

Zhang-Lab 生信小课堂 第四期

和趣求真  秉实生信

生信数据可视化网站快速开发

Vue && Django

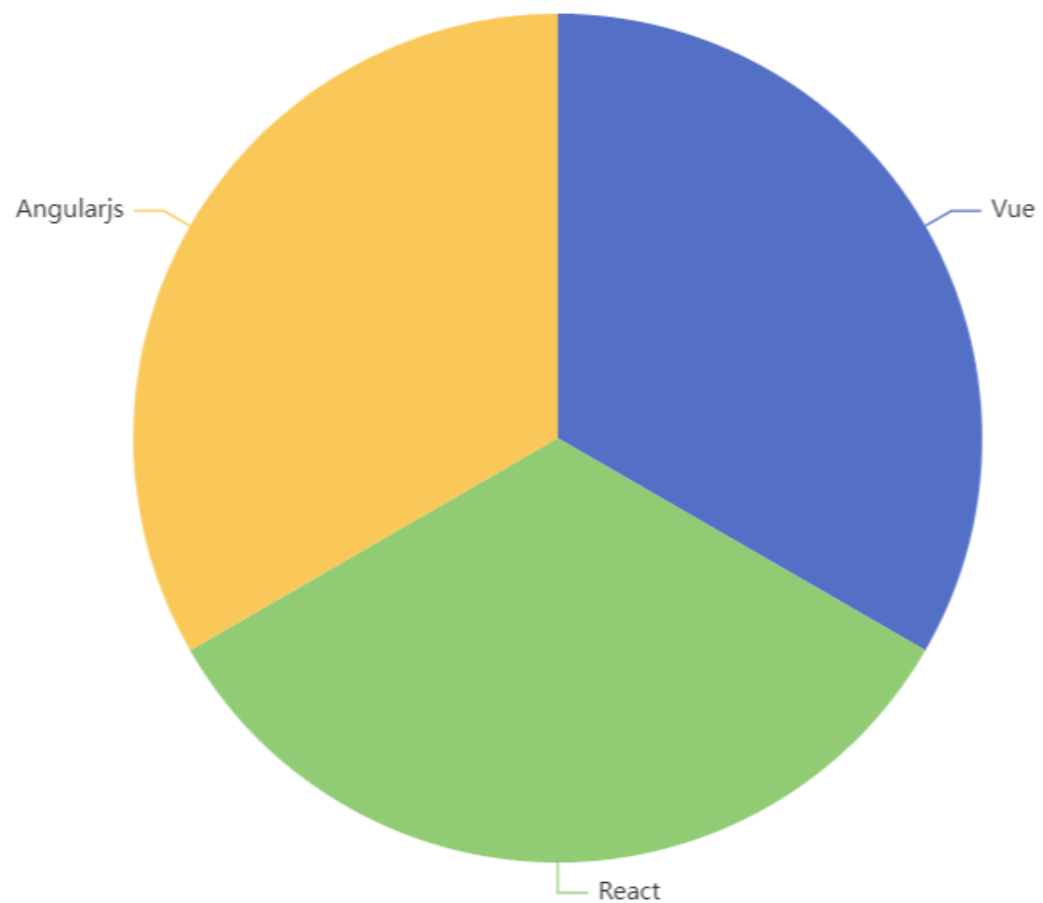
程润东

2022/06/10

Web开发中的三驾马车



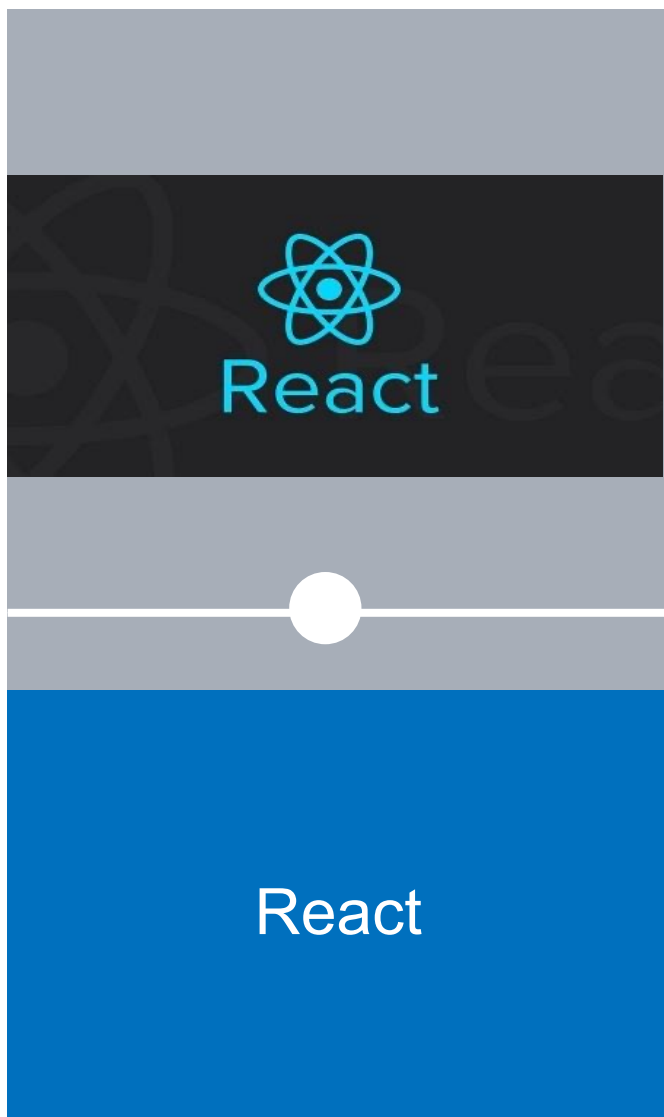
前端框架



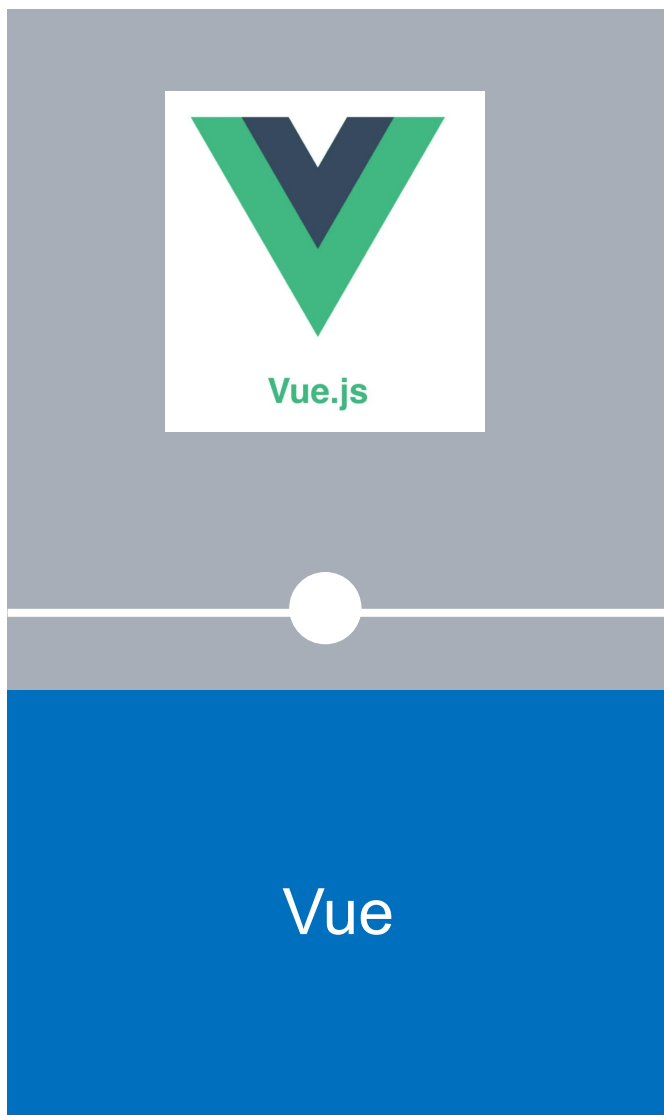


AngularJS

AngularJS诞生于2009年，由Misko Hevery等人创建而成，后来被Google收购，AngularJS弥补了HTML在构建应用方面的不足，它是一款优秀的前端JS框架，通过使用标识符结构来扩展Web应用中的HTML词汇，使得开发者能够使用HTML来声明动态内容。

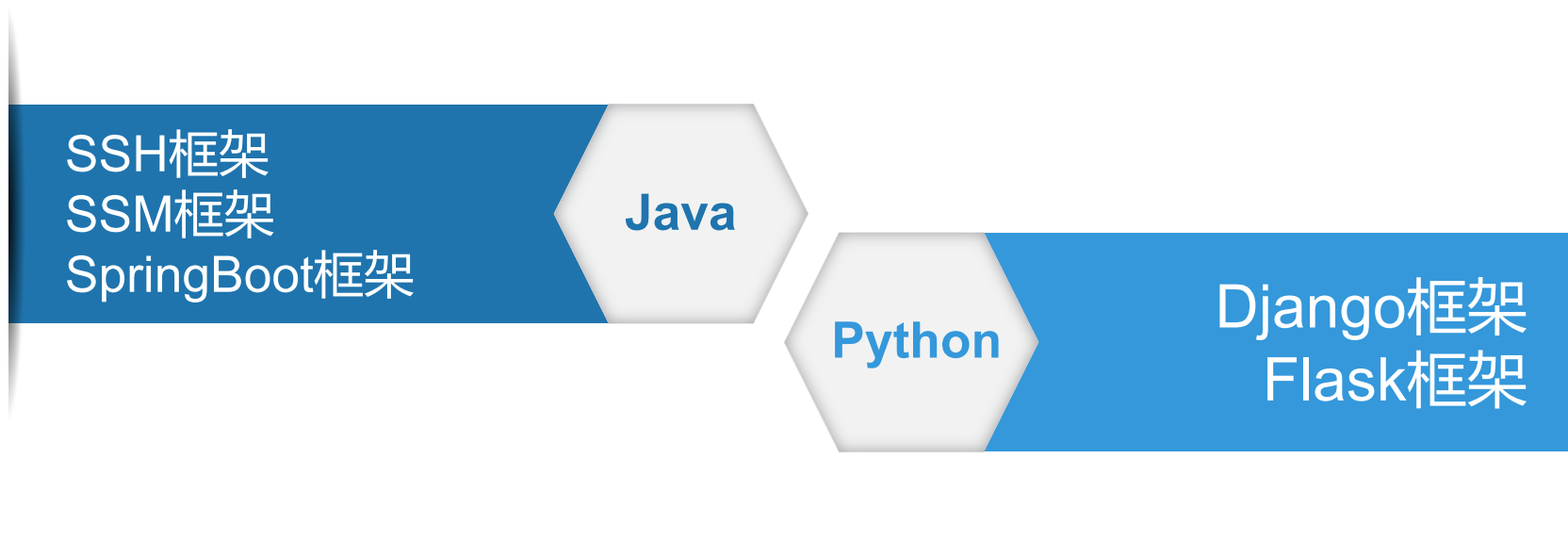


React 是一个 Facebook 和 Instagram 用来创建用户界面的JavaScript 库。很多人认为 React 是 MVC 中的 V（视图）。我们创造 React 是为了解决一个问题：构建随着时间数据不断变化的大规模应用程序。React 可以非常轻松地创建用户交互界面。为你应用的每一个状态设计简洁的视图,在数据改变时 React 也可以高效地更新渲染界面。



Vue.js是一个构建数据驱动的 web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件，相比其它的 MVVM 框架，Vue.js 更容易上手。很多使用过vue的程序员这样评价它，“vue.js兼具angular.js和react.js的优点，并剔除了它们的缺点”。

后端框架



Java框架

SSH框架

SSH框架是以Struts框架进行MVC分离、控制业务跳转，同时使用Hibernate进行持久化，最后配合Spring的统一管理进行实现的开发框架。相对于传统的J2EE开发框架来讲性能相对较高、资源消耗相对较少。

SSM框架

SSM框架看着是Spring+SpringMVC+MyBatis三个框架整合而成的，其实SpringMVC属于Spring框架，所以相当于SSM框架就是Spring和Mybatis两个框架的整合运用。

SpringBoot框架

SpringBoot框架是基于Spring4.0设计的，使用SpringBoot框架进行应用开发能够使用Spring框架所有优秀特性，同时还能够减少各种复杂的配置过程，降低各依赖包的冲突、增强系统的稳定性。

Python框架

Django框架

Django是基于**Python**的免费和开放源代码**Web**框架，它遵循模型-模板-视图(MTV)体系结构模式。**Django**对基础的代码进行了封装并提供相应的 **API**，开发者在使用框架是直接调用封装好的 **API** 可以省去很多代码编写，从而提高工作效率和开发速度。

