### Web框架之Django框架开发

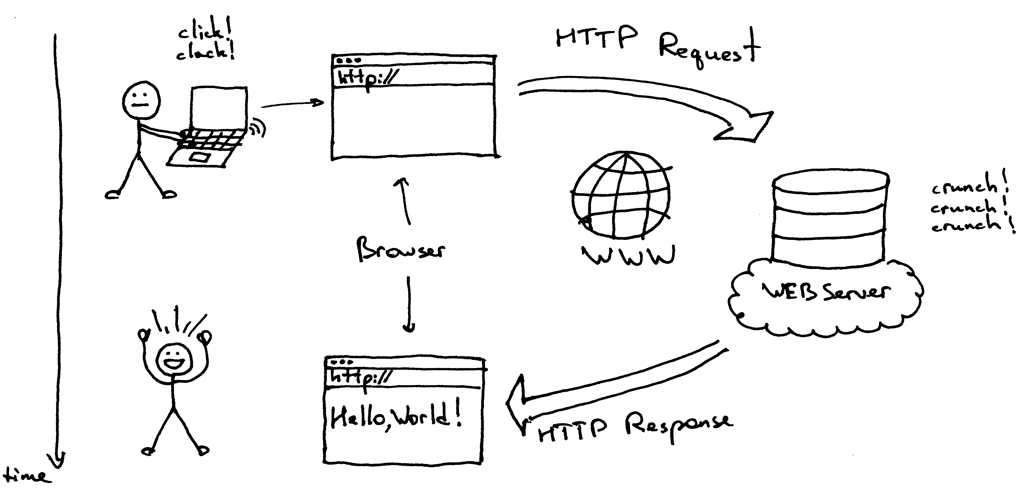
1. 什么是web服务器(了解)

当我们在浏览器输入URL后，浏览器会先请求[DNS服务器，获得请求站点的 IP 地址](http://blog.jobbole.com/94132/" \t "http://python.jobbole.com/85874/_blank)。然后发送一个HTTP Request（请求）给拥有该 IP 的主机，接着就会接收到服务器给我们的 HTTP Response（响应），浏览器经过渲染后，以一种较好的效果呈现给我们。

这个过程中，正是Web服务器在幕后默默做贡献，它永久地等待客户端（主要是浏览器，比如Chrome，Firefox等）发送请求。

当收到请求之后，它会生成相应的响应并将其返回至客户端。

Web服务器通过HTTP协议与客户端通信，因此也被称为HTTP服务器。



1. Web服务器工作原理

可分成如下4个步骤

1. 建立连接：客户机通过TCP/IP协议建立到服务器的TCP连接
2. 请求过程：客户端向服务器发送HTTP协议请求包，请求服务器里的资源文档。
3. 应答过程：服务器向客户机发送HTTP协议应答包，如果请求的资源包含有动态语言的内容，那么服务器会调用动态语言的解释引擎负责处理“动态内容”，并将处理得到的数据返回给客户端。由客户端解释HTML文档，在客户端屏幕上渲染图形结果。
4. 关闭连接：客户机与服务器断开。

http 超文本传输协议 www域名都必须遵守这个协议

https 安全 套接字协议

古代的时候:飞鸽传书

庞广达->寂寞难耐->想认识一个朋友,聊聊天

飞鸽传书的地址 http:// 北京市昌平区(门牌号)/天通苑/东厢房/人物：任君爽

找任君爽 有几种可能

1. 查无此人404

403 forbidden

1. 找到人了

人不来

人来了 互相学习XXOO 200

1. 鸽子累死了
2. 任君爽不舒服 500服务器问题

例如：我们实现一个简单的 Web 服务器，监听本地端口 8000，在浏览器访问 http://localhost:8000 就能看到响应内容。而我们的程序也能够打印出客户端发来的请求内容



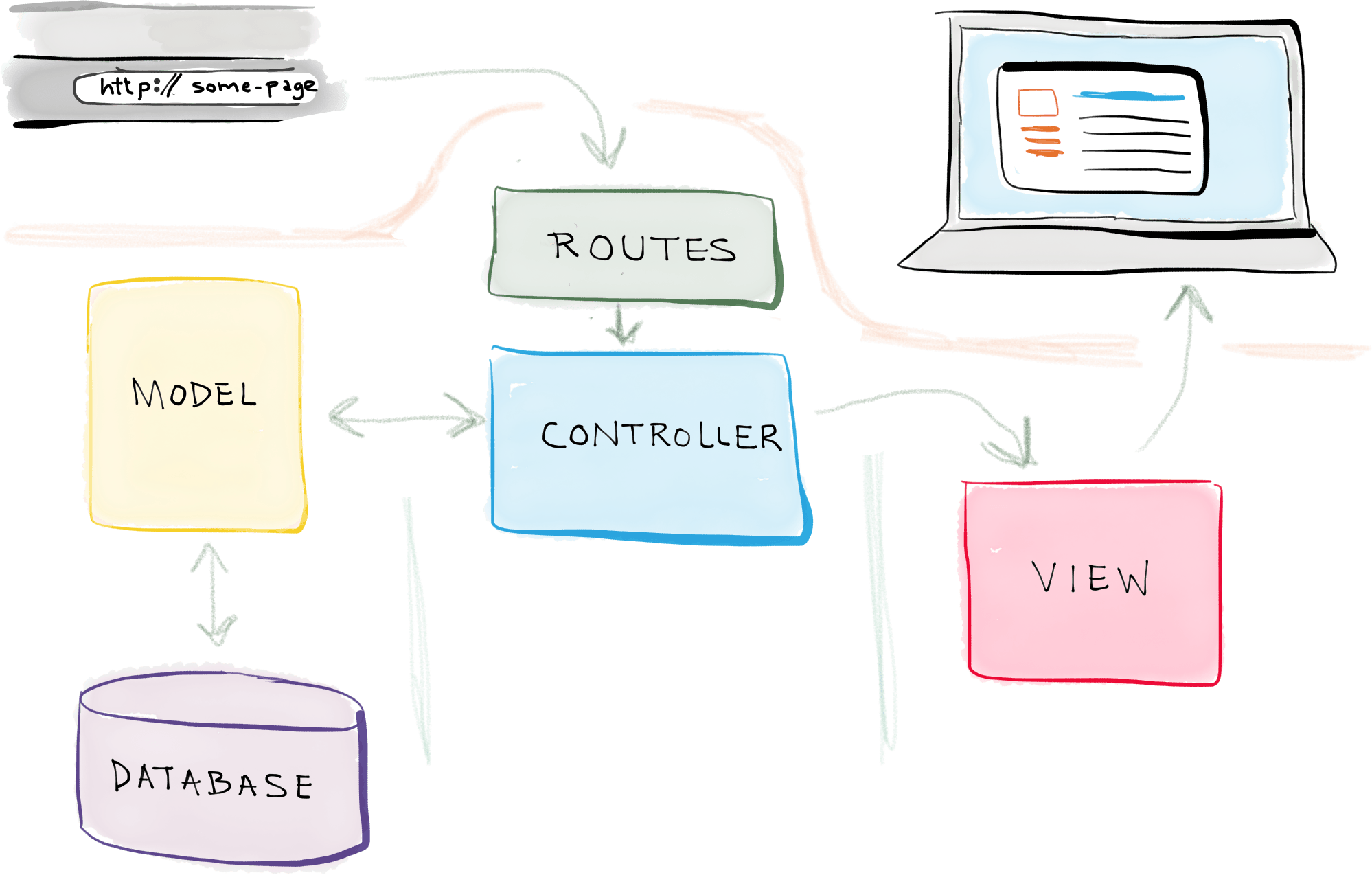
1. 什么是web框架(了解)

