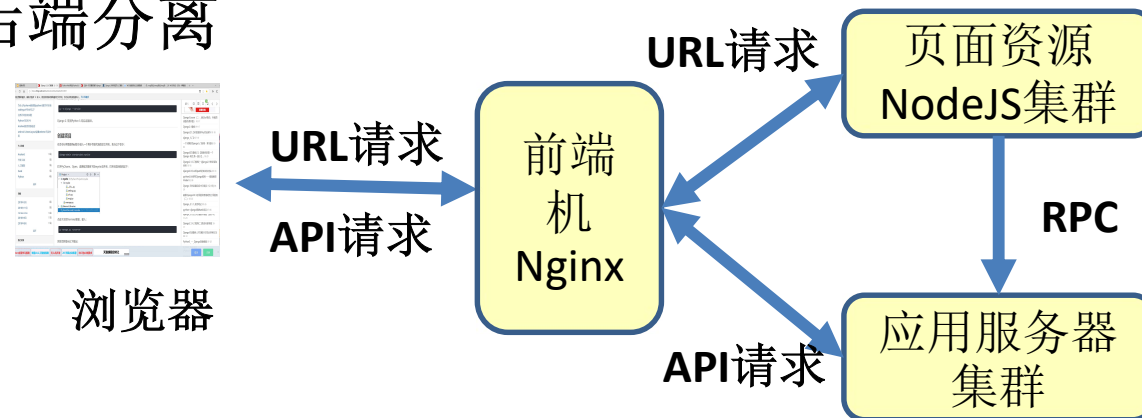
➤ 前后端分离技术

● 新旧技术对比

(5) 架构 传统



前后端分离



第12章 基于web编程前后端分离 程序设计 主要内容

- 前后端分离技术
- 技术栈及研发环境
- 服务程序设计与编写
- 客户端程序设计与编写

===vue.js的安装===

- 前后端分离技术
- 环境

**Python 3.6 +
Django 2.2
Mysql 5.7.3**

**Nodejs 10.15.3
Vue.js 2.9.6
axios**

- 技术栈选择

前端Vue的技术栈: **vue2 + vuex + vue-router +webpack**

UI库: **element-ui**

网络请求: **axios**

前端脚手架构建工具: **vue-cli**

后端技术栈: **Python+Django**

数据库: **MySQL**

➤ 环境

(1) Node.js官网: <https://nodejs.org/en/>

安装后验证: win+r 召唤出cmd: `node -v`

(2) 升级: `npm -g install npm`

(3) 用npm安装cnpm

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

(4) 用cnpm安装脚手架vue-cli

`cnpm install -g vue-cli` (这里用了cnpm,下面也需要,执行完这一步,需要等一下pycharm建立完文件索引,这样下次启动就会快一些)

(5) `cnpm install element-ui`

(6) `cnpm install axios`

(7) `cnpm install vue-resource`

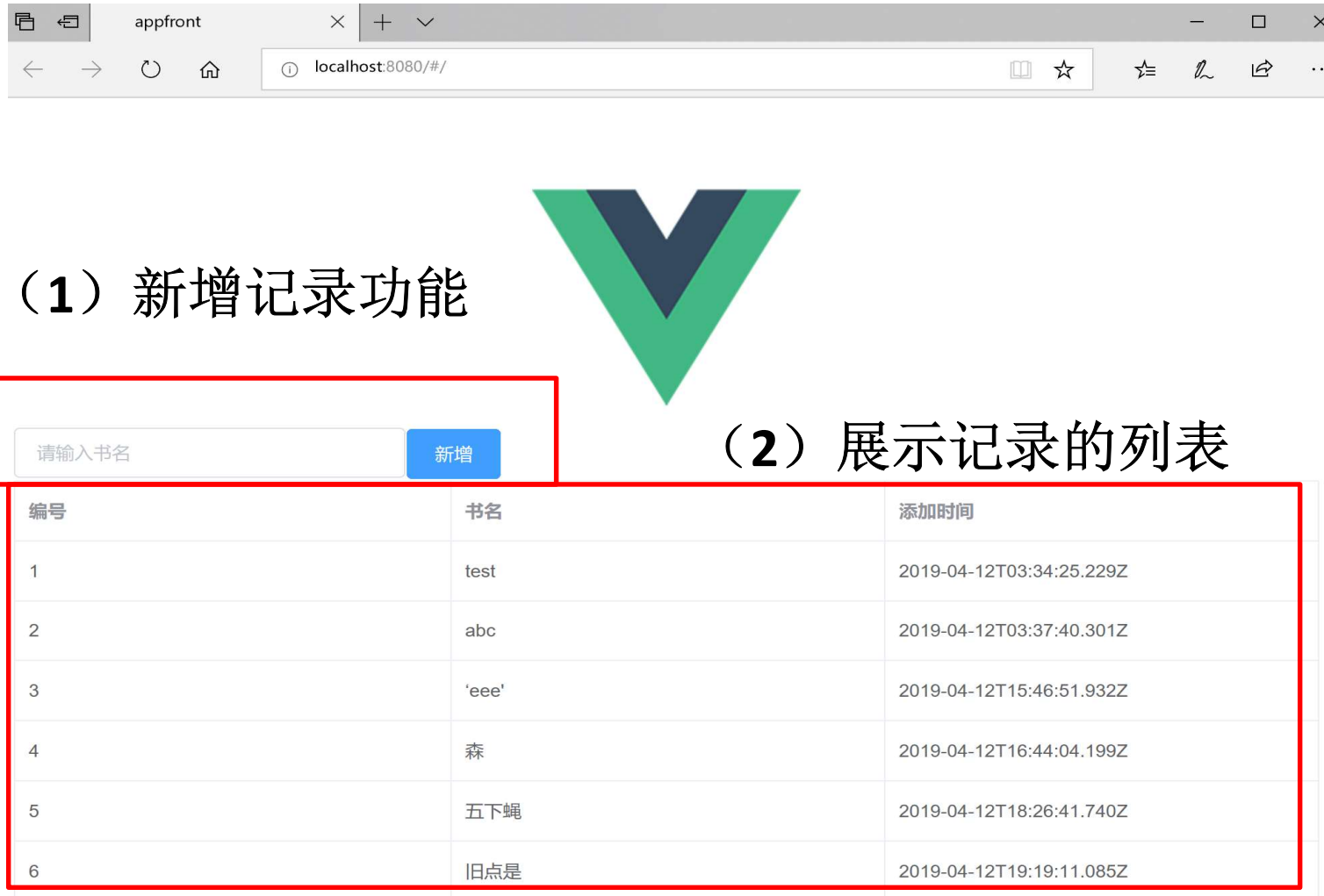
(8) `cnpm install` (这里用cnpm,在创建项目后使用[`vue-init webpack appfront`])