Flask框架

Flask 是 **Python** 编写的一种轻量级 (微) 的 **Web** 开发框架，只提供 **Web** 框架的核心功能，较其他类型的框架更为的自由、灵活、更加适合高度定制化的 **Web** 项目。

Tornado框架

Tornado 是 **Python** 编写的一个强大的可扩展的 **Web** 服务器，在处理高网络流量的时候表现的足够强大，但是在创建的时候，和 **Flask** 类似又足够轻量，并且可以被用到大量的工具当中。

数据库



关系型数据库

关系型数据库是指采用了关系模型来组织数据的数据库。简单来说，关系模式就是二维表格模型。

主要代表

SQL server, Oracle, Mysql



非关系型数据库

NoSQL非关系型数据库，主要是指那些非关系的、分布式的，且一般不保证ACID的数据存储系统。

主要代表

MongoDB, Redis。

关系型数据库

SQL Server数据库

SQL Server是由微软开发的数据库管理系统，是Web上最流行的用于存储数据的数据库，它已广泛用于电子商务、银行、保险、电力等与数据库有关的行业。

MySQL数据库

MySQL是一个快速的、多线程、多用户和健壮的SQL数据库服务器。MySQL服务器支持关键任务、重负载生产系统的使用，也可以将它嵌入到一个大配置(mass-deployed)的软件中去。

Oracle数据库

甲骨文公司的一款关系数据库管理系统。它是在数据库领域一直处于领先地位的产品。可以说Oracle数据库系统是目前世界上流行的关系数据库管理系统，系统可移植性好、使用方便、功能强，适用于各类大、中、小、微机环境。

非关系型数据库

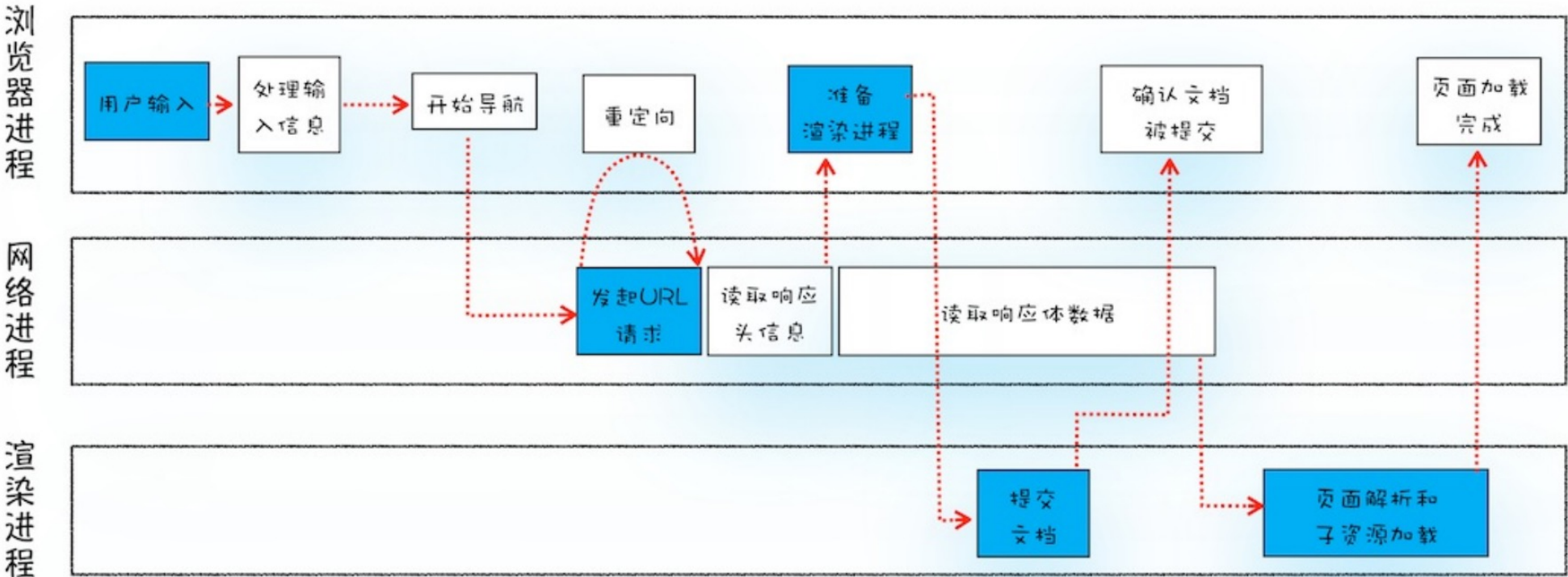
Mongodb数据库

MongoDB是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。

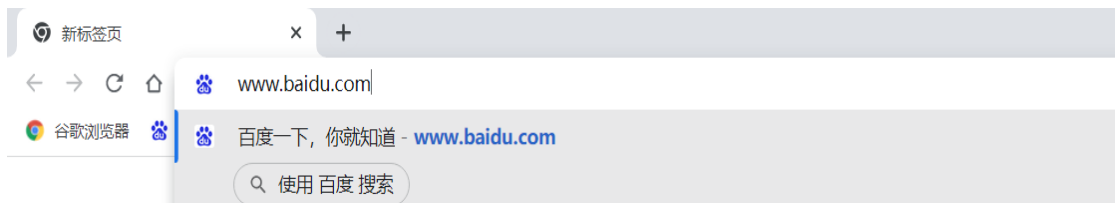
Redis数据库

redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash(哈希类型)。

从输入URL到页面展示完整流程



从输入URL到页面展示完整流程



```
worker_processes 3;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;
    client_max_body_size 2024m;>

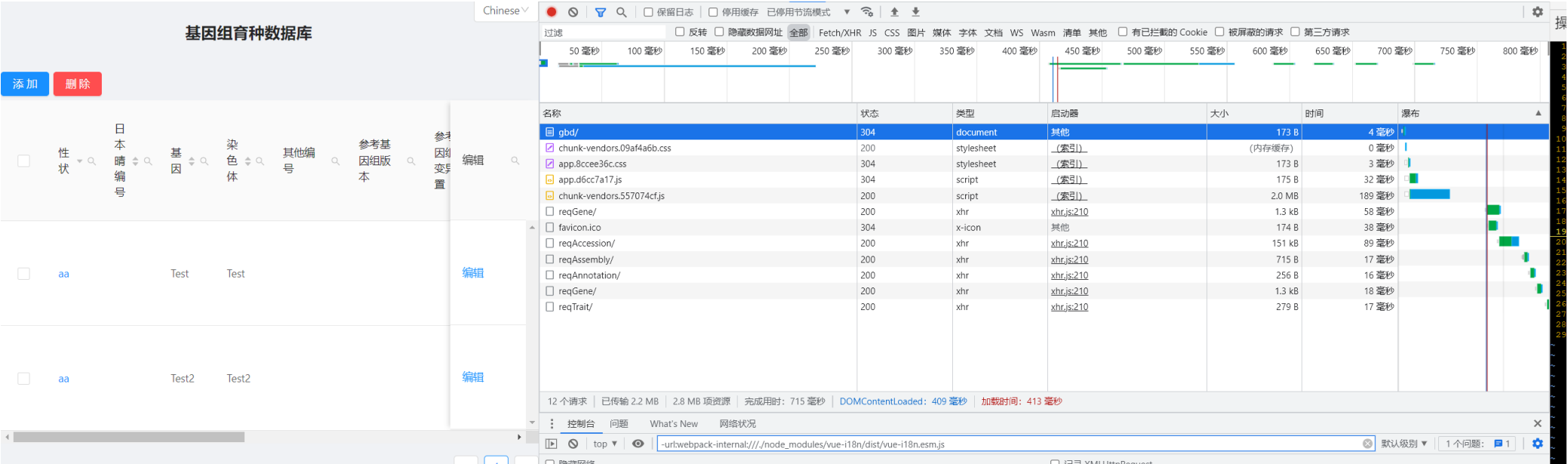
    server {
        listen 2022; # 端口号
        server_name 211.69.128.139; # 服务器IP地址或域名

        >---
        location / {
            root /usr/share/nginx/html; # 网站主页路径
            index index.html index.htm; # index.html为网站主
            >-}

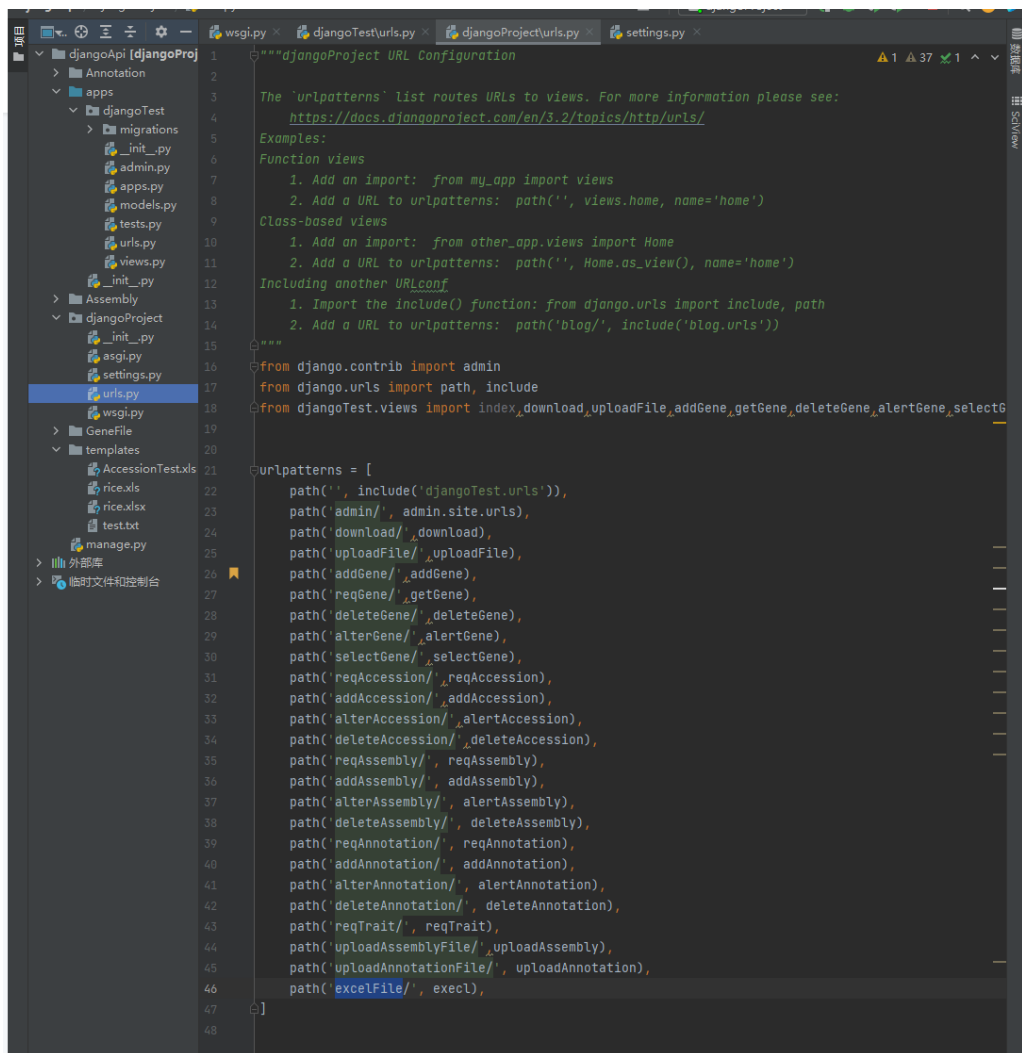
            >---location ^~ /api/ {
            >---#proxy http version 1.1;
            >---#proxy_set_header Connection "";
            >---#rewrite ^/api/(.*)$ /gbdApi/$1 break;
            >---proxy_pass https://211.69.128.139/gbdApi/;
            >---
            >---}

            error_page 500 502 503 504 /50x.html;-
        }
    }
}
```