框架(framework)，特指为解决一个开放性问题而设计的具有一定约束性的支撑结构，使用框架可以帮你快速开发特定的系统，简单地说，就是你用别人搭建好的舞台来做表演。

对于所有的Web应用，本质上其实就是一个socket服务端，用户的浏览器其实就是一个socket客户端。

早期开发站点确做了许多重复性劳动，后来为了减少重复，避免写出庞杂，混乱的代码，人们将 Web 开发的关键性过程提取出来，开发出了各种 Web 框架。

有了框架，就可以专注于编写清晰、易维护的代码，无需关心数据库连接之类的重复性工作。



用户输入 URL，客户端发送请求，控制器（Controller）首先会拿到请求，然后用模型（Models）从数据库取出所有需要的数据，进行必要的处理，将处理后的结果发送给 视图（View），视图利用获取到的数据，进行渲染生成 HtmlResponse返回给客户端。

1. Python web 目前较火的三大框架(了解)
2. **Django**

Django是一个开放源代码的Web应用框架，由Python写成。采用了MTV的框架模式，即模型M，模板T和视图V。

它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的，即是CMS（内容管理系统）软件。

1. **Flask**

Flask是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ，模板引擎则使用 Jinja2 。Flask使用 BSD 授权。

 Flask也被称为 “microframework” ，因为它使用简单的核心，用 extension 增加其他功能。Flask没有默认使用的数据库、窗体验证工具。

 Flask 很轻，花很少的成本就能够开发一个简单的网站。非常适合初学者学习。Flask 框架学会以后，可以考虑学习插件的使用。例如使用 WTForm + Flask-WTForm 来验证表单数据，用 SQLAlchemy + Flask-SQLAlchemy 来对你的数据库进行控制。

1. **Tornado**

 Tornado是一种 Web 服务器软件的开源版本。Tornado 和现在的主流 Web 服务器框架（包括大多数 Python 的框架）有着明显的区别：它是非阻塞式服务器，而且速度相当快。

 得利于其 非阻塞的方式和对epoll的运用，Tornado 每秒可以处理数以千计的连接，因此 Tornado 是实时 Web 服务

的一个 理想框架。

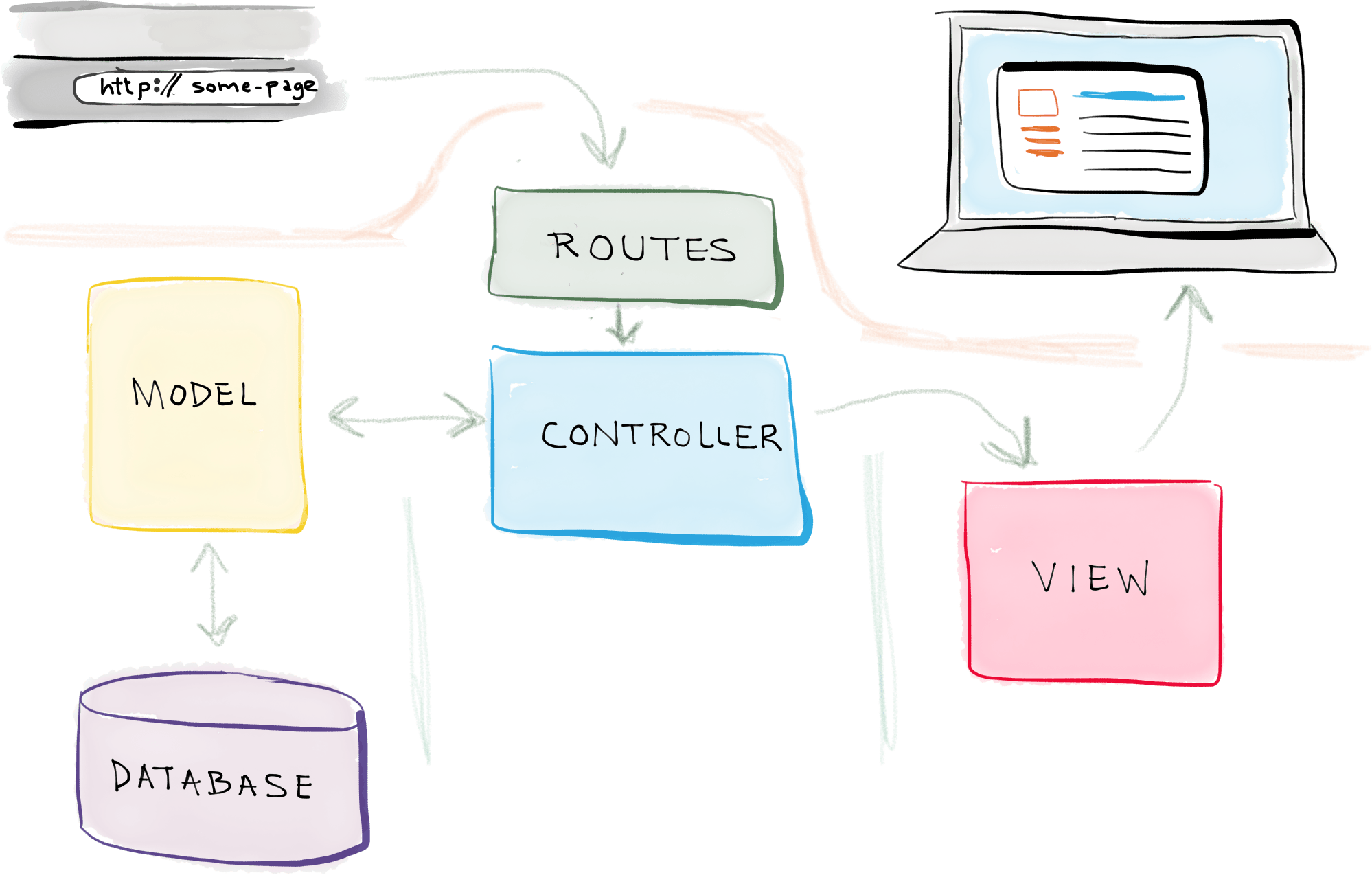
1. MVC/MTV介绍
2. MVC：**全名Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码**，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