第12章 基于web编程前后端分离 程序设计 主要内容

- 前后端分离技术
- 技术栈及研发环境
- 服务程序设计与编写
- 客户端程序设计与编写

===服务程序设计与编写===

➤ 需求分析



(1) 新增记录功能

请输入书名 新增

(2) 展示记录的列表

编号	书名	添加时间
1	test	2019-04-12T03:34:25.229Z
2	abc	2019-04-12T03:37:40.301Z
3	'eee'	2019-04-12T15:46:51.932Z
4	森	2019-04-12T16:44:04.199Z
5	五下蝇	2019-04-12T18:26:41.740Z
6	旧点是	2019-04-12T19:19:11.085Z

===服务程序设计与编写===

➤ 建立服务器端项目

- 建立项目: **mysite**
- 建立App: **myapp**
- 建立数据库连接
打开**mysite**→**settings.py**

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'testcon',  
        'USER': 'root',  
        'PASSWORD': '123456',  
        'HOST': '127.0.0.1',  
    }  
}
```

===服务程序设计与编写===

➤ 建立服务器端项目

● 注册应用

打开mysite→settings.py

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myapp',  
]
```

===服务程序设计与编写===

➤ 建立服务器端项目

● 创建模型

myapp→models.py

```
from __future__ import unicode_literals
from django.db import models
```

注意次序

```
class Book(models.Model):
    book_name = models.CharField(max_length=64)
    add_time = models.DateTimeField(auto_now_add=True)

    def __unicode__(self):
        return self.book_name
```

运行命令： pip install mysqlclient
python manage.py makemigrations myapp
python manage.py migrate

===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API

myapp→views.py

```
from django.shortcuts import render
from django.views.decorators.http import require_http_methods
from django.core import serializers
from django.http import JsonResponse
import json
from .models import Book
```

===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API

myapp→views.py

```
@require_http_methods(["GET"])
def add_book(request):
    response={}
    try:
        book=Book(book_name=request.GET.get('book_name'))
        book.save()
        response['msg']='sucess'
        response['error_num']=0
    except Exception as e:
        response['msg']=str(e)
        response['error_num']=1
    return JsonResponse(response)
```


===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API

myapp→views.py

```
@require_http_methods(["GET"])
def show_books(request):
    response={}
    try:
        books=Book.objects.filter()
        response['list']=json.loads(serializers.serialize("json",books))
        response['msg']='success'
        response['error_num']=0
    except Exception as e:
        response['msg']=str(e)
        response['error_num']=1
    return JsonResponse(response)
```

===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API --添加接口

(1) 在app目录下, 新增加一个urls.py文件

(2) 增加两个接口

```
from django.conf.urls import url,include  
from myapp import views
```

```
urlpatterns=[  
    url(r'^add_book$',views.add_book,),  
    url(r'^show_books$',views.show_books,),  
]
```