从输入URL到页面展示完整流程

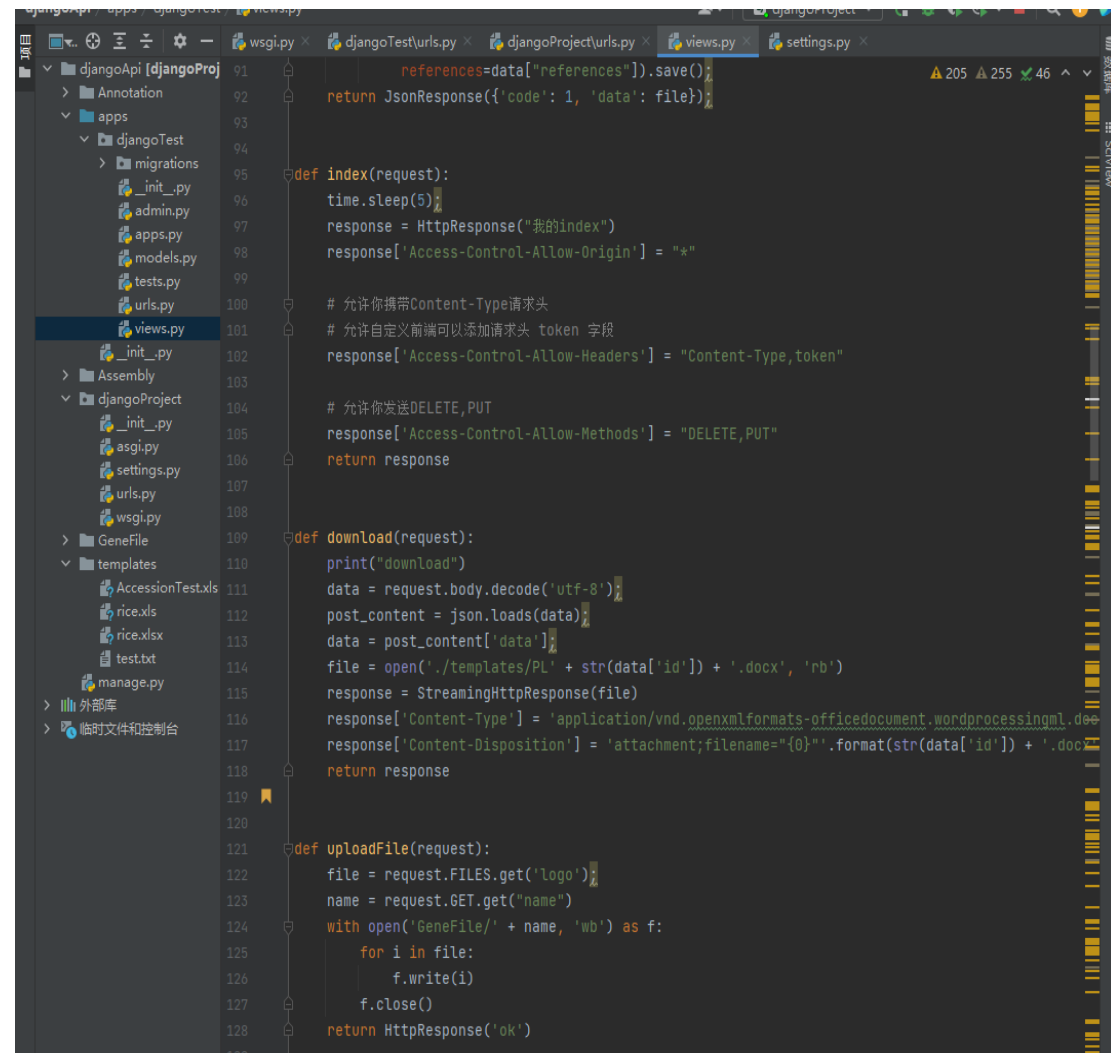


从输入URL到页面展示完整流程



This screenshot shows the `urls.py` file for the `djangoProject` app. The file is titled "djangoProject URL Configuration". It includes comments explaining the `urlpatterns` list and provides examples for function views and class-based views. The `urlpatterns` list is defined at the bottom, including paths for `admin`, `download`, `uploadFile`, `addGene`, `getGene`, `deleteGene`, `alertGene`, `selectGene`, `reqAccession`, `addAccession`, `alterAccession`, `deleteAccession`, `reqAssembly`, `addAssembly`, `alterAssembly`, `deleteAssembly`, `reqAnnotation`, `addAnnotation`, `alterAnnotation`, `deleteAnnotation`, `reqTrait`, `uploadAssemblyFile`, `uploadAnnotationFile`, and `excelFile`.

```
1 """djangoProject URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.2/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 from django.contrib import admin
17 from django.urls import path, include
18 from djangoTest.views import index, download, uploadFile, addGene, getGene, deleteGene, alertGene, selectGene
19
20 urlpatterns = [
21     path('', include('djangoTest.urls')),
22     path('admin/', admin.site.urls),
23     path('download/', download),
24     path('uploadFile/', uploadFile),
25     path('addGene/', addGene),
26     path('reqGene/', getGene),
27     path('deleteGene/', deleteGene),
28     path('alterGene/', alertGene),
29     path('selectGene/', selectGene),
30     path('reqAccession/', reqAccession),
31     path('addAccession/', addAccession),
32     path('alterAccession/', alterAccession),
33     path('deleteAccession/', deleteAccession),
34     path('reqAssembly/', reqAssembly),
35     path('addAssembly/', addAssembly),
36     path('alterAssembly/', alterAssembly),
37     path('deleteAssembly/', deleteAssembly),
38     path('reqAnnotation/', reqAnnotation),
39     path('addAnnotation/', addAnnotation),
40     path('alterAnnotation/', alterAnnotation),
41     path('deleteAnnotation/', deleteAnnotation),
42     path('reqTrait/', reqTrait),
43     path('uploadAssemblyFile/', uploadAssembly),
44     path('uploadAnnotationFile/', uploadAnnotation),
45     path('excelFile/', excel),
46 ]
```



This screenshot shows the `urls.py` file for the `djangoTest` app. It includes the `urlpatterns` list and the `views.py` file. The `urlpatterns` list is defined at the top, including paths for `index`, `download`, and `uploadFile`. The `views.py` file contains the `index`, `download`, and `uploadFile` functions. The `index` function returns a `HttpResponse` with the text "我的index". The `download` function returns a `StreamingHttpResponse` for a file named `download.docx`. The `uploadFile` function returns a `HttpResponse` with the text "ok".

```
1 """djangoTest URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.2/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 from django.contrib import admin
17 from django.urls import path, include
18 from djangoTest.views import index, download, uploadFile
19
20 urlpatterns = [
21     path('', include('djangoTest.urls')),
22     path('index/', index),
23     path('download/', download),
24     path('uploadFile/', uploadFile),
25 ]
26
27 from djangoTest.views import index, download, uploadFile
28
29 def index(request):
30     time.sleep(5)
31     response = HttpResponse("我的index")
32     response['Access-Control-Allow-Origin'] = "*"
33     # 允许你携带Content-Type请求头
34     # 允许自定义前端可以添加请求头 token 字段
35     response['Access-Control-Allow-Headers'] = "Content-Type,token"
36     # 允许你发送DELETE,PUT
37     response['Access-Control-Allow-Methods'] = "DELETE,PUT"
38     return response
39
40 def download(request):
41     print("download")
42     data = request.body.decode('utf-8')
43     post_content = json.loads(data)
44     data = post_content['data']
45     file = open('./templates/PL' + str(data['id']) + '.docx', 'rb')
46     response = StreamingHttpResponse(file)
47     response['Content-Type'] = 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'
48     response['Content-Disposition'] = 'attachment;filename="{0}"'.format(str(data['id']) + '.docx')
49     return response
50
51 def uploadFile(request):
52     file = request.FILES.get('logo')
53     name = request.GET.get("name")
54     with open('GeneFile/' + name, 'wb') as f:
55         for i in file:
56             f.write(i)
57         f.close()
58     return HttpResponse('ok')
```

从输入URL到页面展示完整流程

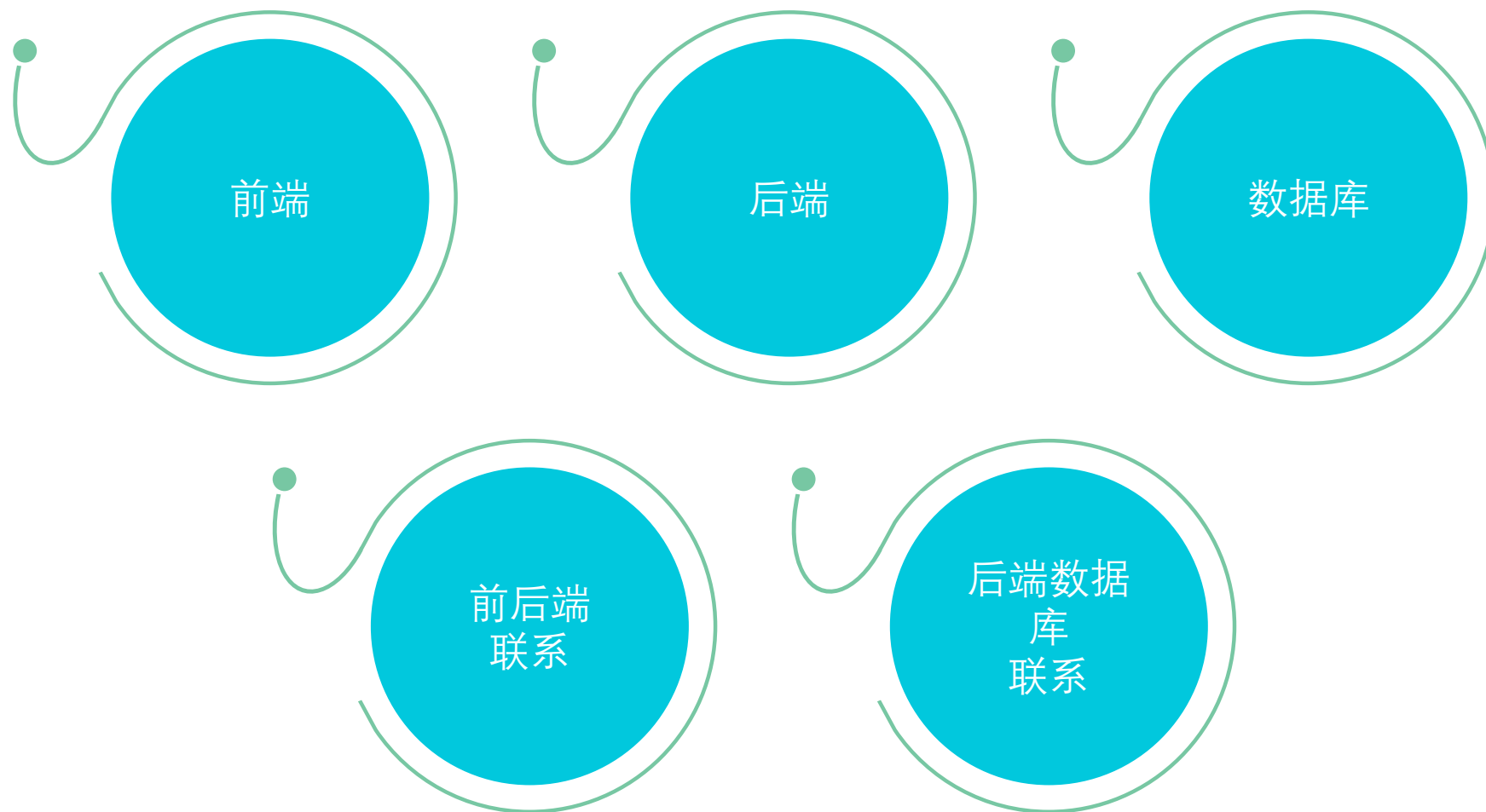
```
export const reqGene = () =>
{
  return requests({
    url: '/reqGene/',
    methods: 'get'
  });
}

export const addGene = (datas) =>
{
  return requests({
    url: '/addGene/',
    method: 'POST',
    data: {
      data: datas
    }
  });
}
```

```
async reqGeneMsg()
{
  this.msg = await reqGene();
  let that = this;
  this.data = this.msg.data.map((item) => {
    console.log(that.AnnotationNameId[item.rap_id]);
    return {
      ...item,
      AnnotationName: that.AnnotationNameId[item.rap_id],
      key: item.id,
      operation: "edit",
    };
  });
}
```

手把手带你从零开始建站

三个部分，两个联系



手把手带你从零开始建站

-前端

1

全局安装vue脚手架 `npm install -g @vue/cli`

2

切换到你要创建项目的目录，使用命令创建项目 `vue create XXX`

3

启动项目 `npm run serve`

手把手带你从零开始建站

-前端

全局安装vue脚手架 `npm install -g @vue/cli` 安装完使用vue测试一下

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1706]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Knight>vue
Usage: vue <command> [options]

Options:
  -V, --version          output the version number
  -h, --help             output usage information

Commands:
  create [options] <app-name>      create a new project powered by vue-cli-service
  add [options] <plugin> [pluginOptions]  install a plugin and invoke its generator in an already created project
  invoke [options] <plugin> [pluginOptions]  invoke the generator of a plugin in an already created project
  inspect [options] [paths...]  inspect the webpack config in a project with vue-cli-service
  serve [options] [entry]        serve a .js or .vue file in development mode with zero config
  build [options] [entry]        build a .js or .vue file in production mode with zero config
  ui [options]                  start and open the vue-cli ui
  init [options] <template> <app-name>  generate a project from a remote template (legacy API, requires @vue/cli-in
it)
  config [options] [value]        inspect and modify the config
  outdated [options]              (experimental) check for outdated vue cli service / plugins
  upgrade [options] [plugin-name] (experimental) upgrade vue cli service / plugins
  migrate [options] [plugin-name] (experimental) run migrator for an already-installed cli plugin
  info                            print debugging information about your environment

Run vue <command> --help for detailed usage of given command.