　　通俗解释：一种文件的组织和管理形式！不要被缩写吓到了，这其实就是把不同类型的文件放到不同的目录下的一种方法，然后取了个高大上的名字。当然，它带来的好处有很多，比如前后端分离。

**模型(model)：定义数据库相关的内容，一般放在models.py文件中。**

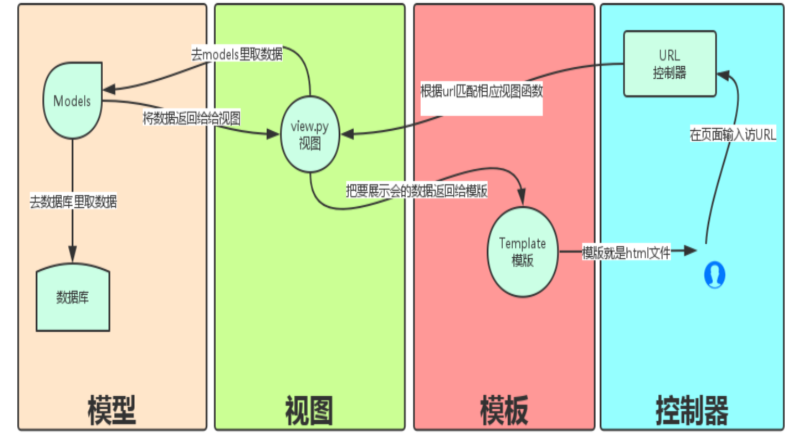
**视图(view)：定义HTML等静态网页文件相关，也就是那些html、css、js等前端的东西。**

**控制器(controller)：定义业务逻辑相关，就是你的主要代码。**

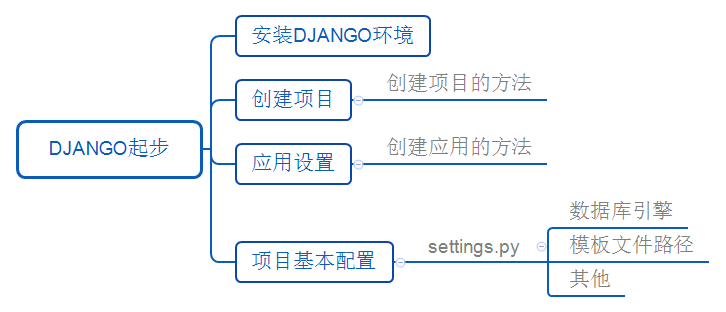


1. MTV: view不再是HTML相关，而是主业务逻辑了

html被放在Templates中，称作模板，于是MVC就变成了MTV。

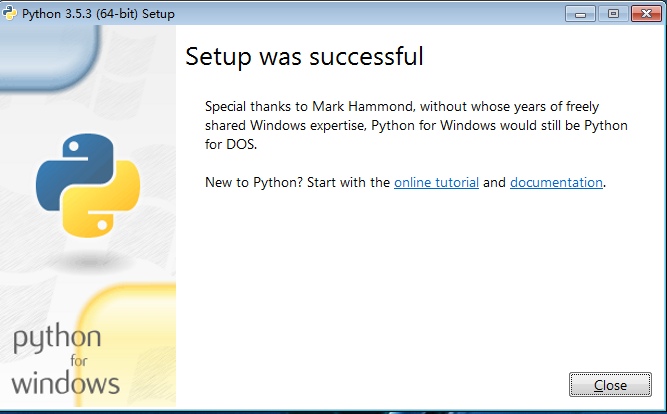
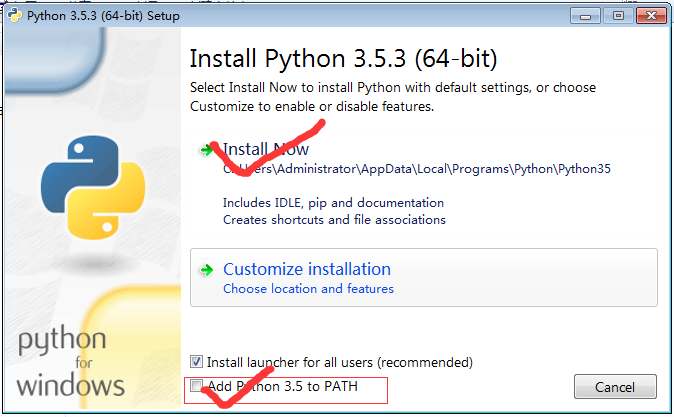