➤ 建立服务器端项目

● 创建服务API --添加接口

- (1) 在app目录下, 新增加一个urls.py文件
- (2) 增加两个接口
- (3) 在mysite增加路由

```
from django.conf.urls import url,include
from django.contrib import admin
from django.views.generic import TemplateView
import myapp.urls

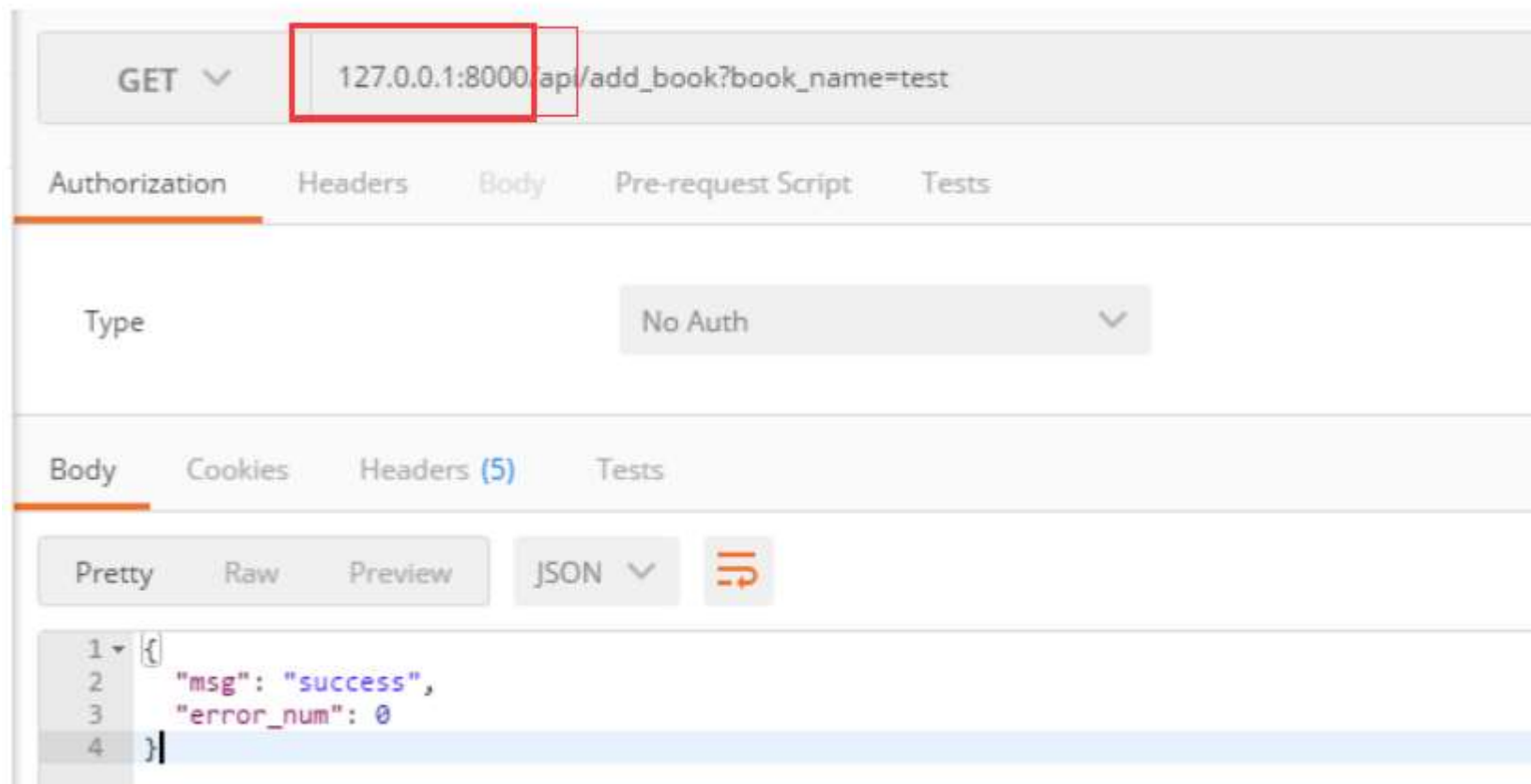
urlpatterns=[
    url(r'admin/',admin.site.urls),
    url(r'^api/',include(myapp.urls)),
    url(r'^$',TemplateView.as_view(template_name="index.html"))
]
```