C:\Users\Knight>
```

手把手带你从零开始建站

-前端

切换到你要创建项目的目录，使用命令创建项目 `vue create XXX`

```
C:\Users\Knight>mkdir VueTest
```

```
C:\Users\Knight>cd VueTest
```

```
C:\Users\Knight\VueTest>vue create project
```



```
Vue CLI v4.5.15
[ ] Creating project in C:\Users\Knight\VueTest\project.
[ ] Installing CLI plugins. This might take a while...
```

```
added 1300 packages in 24s
```

```
🔍 Invoking generators...
🔍 Installing additional dependencies...
```

```
added 55 packages in 3s
```

```
[ ] Running completion hooks...
```

```
🔍 Generating README.md...
```

```
🔍 Successfully created project project.
🔍 Get started with the following commands:
```

```
$ cd project
$ npm run serve
```

手把手带你从零开始建站

- 前端

按照上面提示输入命令，就可以访问localhost:8080就可以得到下面的画面

```
C:\Users\Knight\VueTest>cd project
C:\Users\Knight\VueTest\project>npm run serve_
```

```
DONE Compiled successfully in 922ms
```

```
App running at:
- Local:   http://localhost:8080/
- Network: unavailable
```

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



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](#)

手把手带你从零开始建站

-后端

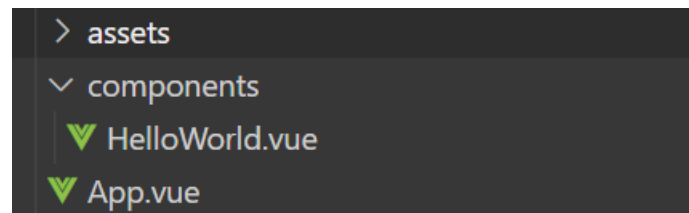
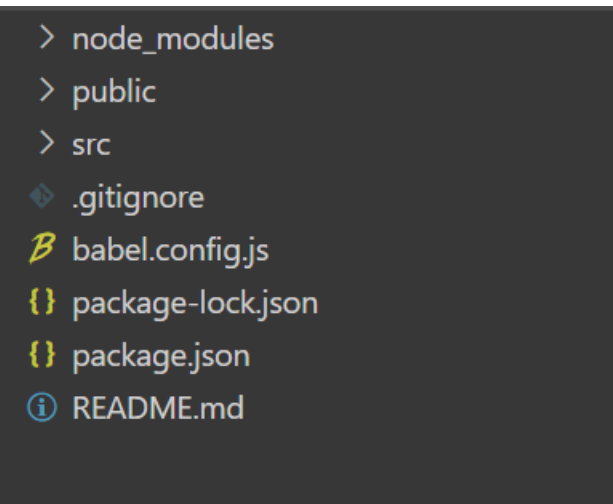
pycharm :

<https://www.jetbrains.com/pycharm/download/#section=windows>

vscode :

<https://code.visualstudio.com/>

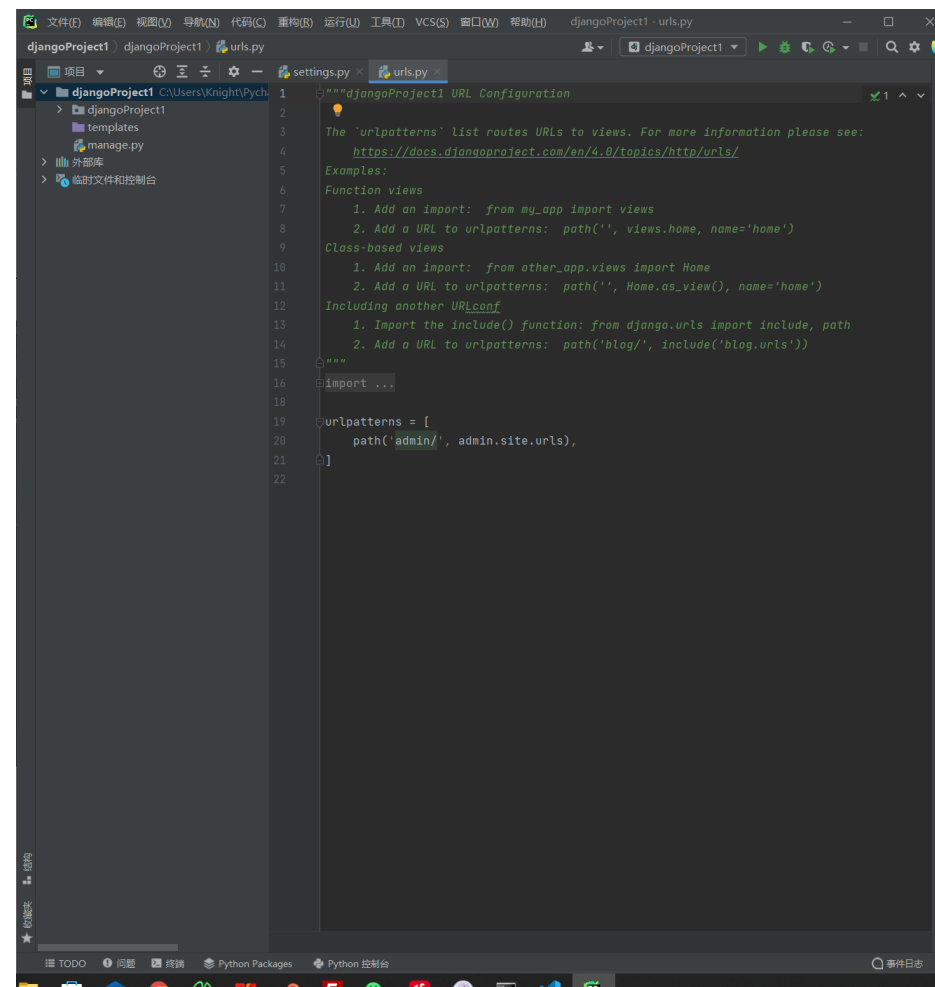
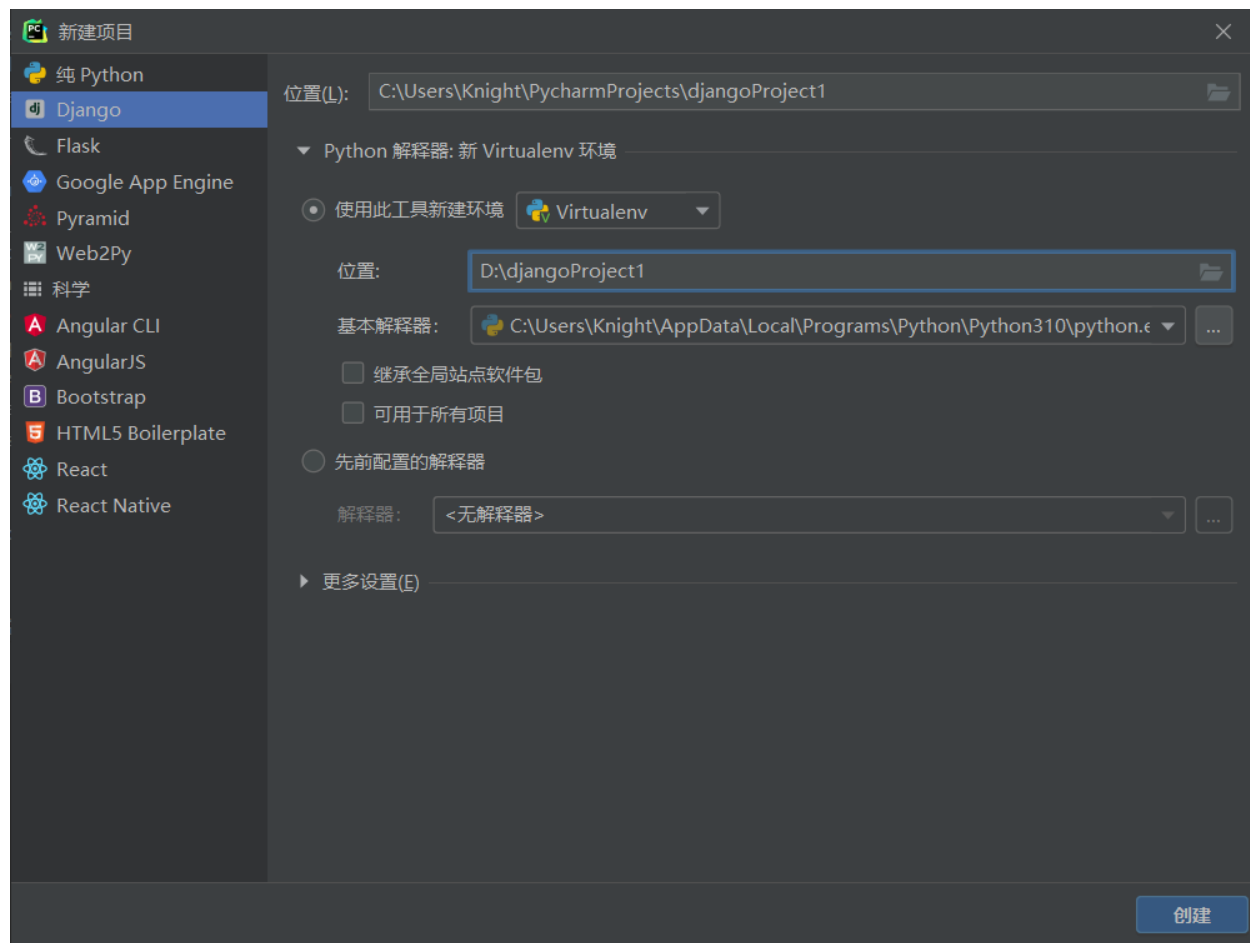
用vscode打开刚刚的项目文档



手把手带你从零开始建站

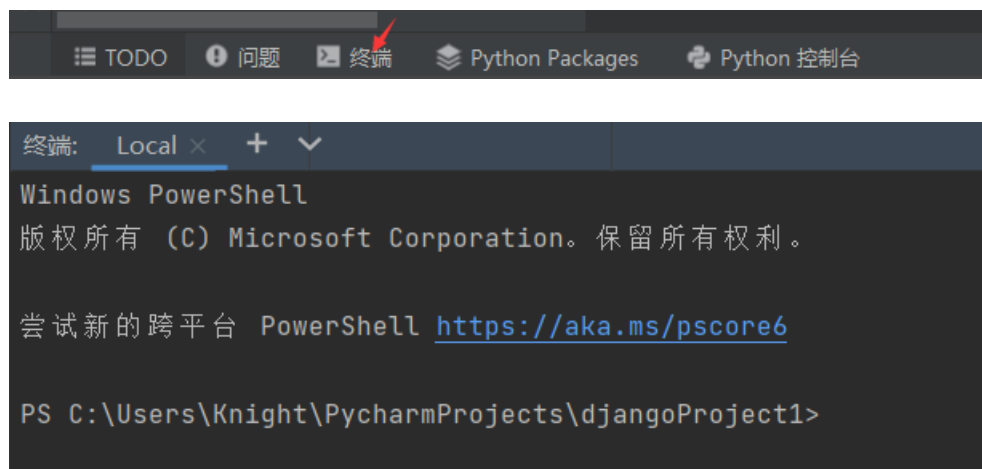
打开pycharm，新建一个项目

-后端



手把手带你从零开始建站

-后端

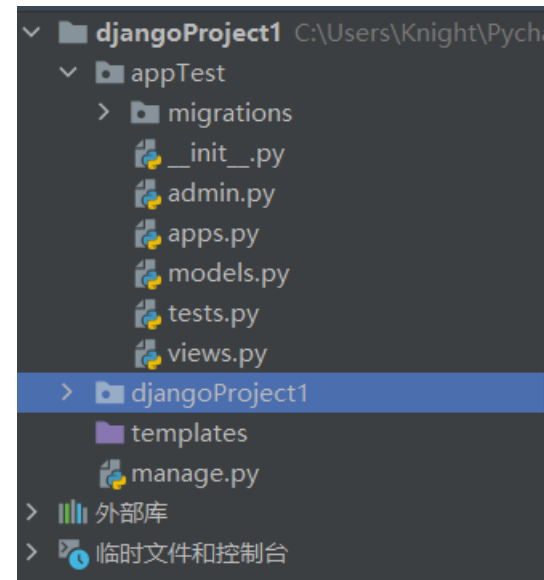
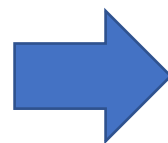


A screenshot of the PyCharm terminal window. The top bar shows tabs for 'TODO', '问题' (Issues), '终端' (Terminal) with a red checkmark, 'Python Packages', and 'Python 控制台' (Python Console). The terminal content shows a Windows PowerShell prompt at 'PS C:\Users\Knight\PycharmProjects\djangoProject1>'.