1. Django框架起步**(重点)**

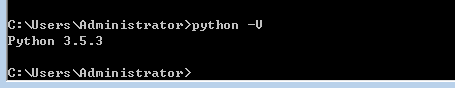


1. 安装Django环境
2. 配置虚拟环境

安装Python-3.5.3 amd64位

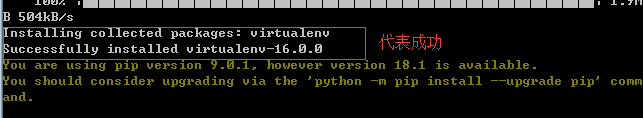


***Win+R小黑窗键入cmd命令后执行以下命令：*** python -V

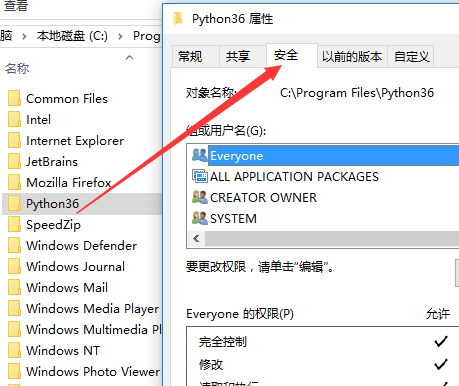


***Win+R小黑窗键入cmd命令后执行以下命令：***

命令1：pip install virtualenv

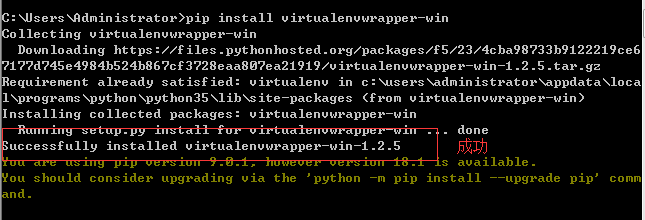


如果python安装在默认的C盘目录，有可能因为文件夹的权限不够导致安装失败，解决方案：将python文件夹给足权限



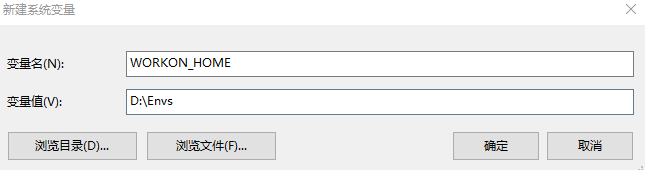
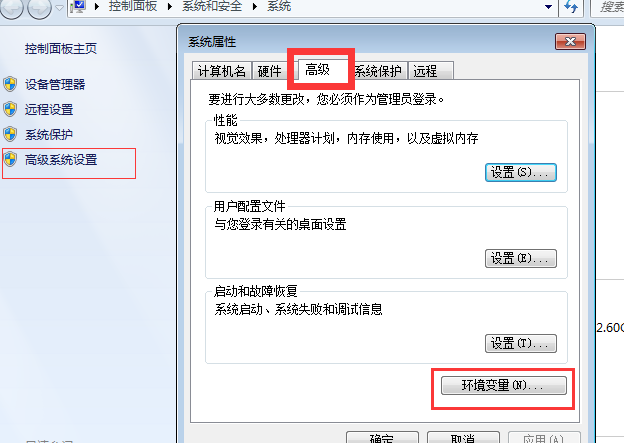
1. 安装virtualenv的扩展管理包，用于更方便的管理虚拟环境

命令2：pip install virtualenvwrapper-win

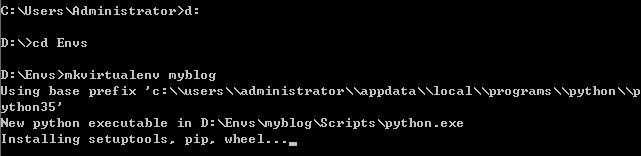


1. 设置系统环境变量,变量值是装载虚拟环境的目录，建议选择除C盘外的盘符

我的电脑->右键->属性->高级系统设置->高级->环境变量

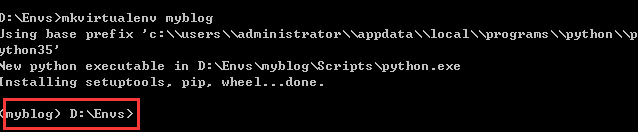


1. 虚拟目录的操作
2. 创建并进入到D:\Envs



1. 创建虚拟目录

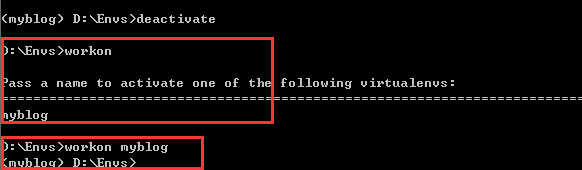
命令3：mkvirtualenv 虚拟环境名称



1. 进入虚拟目录

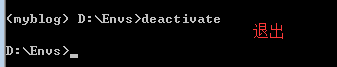
命令4：workon 虚拟目录名称

workon 查看全部的虚拟目录



1. 退出虚拟目录

deactivate

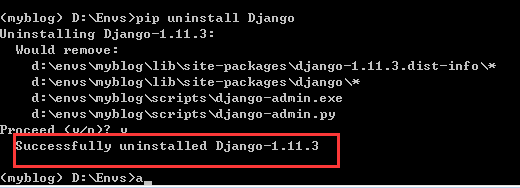


1. 安装以及运行Django环境

命令：pip install Django==1.11.3



卸载：pip uninstall Django



1. 创建项目以及应用的方法

方法一：命令行方法创建

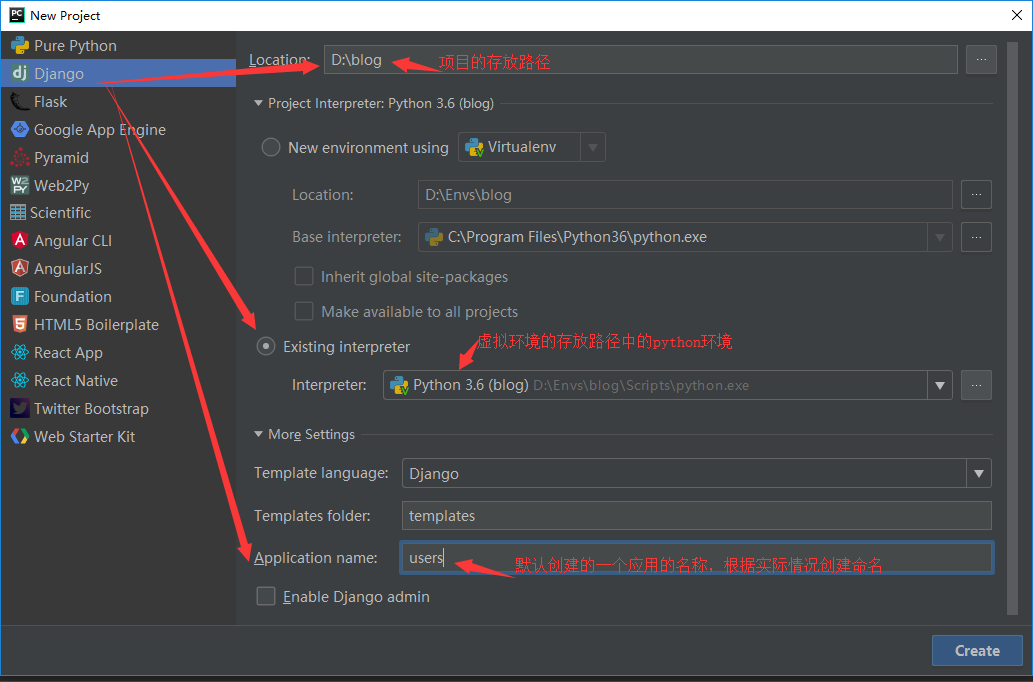
1. 创建项目命令如下：

django-admin startproject 项目名称

1. 创建应用命令如下：

django-admin startapp 应用名称

方法二：**利用pycharm方式创建(推荐)**



1. 项目的settings.py文件的基本配置
2. DEBUG：一个布尔值，该布尔函数用以打开项目的调试模式。如果设置为true，Django会显示详细的错误页面。当你移动到一个生产环境，记得必须设置它为False。不要将网站以调试模式部署到生产中，这样会暴露你的项目的敏感数据。
3. ALLOWED\_HOSTS：在调试模式或正在运行测试时，这个选项不起作用。一旦你关闭调试模式，并将你的网站布置在正式的生产环境，就必须在settings.py中添加部署环境的域名/主机，以便允许服务器运行Django站点。
4. INSTALLED\_APPS ：这个设置告诉Django哪些app是启用的。