===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API-测试接口

add_book

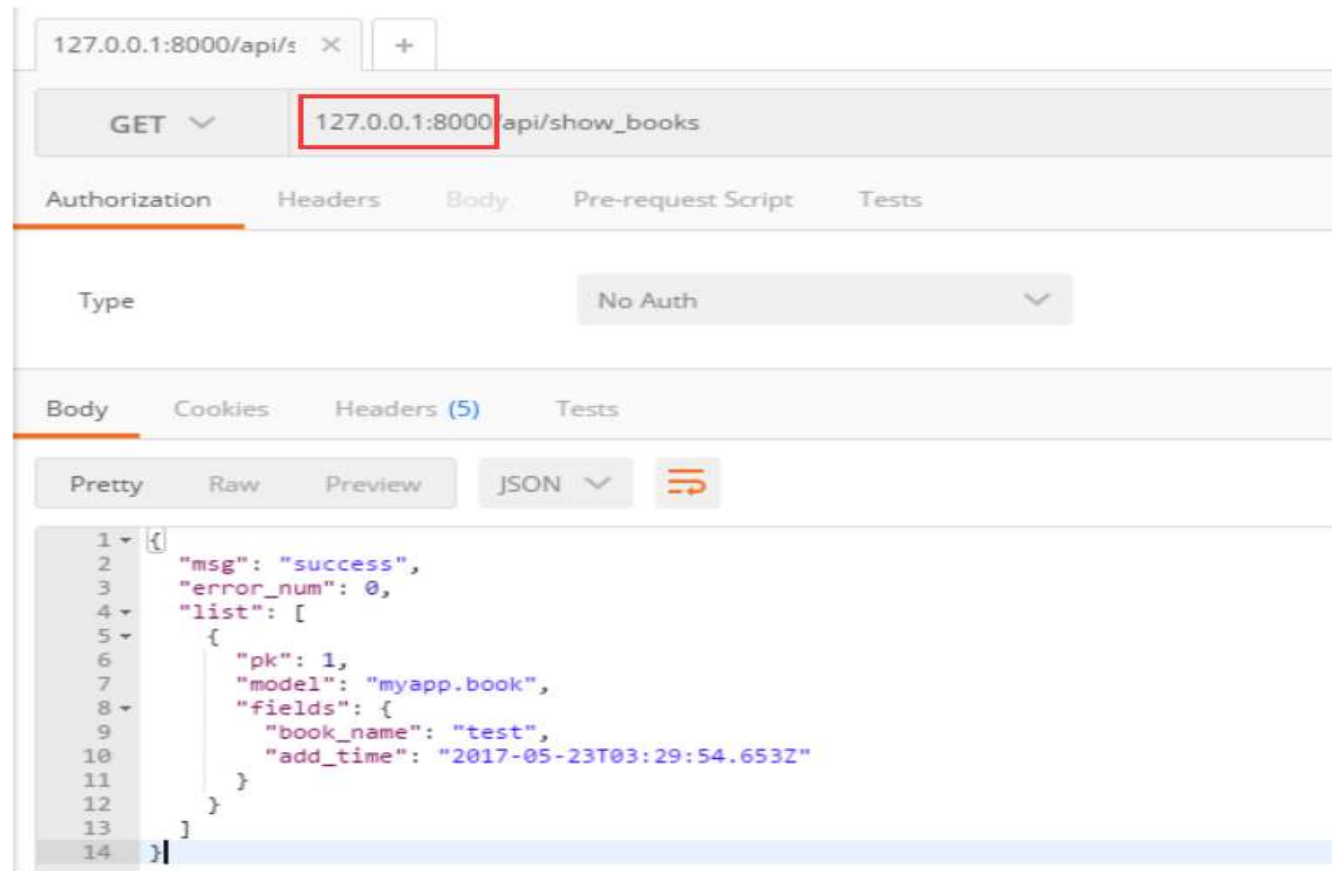


===服务程序设计与编写===

➤ 建立服务器端项目

● 创建服务API-测试接口

show_books



===服务程序设计与编写===

```

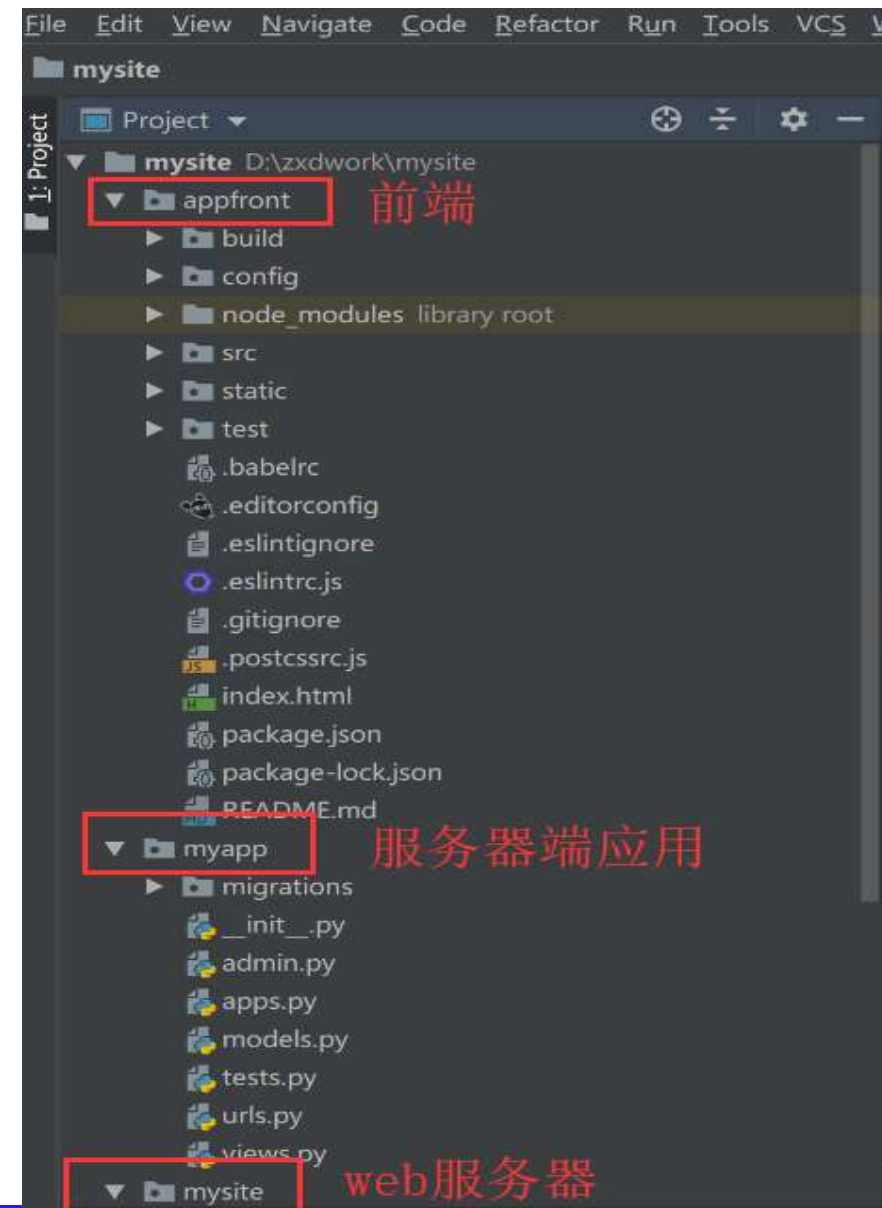
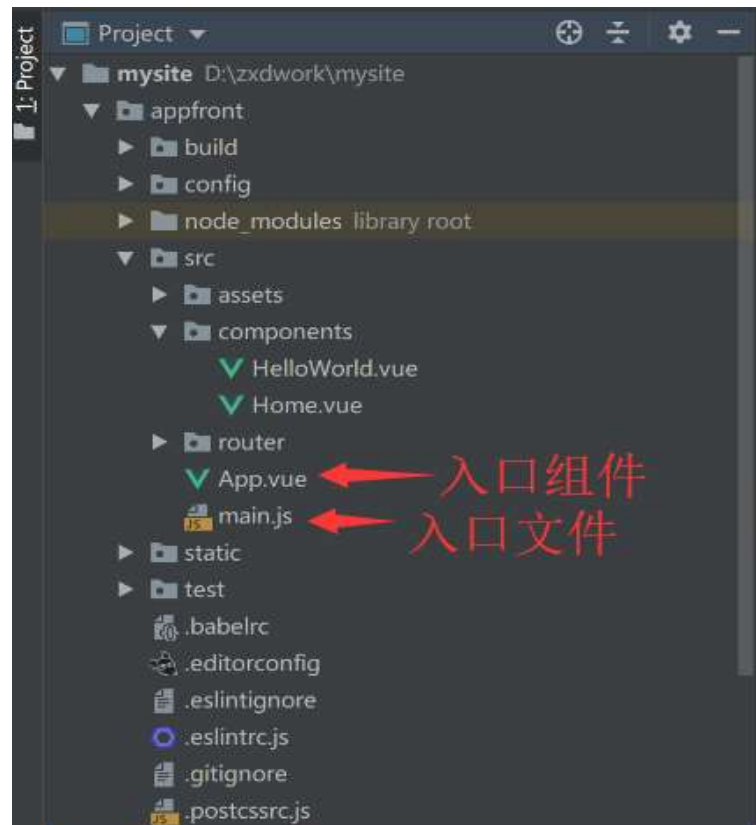
Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Please pick a preset: Manually select features
? Please pick a preset: Manually select features
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter,
? Use history mode for router? (Requires proper server setup for index fallback in production)
>
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter,
? Use history mode for router? (Requires proper server setup for index fallback in production)
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): S
Vue CLI v4.3.1
> ? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Li
? Use history mode for router? (Requires proper server setup for index fallback in production)
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): S
Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processor
? Use history mode for router? (Requires proper server setup for index fallback in p
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by d
? Pick a linter / formatter config: Prettier
? Pick additional lint features: Lint on save
? Pick a unit testing solution: Jest
? Where do you prefer placing config for Babel, ESLint, etc.?
  In dedicated config files
> In package.json