```
终端: Local x + v
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/powershell

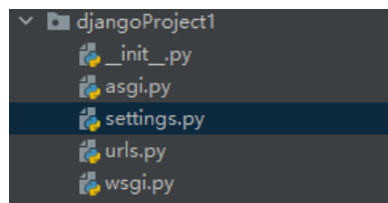
PS C:\Users\Knight\PycharmProjects\djangoProject1>
```



在自己项目目录下创建一个app `django-admin startapp XXX`

```
PS C:\Users\Knight\PycharmProjects\djangoProject1> django-admin startapp appTest
```

在settings.py中注册一下app，让项目知道我们创建了个APP



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

手把手带你从零开始建站

在终端中输入 `python manage.py runserver 127.0.0.1:3000`

-后端

```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.  
Run 'python manage.py migrate' to apply them.  
June 08, 2022 - 17:25:38  
Django version 4.0.5, using settings 'djangoProject1.settings'  
Starting development server at http://127.0.0.1:3000/  
Quit the server with CTRL-BREAK.  
^
```



django

View [release notes](#) for Django 4.0



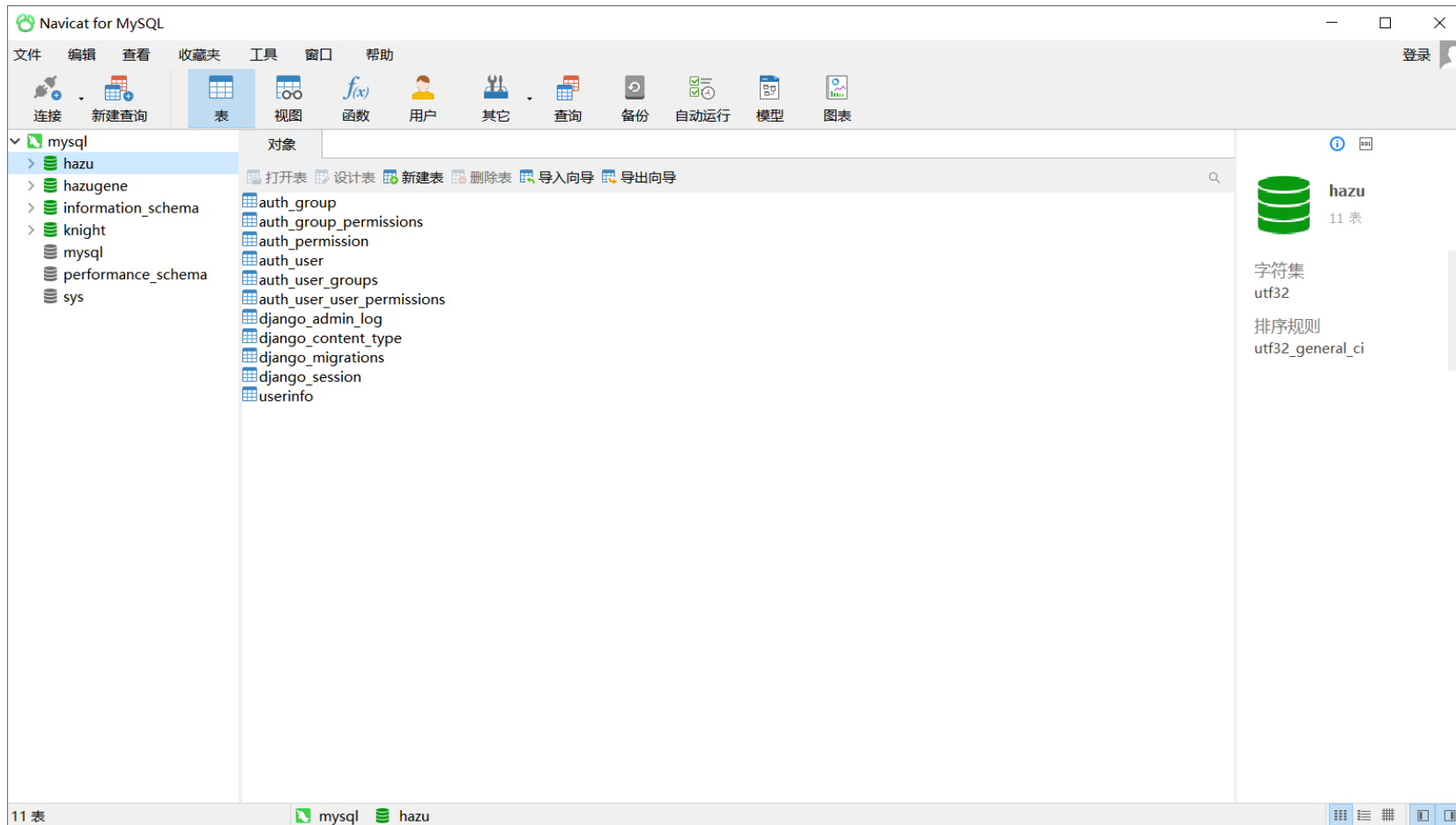
The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

手把手带你从零开始建站

-数据库

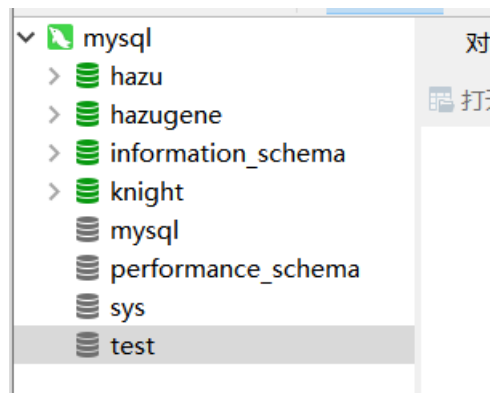
下载Navicat for MySQL <http://www.navicat.com.cn/download/navicat-for-mysql>
方便观察表以及数据库的情况，可以可视化操作



手把手带你从零开始建站

-数据库

新建一个数据库test

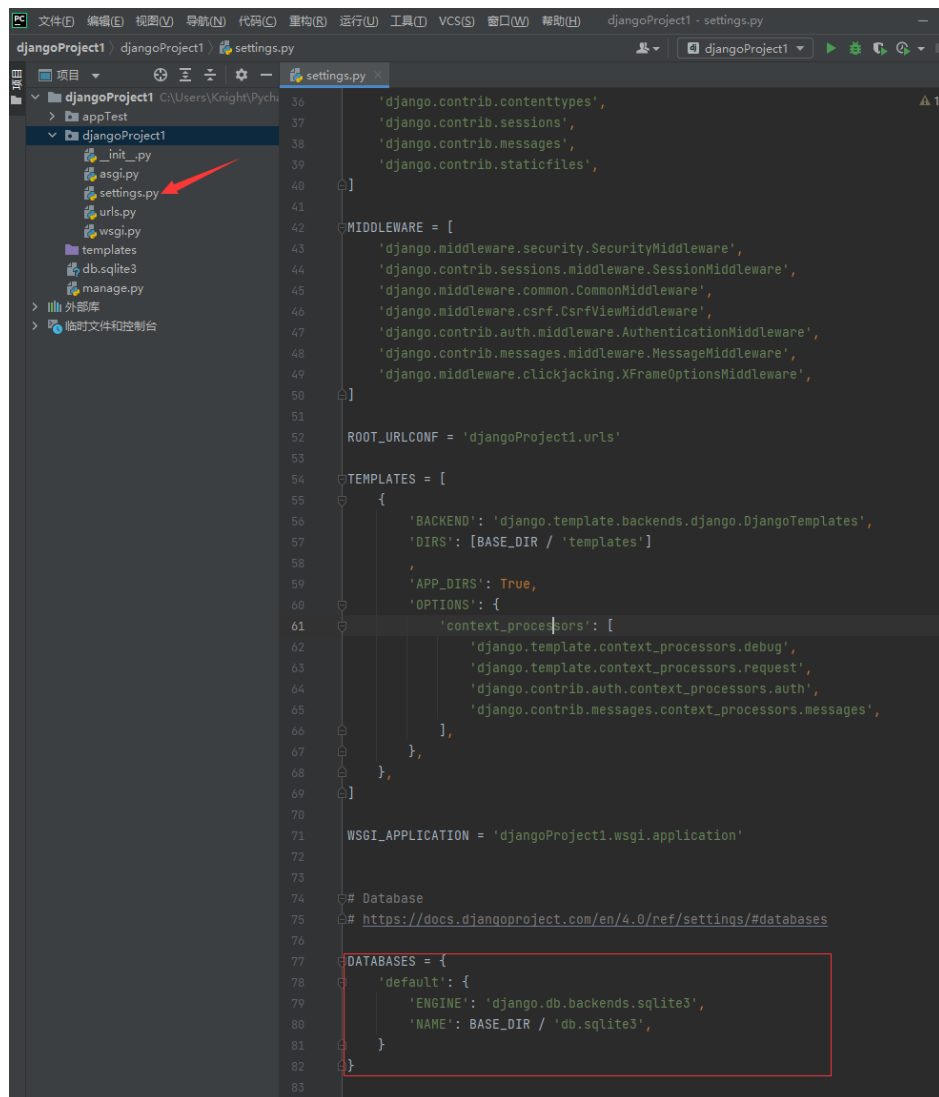


目前已经准备好了前端，后端以及数据库，开始连接前端和后端，后端和数据库

手把手带你从零开始建站

打开项目目录下的setting.py

-后端数据库联系



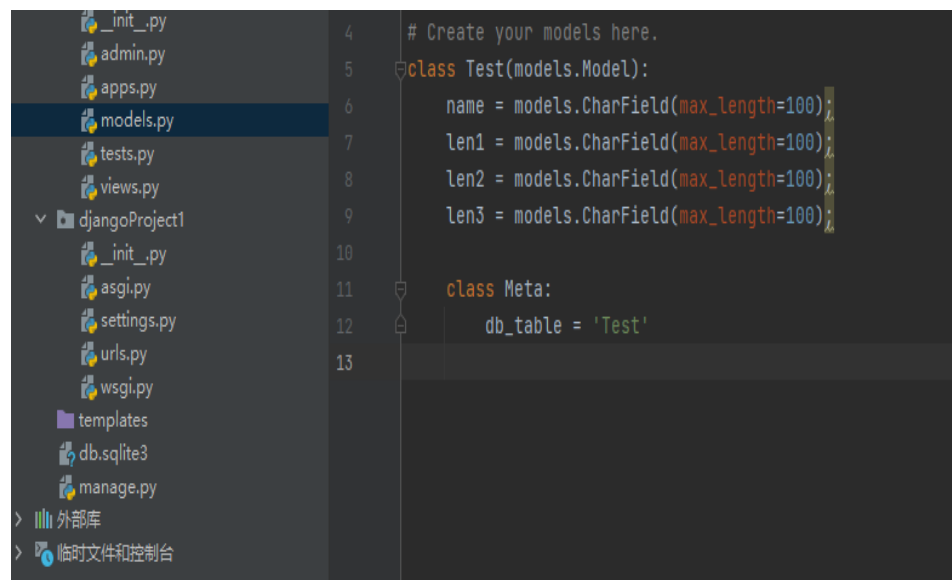
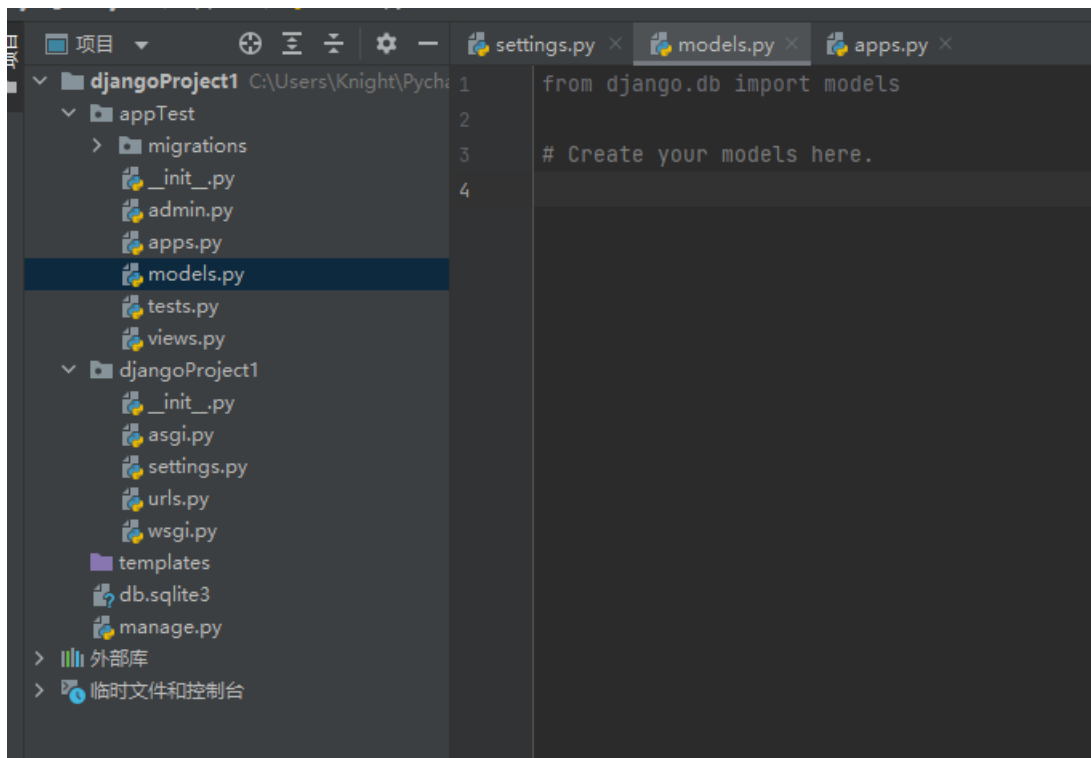
将原来的自带的sqlite3转换为mysql的配置
NAME是数据库名称

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'hazugene',
        'USER': 'root',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

手把手带你从零开始建站

-后端数据库联系

打开我们创建的APP下面的model.py，model.py存的就是数据库的所有表
在这里因为我目前的数据是只有名字和三个长度，就写了四个字段，可以根据此自由添加



手把手带你从零开始建站

-后端数据库联系

这时候我们没有安装mysql的client需要在终端地方按下ctrl+c结束运行，然后运行命令
pip install mysqlclient