默认情况下，Django包含下面application

1. django.contrib.admin： Django管理站点.
2. django.contrib.auth： Django认证模块.
3. django.contrib.contenttypes: Django内容类型框架.
4. django.contrib.sessions： Django会话框架.
5. django.contrib.messages： Django消息传递框架.
6. django.contrib.staticfiles： Django管理静态文件的框架
7. MIDDLEWARE\_CLASSES ：一个元组，包含需要执行的中间件
8. DATABASES：是一个用于在项目中使用的，包含所有数据库的字典。

默认的配置使用sqlite3数据库

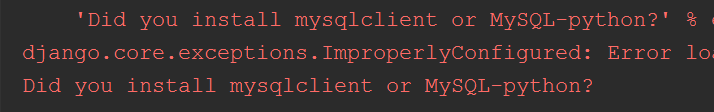


拓展：数据库引擎可以是如下：

django.db.backends.sqlite3(默认)、

django.db.backends.mysql(常用)、

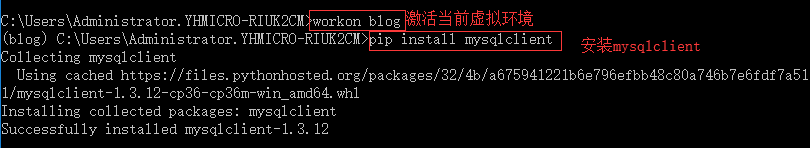
因为修改了配置中的数据库引擎，运行后会报如下错误：



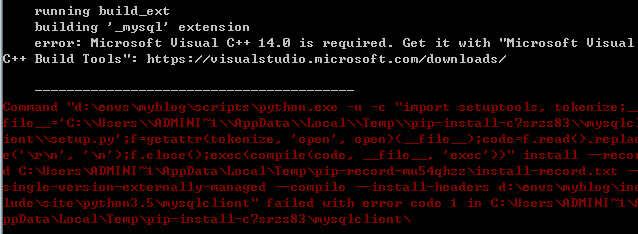
需要在虚拟环境中安装mysqlclient才能支持mysql驱动

命令:

pip install mysqlclient



可能的出错！



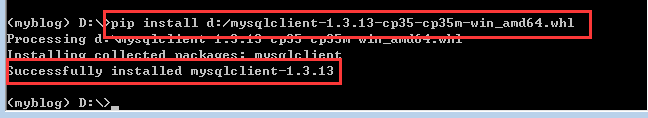
解决办法:

运行已经下在好的mysqlclient-1.3.13-cp35-cp35m-win\_amd64.whl 和Python版本一致，64位

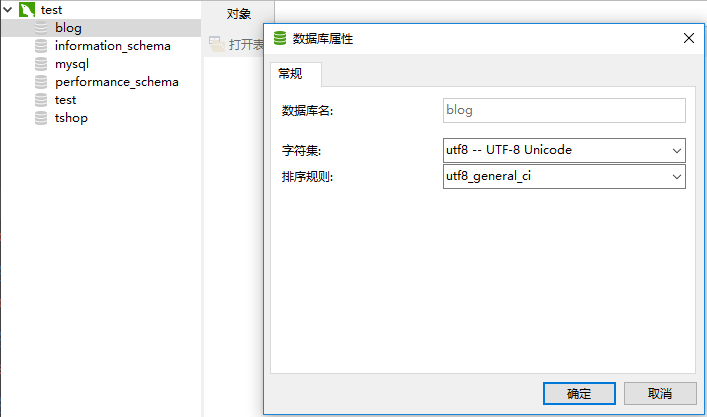
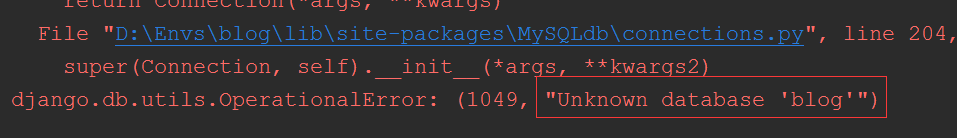
命令:

pip install d:/mysqlclient-1.3.13-cp35-cp35m-win\_amd64.whl

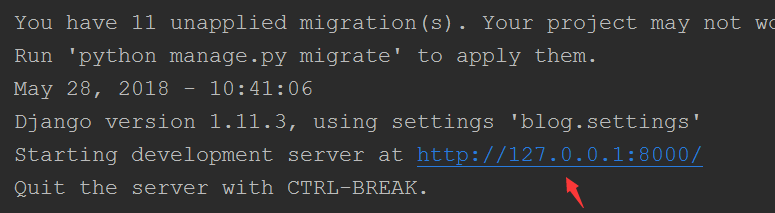
成功！



重新运行后仍然报错，原因是没有创建实际名称的数据库



再次运行项目，就不会报错了



1. LANGUAGE\_CODE：为Django项目定义默认的使用语言。



1. 在TIME\_ZONE配置中修改默认的时区配置



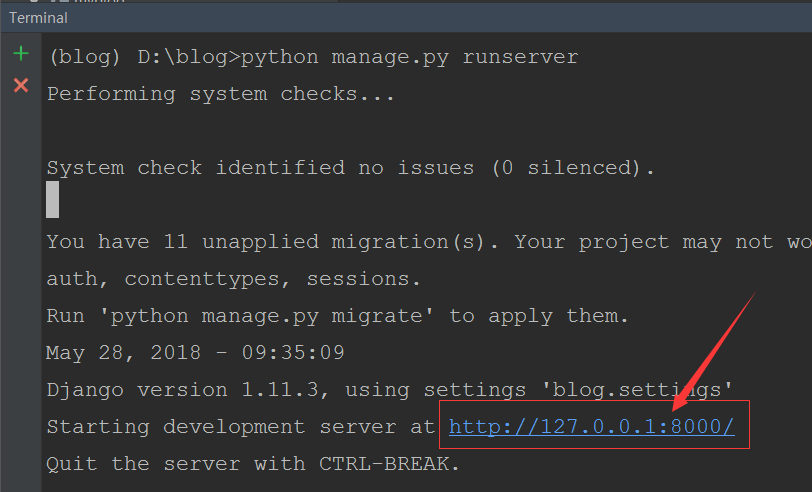
1. 在USE\_TZ配置中修改默认的时间配置



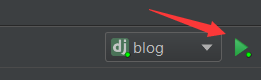
1. 安装环境之后，运行django环境

方法一：命令运行

python manage.py runserver (切换对应的磁盘目录)