```


===服务程序设计与编写===

➤ 建立客户端项目

- 创建前端工程目录
vue create appfront
or vue ui

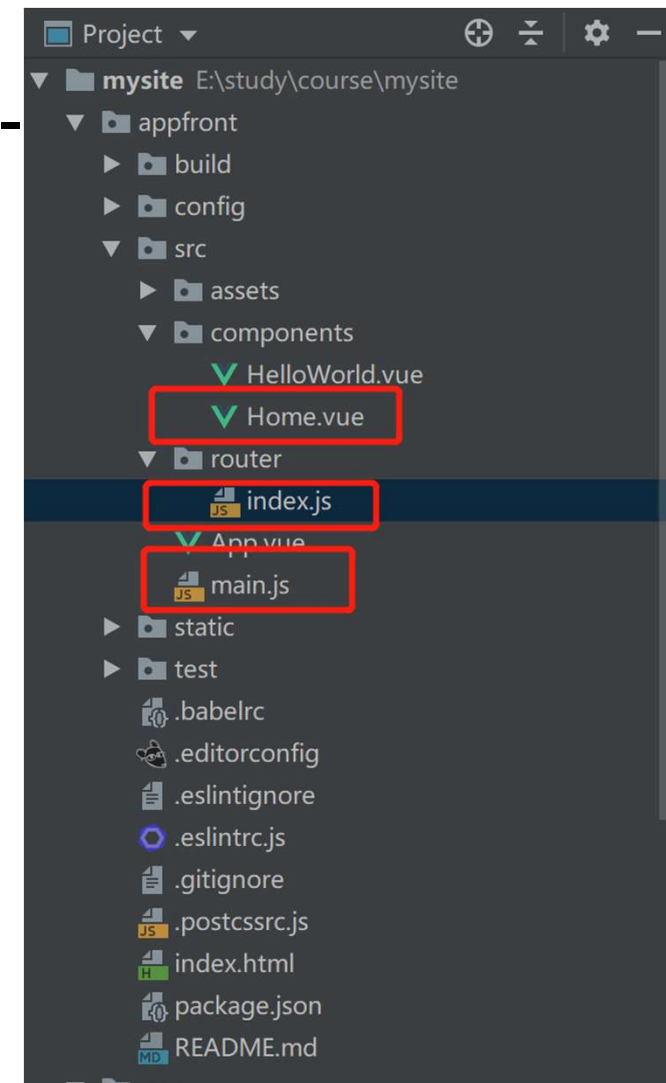


===服务程序设计与编写===

➤ 建立客户端项目

- 创建前端工程目录—引入element-
(1) 修改main.js文件

```
import Vue from 'vue'  
import App from './App'  
import router from './router'  
import VueResource from 'vue-resource'  
import axios from 'axios'  
import ElementUI from 'element-ui'  
import 'element-ui/lib/theme-chalk/index.css'
```



➤ 建立客户端项目

- 创建前端工程目录—引入element-ui和axios
(1) 修改main.js文件

```
Vue.use(ElementUI)
Vue.use(VueResource)
Vue.config.productionTip = false
Vue.prototype.$axios = axios
axios.defaults.headers.post['Content-Type'] = 'application/json'
```

```
new Vue({
  el: '#app',
  router,
  components: { App },
  template: '<App/>'
})
```

➤ 建立客户端项目

- 创建前端工程目录—引入element-ui和axios
(2) 修改index.js文件—前端路由

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from '@components/Home'
var axios = require('axios')
axios.defaults.baseUrl = 'http://127.0.0.1:8020/api'
Vue.use(Router)
export default new Router({
  routes: [
    {
      path: '/',
      name: 'Home',
      component: Home
    }
  ]
})
```

➤ 建立客户端项目

- 创建前端工程目录—引入element-ui和axios
(2) 修改index.js文件—前端路由 跨域

- 安装组件在Django层注入header
pip install django-cors-headers

- 修改settings

```
ALLOWED_HOSTS = ['*']  
CORS_ALLOW_CREDENTIALS = True  
CORS_ORIGIN_ALLOW_ALL = True  
CORS_ALLOW_HEADERS = ('*')
```

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    "corsheaders.middleware.CorsMiddleware",  
    'django.middleware.common.CommonMiddleware',  
    .....  
]
```

```
<template>
  <div class="home">
    <el-row display="margin-top:10px">
      <el-input v-model="input" placeholder="请输入书名"
        style="display:inline-table; width: 30%; float:left">
      </el-input>
      <el-button type="primary" @click="addBook()"
        style="float:left; margin: 1px;">新增</el-button>
    </el-row>
    <el-row>
      <el-table :data="bookList" style="width: 100%" border>
        <el-table-column prop="id" label="编号" min-width="100">
          <template scope="scope"> {{ scope.row.pk }} </template>
        </el-table-column>
        <el-table-column prop="book_name" label="书名" min-width="100">
          <template scope="scope"> {{ scope.row.fields.book_name }} </template>
        </el-table-column>
        <el-table-column prop="add_time" label="添加时间" min-width="100">
          <template scope="scope"> {{ scope.row.fields.add_time }} </template>
        </el-table-column>
      </el-table>
    </el-row>
  </div>
</template>
```