```
PS C:\Users\Knight\PycharmProjects\djangoProject1> pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-2.1.0-cp310-cp310-win_amd64.whl (180 kB)
    |#####| 180 kB 652 kB/s
Installing collected packages: mysqlclient
Successfully installed mysqlclient-2.1.0
WARNING: You are using pip version 21.1.2; however, version 22.1.2 is available.
You should consider upgrading via the 'D:\djangoProject1\Scripts\python.exe -m pip install --upgrade pip' command.
```

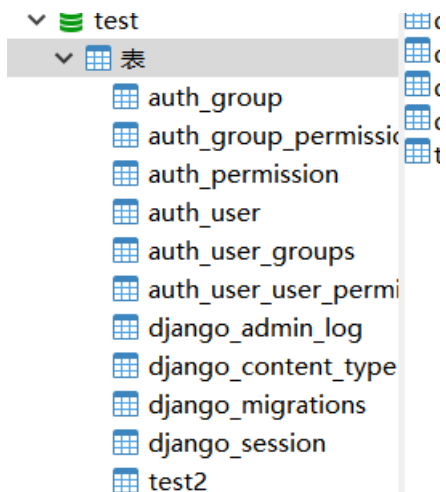
model.py需要跟数据库进行融合，所以执行两条命令
python manage.py makemigrations
python manage.py migrate

```
PS C:\Users\Knight\PycharmProjects\djangoProject1> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
```

手把手带你从零开始建站

-后端数据库联系

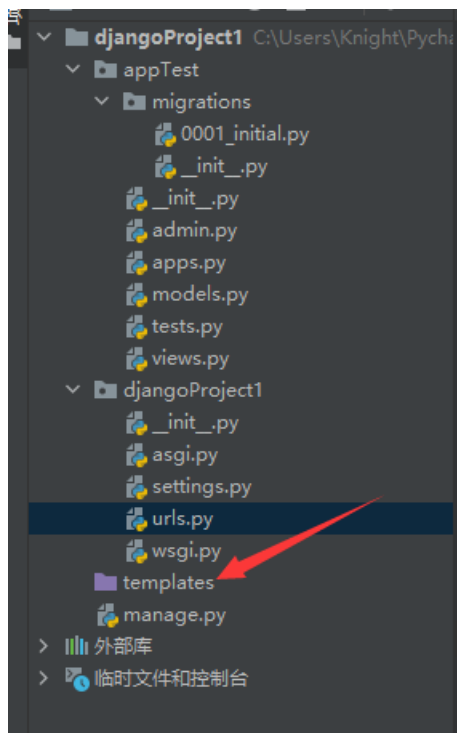
我们从Navicat for MySQL可以看到已经创建了很多表，创建我们刚刚model.py的test2，代表已经成功



手把手带你从零开始建站

-后端数据库联系

将我们的数据导入数据库，主要是excel文件导入数据库，将excel文件放到templates文件夹下在view中写下一个导入的方法，可以复制我的代码或者自己找个方法导入到数据库



```
from django.http import JsonResponse
from django.shortcuts import render

# Create your views here.

import xlrd

from appTest.models import Test2

def write(request):
    print(1)
    wb = xlrd.open_workbook(r".\templates\MH63RS3.xls")
    sht = wb.sheets()[0]
    for i in range(0, 1000):
        data = {}
        for j in range(0, 4):
            bb = sht.cell_value(i, j)
            if j==0:
                data["name"] = bb
            if j==1:
                data["len1"] = bb
            if j==2:
                data["len2"] = bb
            if j==3:
                data["len3"] = bb
        Test2(name=data["name"], len1=data["len1"], len2=data["len2"], len3=data["len3"])
    return JsonResponse({'code': 1, 'data': None})
```

手把手带你从零开始建站

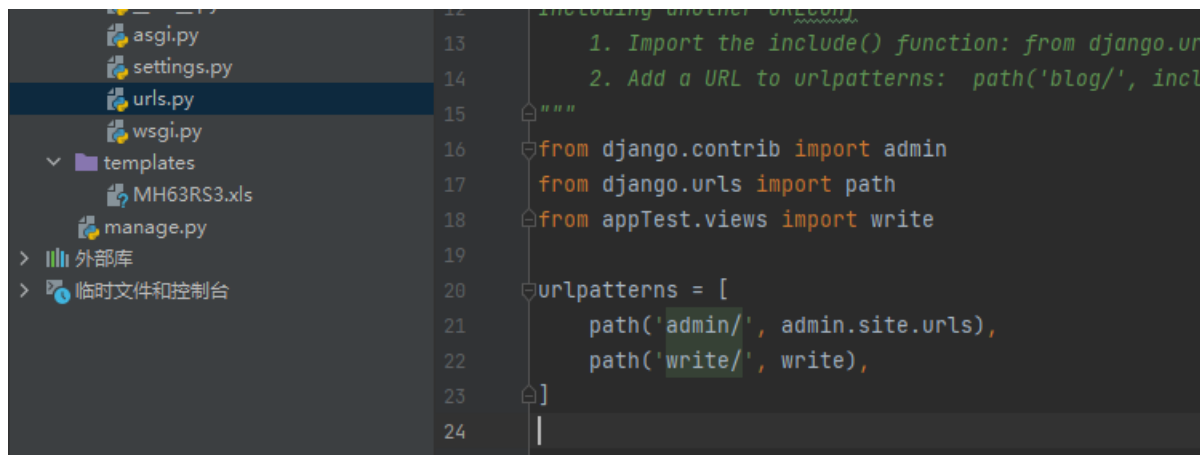
-后端数据库联系

```
import xlrd;
from appTest.models import Test2
def write(request):
    wb = xlrd.open_workbook(r".\templates\MH63RS3.xls")
    sht = wb.sheets()[0]
    for i in range(0, 1000):
        data = {}
        for j in range(0, 4):
            bb = sht.cell_value(i, j)
            if j==0:
                data["name"] = bb;
            if j==1:
                data["len1"] = bb;
            if j==2:
                data["len2"] = bb;
            if j==3:
                data["len3"] = bb;
        Test2(name=data["name"],len1=data["len1"],len2=data["len2"],len3=data["len3"]).save();
    return JsonResponse({'code': 1, 'data': None});
```

手把手带你从零开始建站

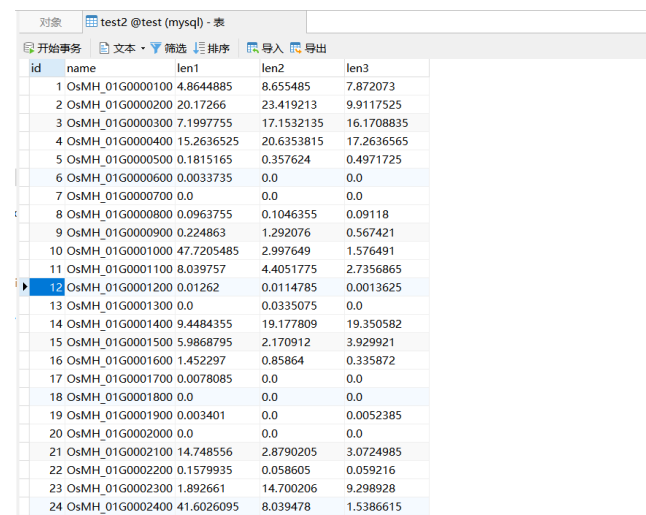
-后端数据库联系

定义了方法，需要去访问到找个方法，就要找到urls.py，添加对应的关系



```
12 including another URLconf
13     1. Import the include() function: from django.ur
14     2. Add a URL to urlpatterns: path('blog/', incl
15
16 from django.contrib import admin
17 from django.urls import path
18 from appTest.views import write
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('write/', write),
23 ]
24
```

这时候重新运行一下
访问<http://127.0.0.1:3000/write/>
就可以将数据导入进去，可以通过我们的sql可视化软件看到



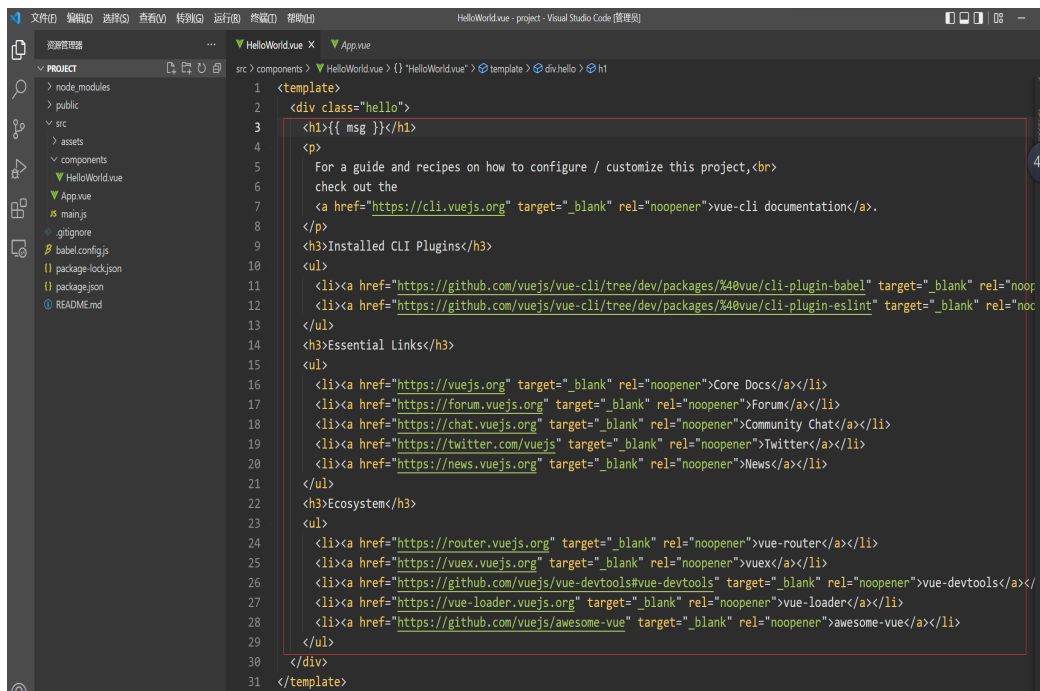
id	name	len1	len2	len3
1	OsMH_01G0000100	4.8644885	8.655485	7.872073
2	OsMH_01G0000200	20.17266	23.419213	9.9117525
3	OsMH_01G0000300	7.1997755	17.1532135	16.1708835
4	OsMH_01G0000400	15.2636525	20.6353815	17.2636565
5	OsMH_01G0000500	0.1815165	0.357624	0.4971725
6	OsMH_01G0000600	0.0033735	0.0	0.0
7	OsMH_01G0000700	0.0	0.0	0.0
8	OsMH_01G0000800	0.0963755	0.1046355	0.09118
9	OsMH_01G0000900	0.224863	1.292076	0.567421
10	OsMH_01G0001000	47.7205485	2.997649	1.576491
11	OsMH_01G0001100	8.039757	4.4051775	2.7356865
12	OsMH_01G0001200	0.01262	0.0114785	0.0013625
13	OsMH_01G0001300	0.0	0.0335075	0.0
14	OsMH_01G0001400	9.4484355	19.177809	19.350582
15	OsMH_01G0001500	5.9868795	2.170912	3.929921
16	OsMH_01G0001600	1.452297	0.85864	0.335872
17	OsMH_01G0001700	0.0078085	0.0	0.0
18	OsMH_01G0001800	0.0	0.0	0.0
19	OsMH_01G0001900	0.003401	0.0	0.0052385
20	OsMH_01G0002000	0.0	0.0	0.0
21	OsMH_01G0002100	14.748556	2.8790205	3.0724985
22	OsMH_01G0002200	0.1579935	0.058605	0.059216
23	OsMH_01G0002300	1.892661	14.700206	9.298928
24	OsMH_01G0002400	41.6026095	8.039478	1.5386615

手把手带你从零开始建站

- 前端后端联系

我们最终实现的是一个查询的功能，所以需要在前端写一个查询框，而且需要进行前后端的交互，前端需要将数据提交到后端，然后后端查询后传入到前端

将文中红色框内的内容删除，增加dataTest, select和查询的前端按钮