方法二：在pycharm环境中点击运行按钮



在地址栏中敲入http:127.0.0.1:8000,能看到如下默认的运行界面



看到了该页面说明Django环境搭建成功了。

1. Django框架项目目录介绍

manage.py ： Django项目里面的工具，通过它可以调用django shell和数据库等。

myblog/

| ---  settings.py ： 包含了项目的默认设置，包括数据库信息，调试标志以及其他一些工作的变量。

| ---  urls.py ： 负责把URL模式映射到应用程序。

| --- wsgi.py :  用于项目部署。

blog /

| --- admin.py  :  django 自带admin后台管理，将models.py 中表映射到后台。

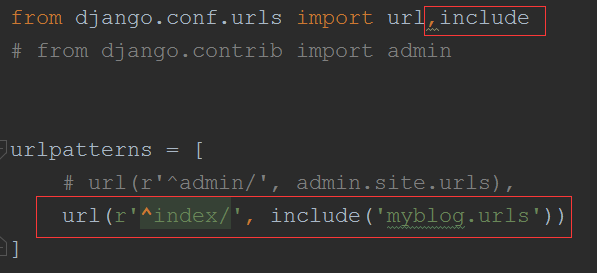
| --- apps.py :  应用的相关配置。

| --- models.py  : Django 自带的ORM，用于设计数据库表。

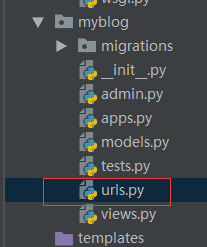
| --- tests.py  :  用于编写Django单元测试。

| --- views.py ：视图文件，用于编写功能的主要处理逻辑

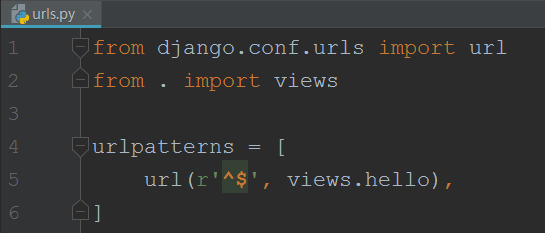
1. Django框架第一个简单的小项目
2. 在项目中的urls.py中添加对应的url映射路径



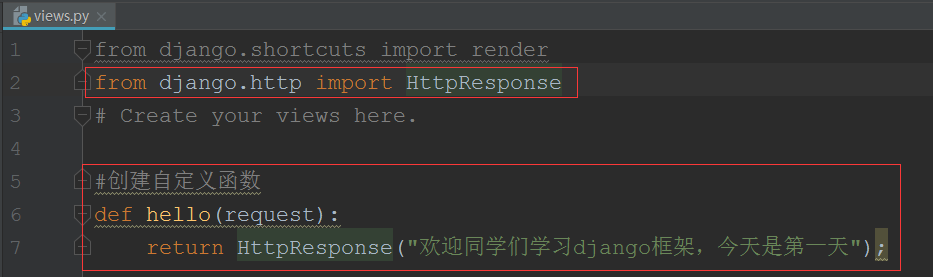
1. 为了方便管理每一个应用，建议在不同应用中添加urls.py文件



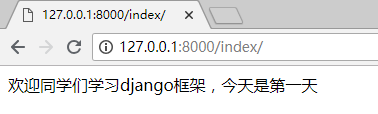
添加如下代码：



1. 在应用的views.py文件中添加如下代码：



点击运行链接，能够看到自定义函数中的内容

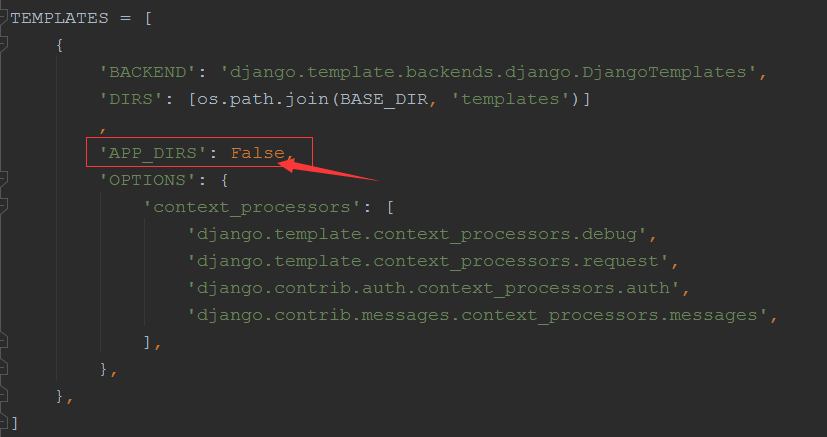


1. 自定义模板和静态文件位置



1. 自定义模板位置

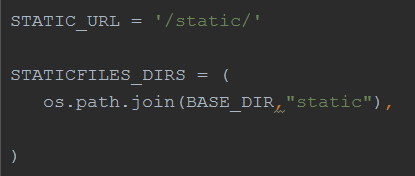
编辑项目下的settings.py文件，修改TEMPLATES的值



将APP\_DIRS设置为False，即不再允许Django按照默认方式寻找模板文件

1. 自定义静态文件位置

编辑项目下的settings.py文件，修改TEMPLATES的值



其中，第一行说明了Django中认定的静态文件是什么，第二行说明怎样查找静态文件地址

1. 通用静态文件和基础模板

模板标签语法：

1. 引入静态文件

{% static ‘文件地址’ %}

1. 加载静态文件

{% load static %}

1. 文件包含

{% include ‘文件地址’%}

1. Block块标签

{% block 自定义名称 %}{% endblock %}

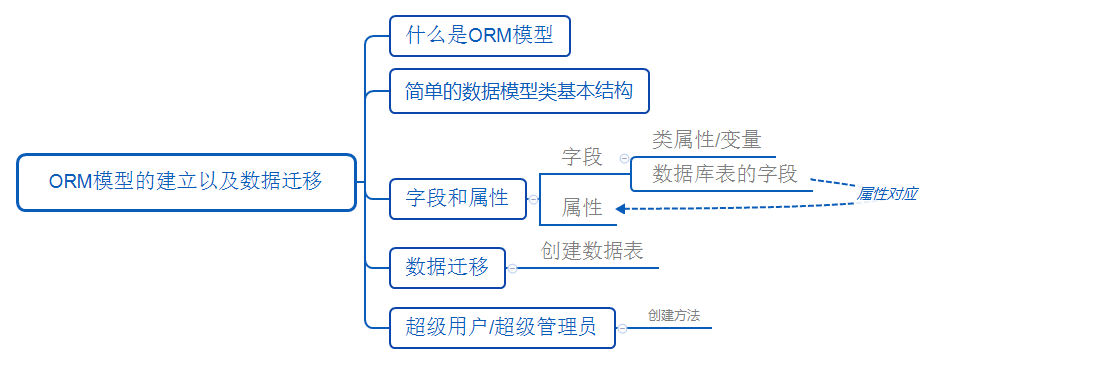
1. 注释标签

{# 注释内容 #}

1. 模板继承标签

{% extends “文件地址” %}

1. 模型ORM的建立以及数据迁移



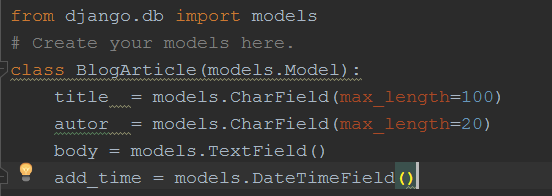
1. 什么是ORM模型

ORM的作用是在关系型数据库和业务实体对象之间进行映射，这样在操作业务对象时，就不需要再去和复杂的SQL语句打交道，只需要简单的操作对象的属性和方法。

ORM的表现方式就是编写数据模型类，这些类可以写到任何文件中，通常写在每个应用的models.py文件中，每个数据模型类都是django.db.models.Model的子类