===服务程序设计与编写===

➤ 建立客户端项目

- 创建前端工程目录—编写交互界面
(1) 功能方法

```
<script>
  export default {
    name: 'home',
    data () {
      return {
        input: "",
        bookList: []
      }
    },
    mounted: function () {
      this.showBooks()
    },
```

➤ 建立客户端项目

● 创建前端工程目录—编写交互界面 (1) 功能方法

```
methods: {  
  addBook () {  
    this.$http.get('http://127.0.0.1:8020/api/add_book?book_name=' + this.input)  
      .then((response) => {  
        var res = JSON.parse(response.bodyText)  
        if (res['error_num'] === 0) {  
          this.showBooks()  
        } else {  
          this.$message.error('新增书籍失败，请重试')  
          console.log(res['msg'])  
        }  
      })  
  },  
}
```

➤ 建立客户端项目

● 创建前端工程目录—编写交互界面 (1) 功能方法

```
showBooks () {  
  this.$http.get('http://127.0.0.1:8020/api/show_books')  
    .then((response) => {  
      var res = JSON.parse(response.bodyText)  
      console.log(res)  
      //alert(res.error_num)  
      if (res.error_num === 0) {  
        this.bookList = res['list']  
      } else {  
        this.$message.error('查询书籍失败')  
        console.log(res['msg'])  
      }  
    })  
  }  
</script>
```

===服务程序设计与编写===

➤ 建立客户端项目

● 创建前端工程目录—编写交互界面
(1) 功能方法

```
<style scoped>
  h1, h2 {
    font-weight: normal;
  }
  ul {
    list-style-type: none;
    padding: 0;
  }
  li {
    display: inline-block;
    margin: 0 10px;
  }
  a {
    color: #42b983;
  }
</style>
```


本章小结

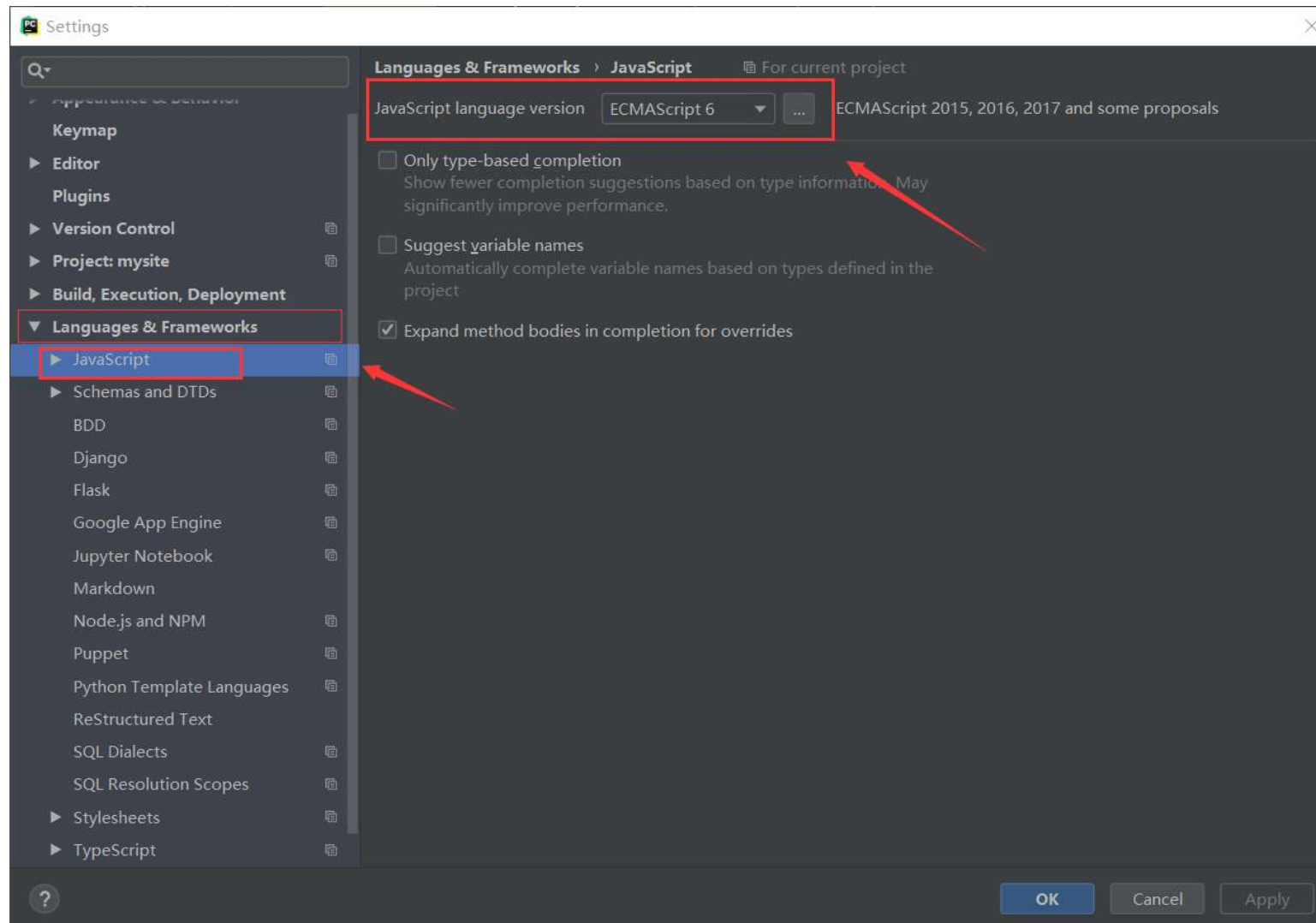
- Django的安装
- Django编程的重要概念
- 基于Django编程的应用实例