```
1 <template>
2   <div class="hello">
3     <h1>{{ msg }}</h1>
4     <p>
5       For a guide and recipes on how to configure / customize this project,<br>
6       check out the
7       <a href="https://cli.vuejs.org" target="_blank" rel="noopener">vue-cli documentation</a>.<br>
8     </p>
9     <h3>Installed CLI Plugins</h3>
10    <ul>
11      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-babel" target="_blank" rel="noopener">https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-babel</a>.</li>
12      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-eslint" target="_blank" rel="noopener">https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-eslint</a>.</li>
13    </ul>
14    <h3>Essential Links</h3>
15    <ul>
16      <li><a href="https://vuejs.org" target="_blank" rel="noopener">Core Docs</a></li>
17      <li><a href="https://forum.vuejs.org" target="_blank" rel="noopener">Forum</a></li>
18      <li><a href="https://chat.vuejs.org" target="_blank" rel="noopener">Community Chat</a></li>
19      <li><a href="https://twitter.com/vuejs" target="_blank" rel="noopener">Twitter</a></li>
20      <li><a href="https://news.vuejs.org" target="_blank" rel="noopener">News</a></li>
21    </ul>
22    <h3>Ecosystem</h3>
23    <ul>
24      <li><a href="https://router.vuejs.org" target="_blank" rel="noopener">vue-router</a></li>
25      <li><a href="https://vuex.vuejs.org" target="_blank" rel="noopener">vuex</a></li>
26      <li><a href="https://github.com/vuejs/vue-devtools#vue-devtools" target="_blank" rel="noopener">vue-devtools</a></li>
27      <li><a href="https://vue-loader.vuejs.org" target="_blank" rel="noopener">vue-loader</a></li>
28      <li><a href="https://github.com/vuejs/awesome-vue" target="_blank" rel="noopener">awesome-vue</a></li>
29    </ul>
30  </div>
31 </template>
```

```
<template>
  <div class="hello">
    <input type="text" id="answer"><button @click="select">查询</button>
    <textarea id="result"></textarea>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  },
  data() {
    return {
      dataTest: {},
    }
  },
  methods: {
    select()
    {
    }
  },
}
```

手把手带你从零开始建站

- 前端后端联系

我们需要理清查询的逻辑，通过之前的请求流程，首先是点击查询按钮，获取到输入框中的内容，将内容传输到后端，后端将内容进行查询，获取的数据返回到前端

写select方法中的逻辑，会用到前后端交互的技术(ajax)，所以需要引入ajax的包，jquery

在终端的地方先安装jquery `npm install jquery --save`

然后在main.js中引入

```
Vue.prototype.$ = jquery;
```

```
Vue.config.productionTip = false
```

```
3  import jquery from "jquery";  
4  Vue.prototype.$ = jquery;
```

在你需要用的文件中引入，我们现在用的就是HelloWorld.vue，在其中引用

```
<script>  
import $ from 'jquery'  
export default {
```

手把手带你从零开始建站

第一步就是写select方法的中的逻辑

- 前端后端联系

```
select()  
{  
  let value = document.getElementById("answer").value;  
  console.log(value);  
  let that = this;  
  $.ajax({  
    url: '/api/select/',  
    type: 'POST',  
    data: value,  
    success: function(msg){  
      that.dataTest = msg['data'][0];  
      console.log(msg);  
    },  
    error: function(msg){  
      console.log(msg);  
    }  
  });  
}
```

手把手带你从零开始建站

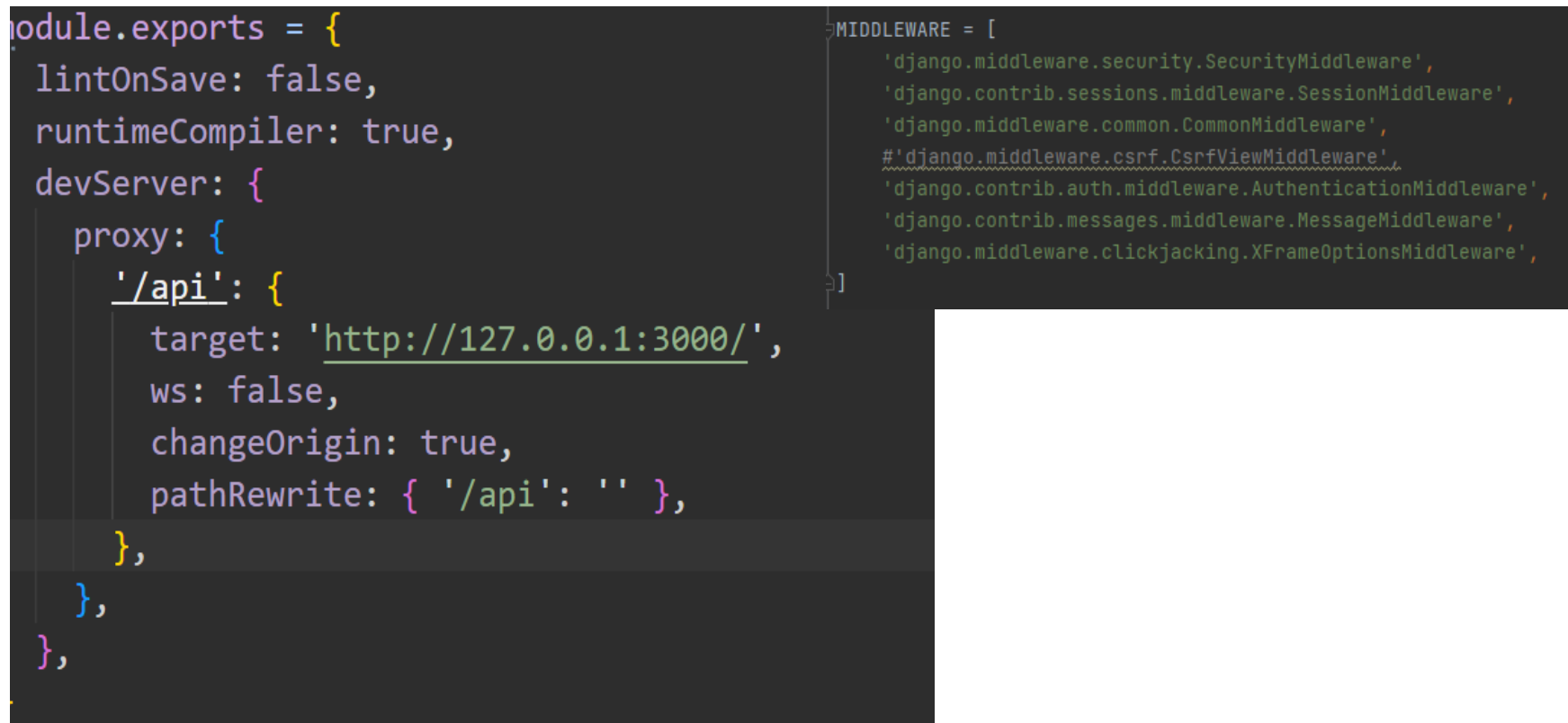
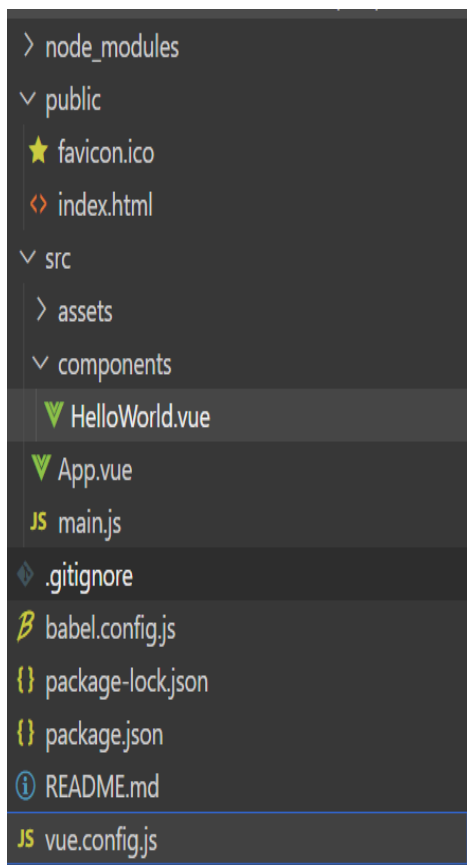
- 前端后端联系

```
select()
{
  let value = document.getElementById("answer").value;
  console.log(value);
  let that = this;
  $.ajax({
    url: '/api/select/',
    type: 'POST',
    data: value,
    success: function(msg){
      that.dataTest = msg['data'][0];
      console.log(msg);
    },
    error: function(msg){
      console.log(msg);
    }
  });
}
```

手把手带你从零开始建站

- 前端后端联系

第二步需要处理前后端跨域的问题，目前有的解决方案有很多，比如jsoup，中间件以及代理，我们采用的方法是代理，在vue目录下创建一个vue.config.js，配置一下跨域的代理，并且注释掉django中settings.py中的
'django.middleware.csrf.CsrfViewMiddleware'



手把手带你从零开始建站

- 前端后端联系

并且在django中setting.py中加入下面的代码

```
CSRF_TRUSTED_ORIGINS = [  
    'http://localhost:8080'  
]  
ALLOWED_HOSTS = [  
    '127.0.0.1',  
    'localhost'  
]  
CORS_ORIGIN_WHITELIST = [  
    'http://localhost:8080',  
]
```

```
SECRET_KEY = 'django-insecure-q_yls-09ihm4p!jj1n_)hm)xpgtzbm0b0^*b+-onlo9w1g#=0y'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
CSRF_TRUSTED_ORIGINS = [  
    'http://localhost:8080'  
]  
ALLOWED_HOSTS = [  
    '127.0.0.1',  
    'localhost'  
]  
CORS_ORIGIN_WHITELIST = [  
    'http://localhost:8080',  
]  
|  
  
# Application definition  
  
INSTALLED_APPS = [  
    'django.contrib.admin',  
]
```

手把手带你从零开始建站

- 前端后端联系

```
module.exports = {  
  lintOnSave: false,  
  runtimeCompiler: true,  
  devServer: {  
    proxy: {  
      '/api': {  
        target: 'http://127.0.0.1:3000/',  
        ws: false,  
        changeOrigin: true,  
        pathRewrite: { '/api': '' },  
      },  
    },  
  },  
}
```

手把手带你从零开始建站

- 前端后端联系

第二步就是写后端的方法，先去view.py写方法，然后再去url.py添加对应关系

```
def select(request):  
    data = request.body.decode("utf-8");  
    print(data)  
    callbackData = Test2.objects.filter(name=data).values();  
    callbackData = list(callbackData)  
    callbackData[0]["id"] = str(callbackData[0]["id"]);  
    print(list(callbackData))  
    return JsonResponse({'code': 1, 'data': callbackData});
```

```
from django.contrib import admin  
from django.urls import path  
from appTest.views import write, select  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('write/', write),  
    path('select/', select)  
]
```

```
def select(request):  
    data = request.body.decode("utf-8");  
    print(data)  
    callbackData = Test2.objects.filter(name=data).values();  
    callbackData = list(callbackData)  
    callbackData[0]["id"] = str(callbackData[0]["id"]);  
    print(list(callbackData))  
    return JsonResponse({'code': 1, 'data': callbackData});
```

手把手带你从零开始建站

- 前端后端联系

这时候可以打开浏览器看一下，返回了获取的信息，并且赋值给了dataTest

接下来就是将数据显示到前端