组成方式：应用的名称(小写字母)+数据模型类(小写字母) 例如：blog\_blogarticles

1. 简单的数据模型类基本结构



数据模型类(model)依然是Python中的类，可以在其中命名属性(变量)和方法。

每个属性(变量)对应着数据库中的一个字段，变量也被称为字段，Django会为每个数据模型类设置一个自增长的id字段并作为该表的主键，不需要再数据模型类中写出来，

字段属性类型

1. CharField：用于保存字符串，再使用时需要指明长度
2. TextField：和CharField一样，区别在于无上限，通常用于保存大文本
3. DateField和DateTimeField：用于保存时间

title = models.CharField(max\_length=50)

字段名 varchar(100)

content=models.TextField()

字段content longtext 长文本

add\_time = models.DateTimeField()

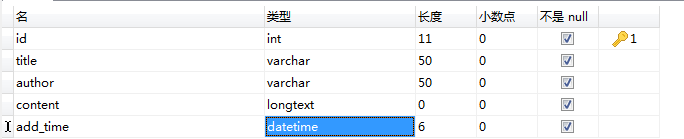
字段add\_time datetime 年月日时分秒

注意：

date -- > '1000-01-01' to '9999-12-31'.

datetime --> '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

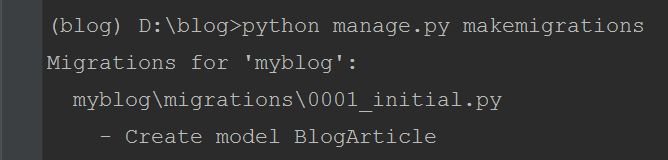
timestamp -- > '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC



1. 数据迁移

在虚拟环境下执行数据迁移命令

命令：python manage.py makemigrations

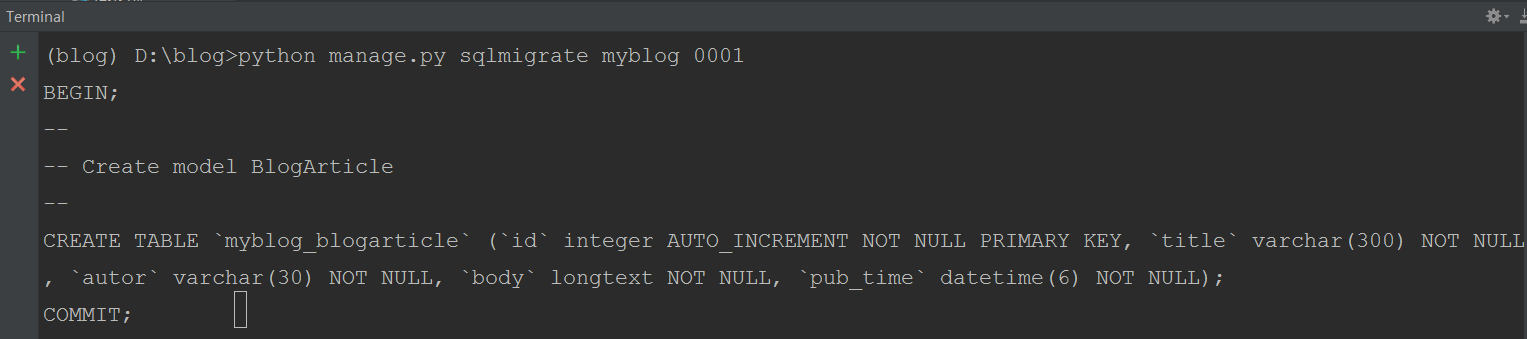


在对应的应用里面的migrations文件夹中产生了一个0001\_initial.py文件

1. 执行sqlmigrate命令查看生成的文件本质

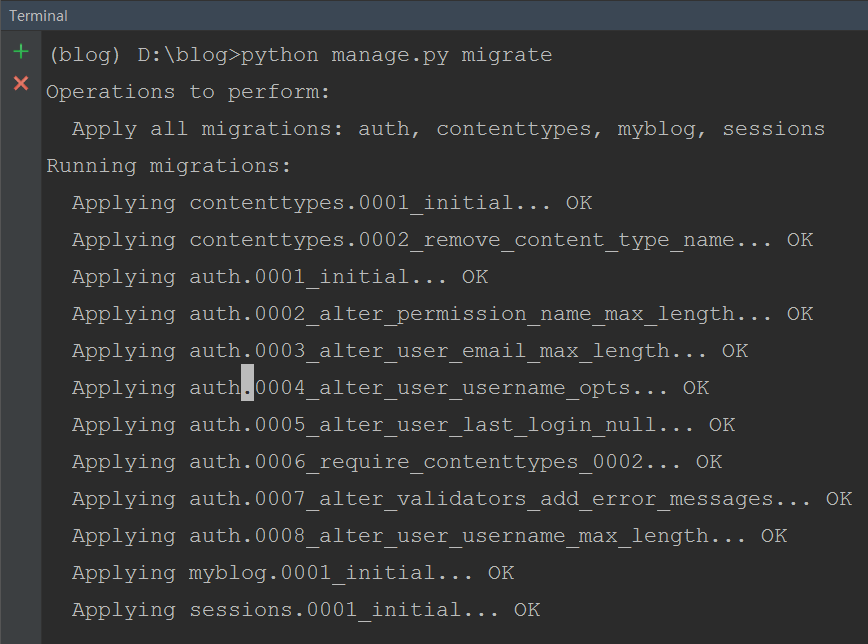
命令：python manage.py sqlmigrate 应用名 文件名前缀

例如：python manage.py sqlmigrate myblog 0001

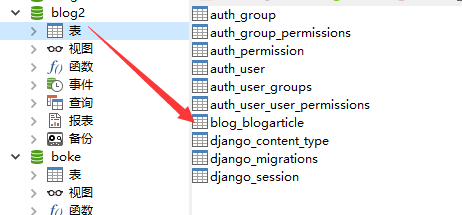


1. 执行migrate命令真正生成数据库表文件

命令:python manage.py migrate



查看navicat mysql工具中的blog数据库，能够看到生成的数据表



1. 项目调试Django+Pycharm
2. 在开发Django程序的过程中，数据库操作形成瓶颈，想要在log中查看程序具体执行了哪些SQL语句调试结果，这时候就可以使需要在settings.py中加上这么几句