第12章 基于web编程前后端分离 程序设计

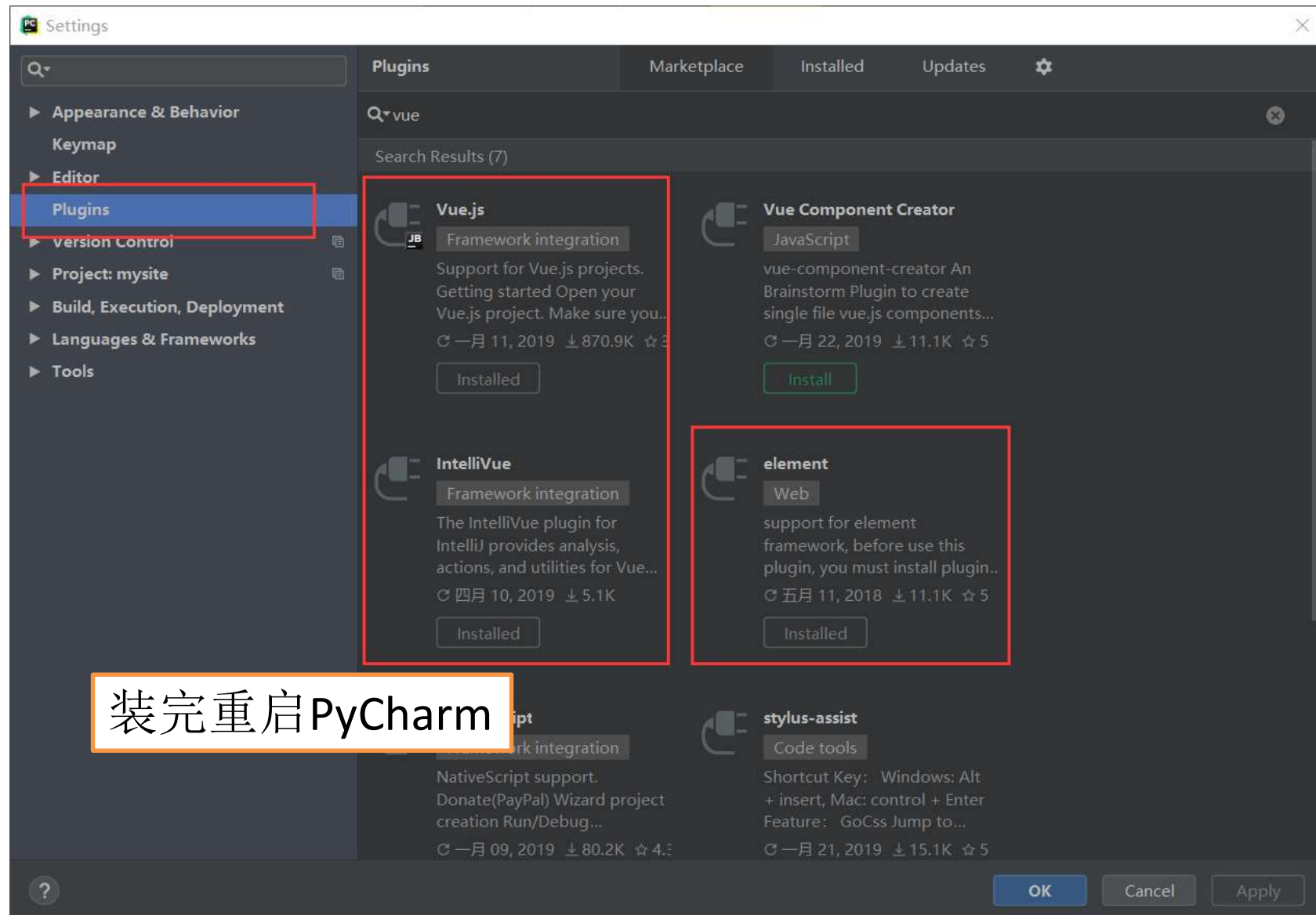
附录

- **Pycharm中安装插件vuejs**
- **postman**

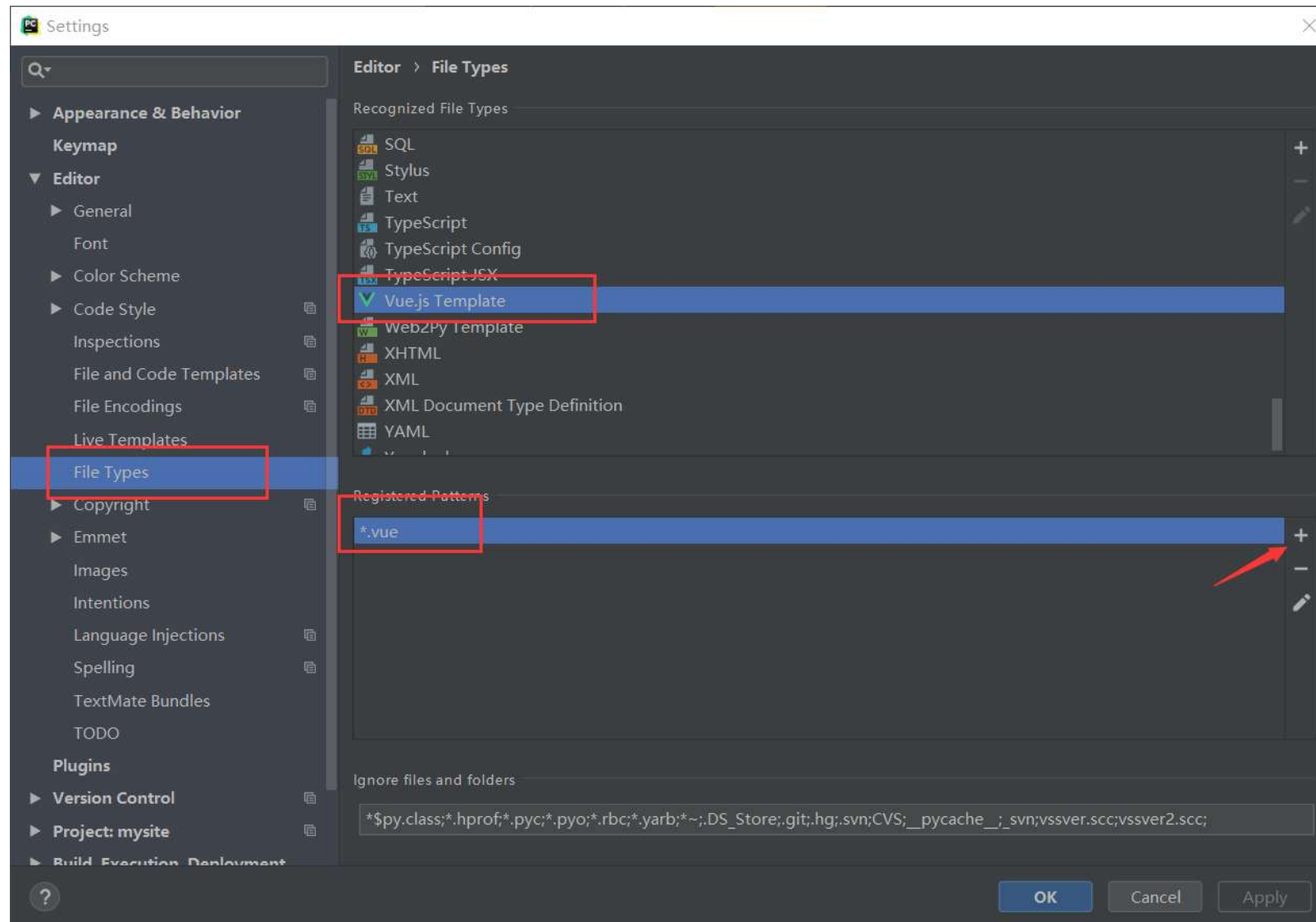
===Pycharm中安装插件vuejs===



===Pycharm中安装插件vuejs===