```
<input type="text" id="answer"> <button @click="select">查询</button>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<div id="text">Name: {{dataTest.name}}
  <br>
  len1: {{dataTest.len1}}
  <br>
  len2: {{dataTest.len2}}
  <br>
  len3: {{dataTest.len3}}
</div>
</div>
```

0sMH_01G0002400

```
▼ {code: 1, data: Array(1)} ⓘ
  code: 1
  data: Array(1)
    ▼ 0:
      id: (...)
      len1: (...)
      len2: (...)
      len3: (...)
      name: (...)
      __ob__: Observer {value: {...}, dep: Dep, vmCount: 0}
      get id: f reactiveGetter()
      set id: f reactiveSetter(newVal)
      get len1: f reactiveGetter()
      set len1: f reactiveSetter(newVal)
      get len2: f reactiveGetter()
      set len2: f reactiveSetter(newVal)
      get len3: f reactiveGetter()
      set len3: f reactiveSetter(newVal)
      get name: f reactiveGetter()
      set name: f reactiveSetter(newVal)
      [[Prototype]]: Object
      length: 1
    [[Prototype]]: Array(0)
  [[Prototype]]: Object
```

手把手带你从零开始建站

- 前端后端联系

最终HelloWorld.vue的代码如下

```
<template>
  <div class="hello">
    <input type="text" id="answer"> <button @click="select">查询</button>
    <br>
    <br>
    <br>

    <br>
    <br>
    <br>
    <div id="text">Name: {{dataTest.name}}
      <br>
      len1: {{dataTest.len1}}
      <br>
      len2: {{dataTest.len2}}
      <br>
      len3: {{dataTest.len3}}

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

手把手带你从零开始建站

最终查询效果



OsMH_01G0002400

查询

Gene Name: OsMH_01G0002400

Gene Expression (FPKM) in Leaf: 41.6026095

Gene Expression (FPKM) in Root: 8.039478

Gene Expression (FPKM) in Panicle: 1.5386615

前端还有很多UI组件帮助我们开发，比如



Ant Design Vue



ECHARTS



Element

Bootstrap

简洁、直观、强悍的前端开发框架，让web开发更迅速、简单。

[Bootstrap3中文文档\(v3.4.1\)](#)

手把手带你从零开始建站

-番外（数据可视化）

首先引入echarts，与jquery的引入类似，首先在终端安装echarts

```
npm install echarts --save
```

在vue中引入（全局引入） 再局部引入

```
import Vue from 'vue'
import App from './App.vue'
import jquery from "jquery";
Vue.prototype.$ = jquery;
Vue.config.productionTip = false
import * as echarts from 'echarts';
Vue.prototype.$echarts = echarts;

new Vue({
  render: h => h(App),
}).$mount('#app')
```

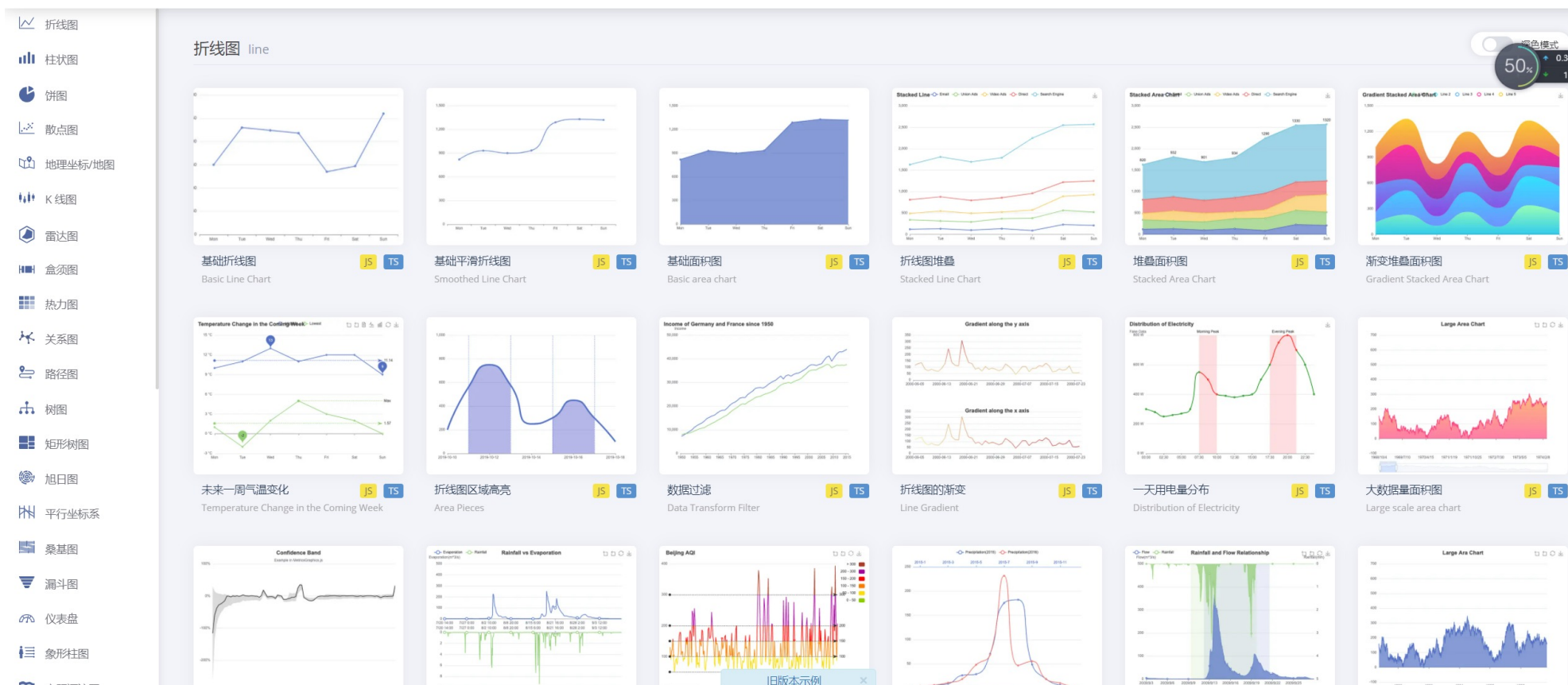
```
<script>
import $ from 'jquery'
import * as echarts from 'echarts';

export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
```

手把手带你从零开始建站

-番外（数据可视化）

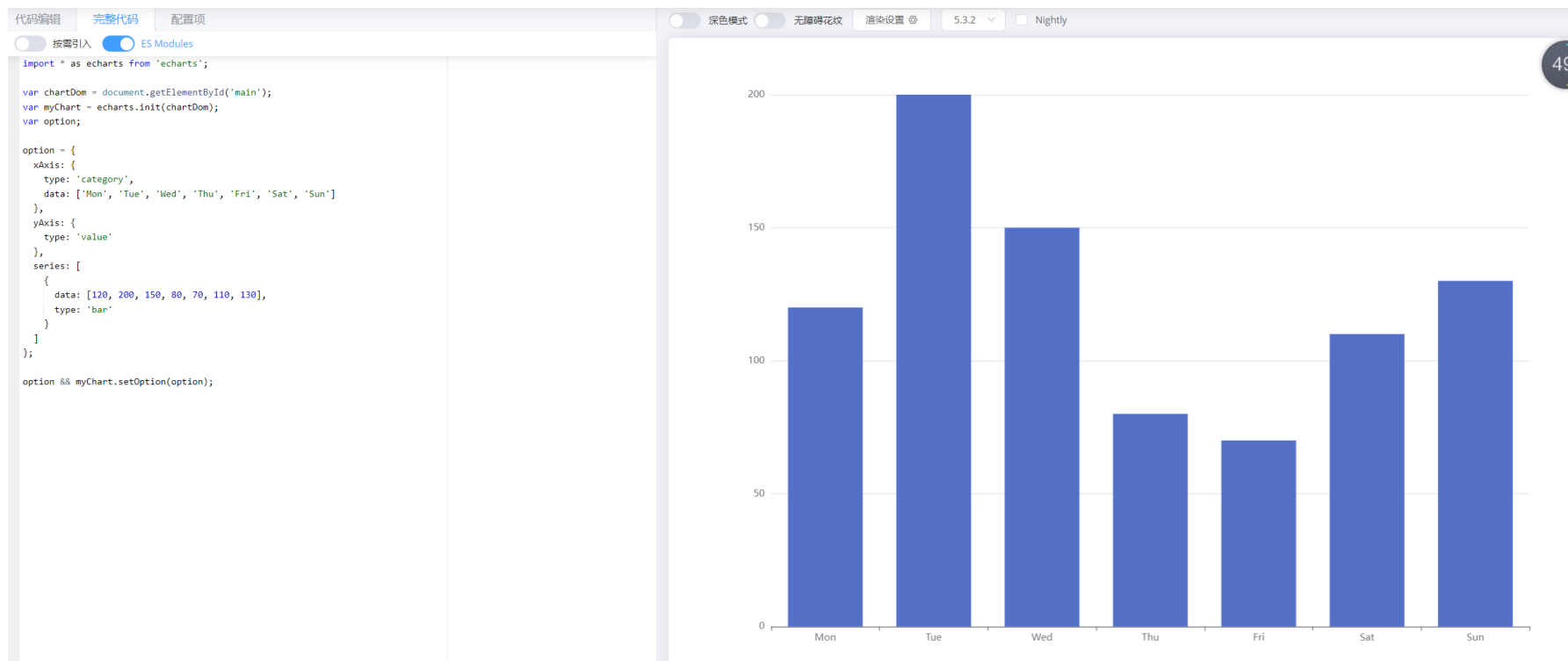
打开echarts官网 <https://echarts.apache.org/examples/zh/>



手把手带你从零开始建站

-番外（数据可视化）

选择一个你想要的可视化的图表，并且点击完整代码



手把手带你从零开始建站

-番外（数据可视化）

在前端中的methods中写一个方法为show，准备可视化的数据在data定义一个chartData，并且在前端中写一个id为main的div，定义一下宽度和高度

```
data() {  
  return {  
    dataTest:{},  
    chartData:[],  
  }  
},
```

```
methods: {
```

```
  show(){
```

```
  },
```

```
  select()
```

```
  {
```

```
    let value = document.getElementById("answer").value;  
    console.log(value);
```

```
<div class="hello">  
  <input type="text" id="answer"> <button @click="select">查询</button>  
  <br>  
  <br>  
  <br>  
  <br>  
  <br>  
  <div id="text">Gene Name: {{dataTest.name}}  
    <br>  
    Gene Expression (FPKM) in Leaf: {{dataTest.len1}}  
    <br>  
    Gene Expression (FPKM) in Root: {{dataTest.len2}}  
    <br>  
    Gene Expression (FPKM) in Panicle: {{dataTest.len3}}  
  </div>  
  <div id="main" style="width:50%;height:376px;margin-left: 30%;"></div>  
</div>
```

手把手带你从零开始建站

-番外（数据可视化）

将echarts中的代码复制到show中，并且将data改成自己的数据

```
show(){  
  var chartDom = document.getElementById('main');  
  var myChart = echarts.init(chartDom);  
  var option;  
  
  option = {  
    xAxis: {  
      type: 'category',  
      data: ['Gene Expression (FPKM) in Leaf', 'Gene Expression (FPKM) in Root', 'Gene Expression (FPKM) in Panicle']  
    },  
    yAxis: {  
      type: 'value'  
    },  
    series: [  
      {  
        data: this.chartData,  
        type: 'bar'  
      }  
    ]  
  };  
}
```

手把手带你从零开始建站

-番外（数据可视化）

```
show(){
  var chartDom = document.getElementById('main');
  var myChart = echarts.init(chartDom);
  var option;

  option = {
    xAxis: {
      type: 'category',
      data: ['Gene Expression (FPKM) in Leaf', 'Gene Expression (FPKM) in Root', 'Gene Expression (FPKM) in Panicle'],
    },
    yAxis: {
      type: 'value'
    },
    series: [
      {
        data: this.chartData,
        type: 'bar'
      }
    ]
  };
}
```

手把手带你从零开始建站

-番外（数据可视化）

在success成功的回调中将chartData数据改成需要可视化的数据，并且调用show方法

```
success:function(msg){  
    that.dataTest = msg['data'][0];  
    that.chartData.push(parseFloat(that.dataTest.len1))  
    that.chartData.push(parseFloat(that.dataTest.len2))  
    that.chartData.push(parseFloat(that.dataTest.len3))  
    that.show();  
    console.log(msg);  
},
```

手把手带你从零开始建站

-番外（数据可视化）

最终可视化的效果图