===Pycharm中安装插件vuejs===



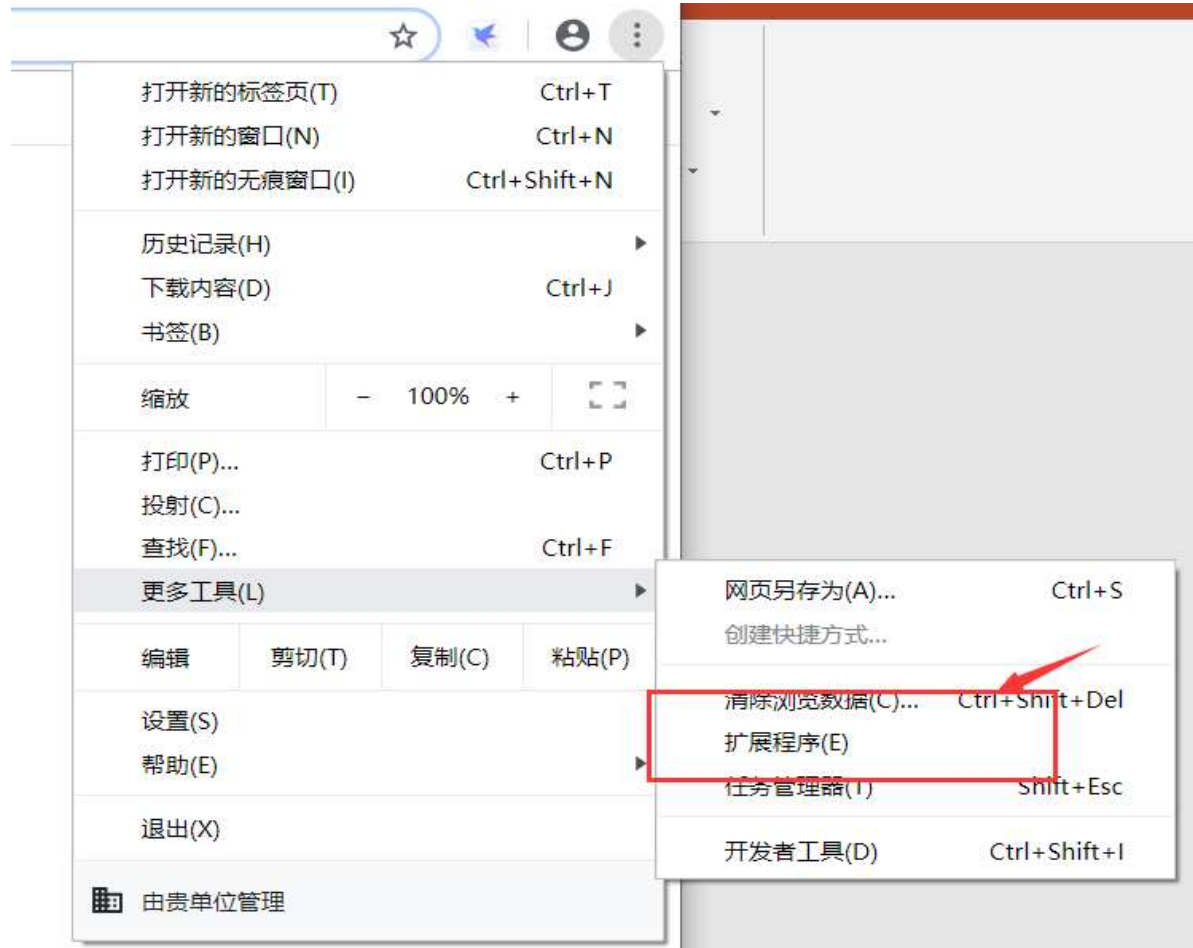
第12章 基于web编程前后端分离 程序设计

附录

- **Pycharm中安装插件vuejs**
- **postman**

===Pycharm中安装插件vuejs===

下载postman软件解压在某个目录下e:\postman





===Pycharm中安装插件vuejs===