LOGGING = {  
  
 'version': 1,  
 'disable\_existing\_loggers': False,  
 'handlers': {  
 'console':{  
 'level':'DEBUG',  
 'class':'logging.StreamHandler',  
 },  
 },  
 'loggers': {  
 'django.db.backends': {  
 'handlers': ['console'],  
 'propagate': True,  
 'level':'DEBUG',  
 },  
 }  
}

1. 方法2

在views.py 中

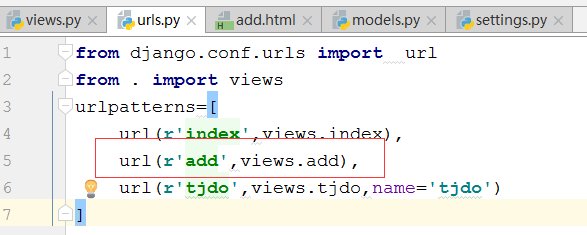
from django.db import connection

使用时 print(connection.queries)

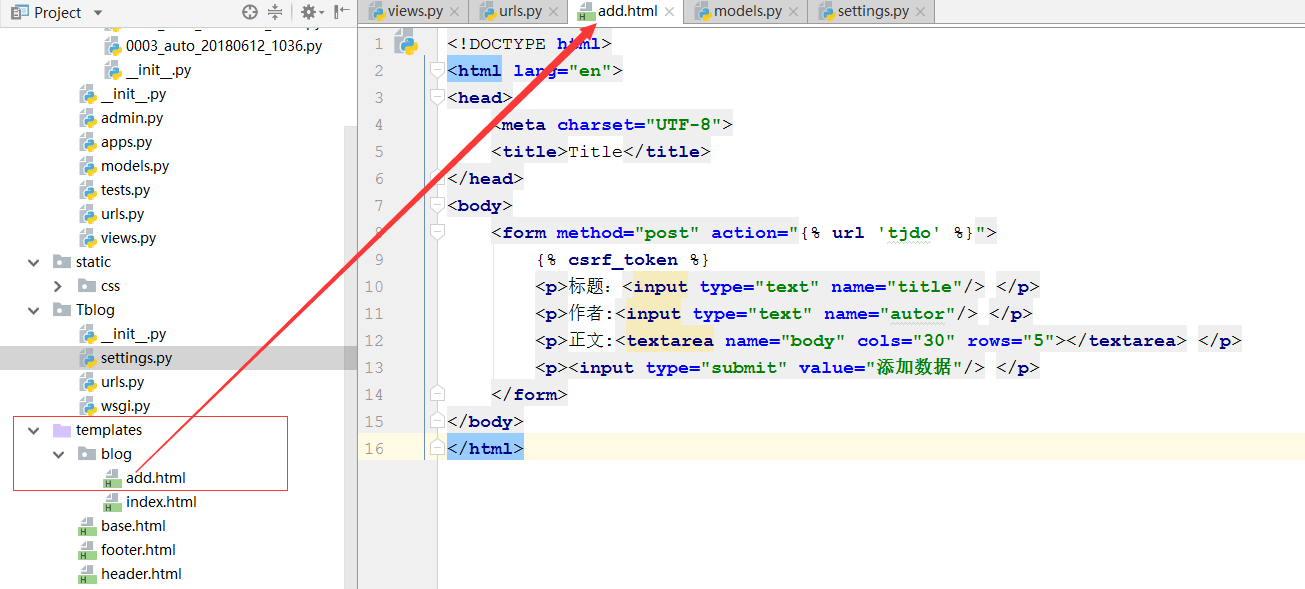


示例：博客的新闻数据添加

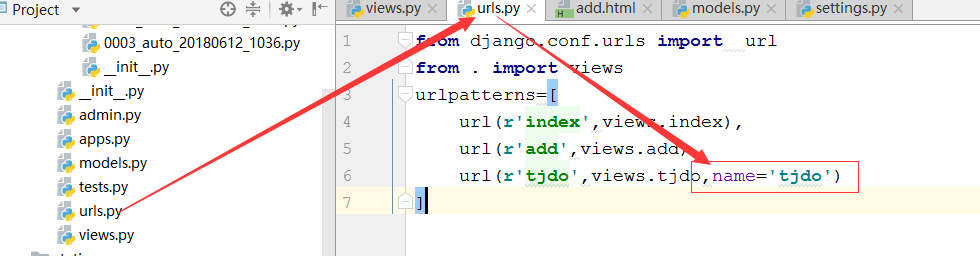
1. 创建应用路由调用控制器中函数



1. 在Templates目录中创建模版文件



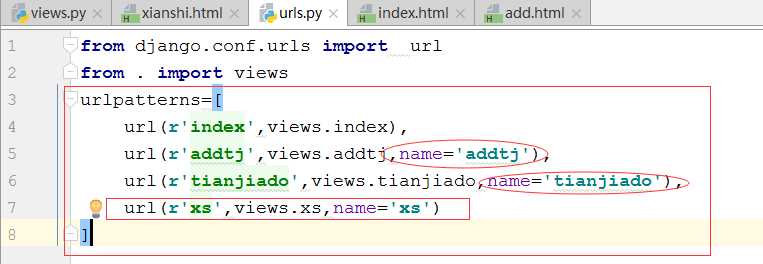
1. 创建提交表单数据的路由，注意：这是是**url路由的反向解析**



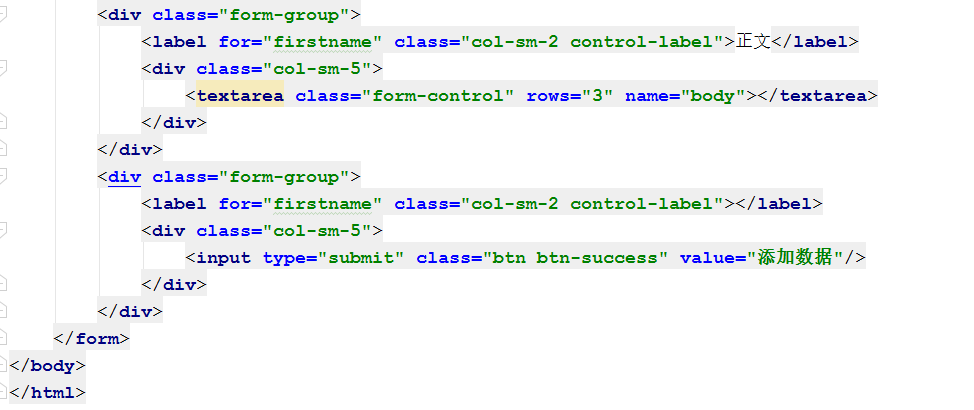
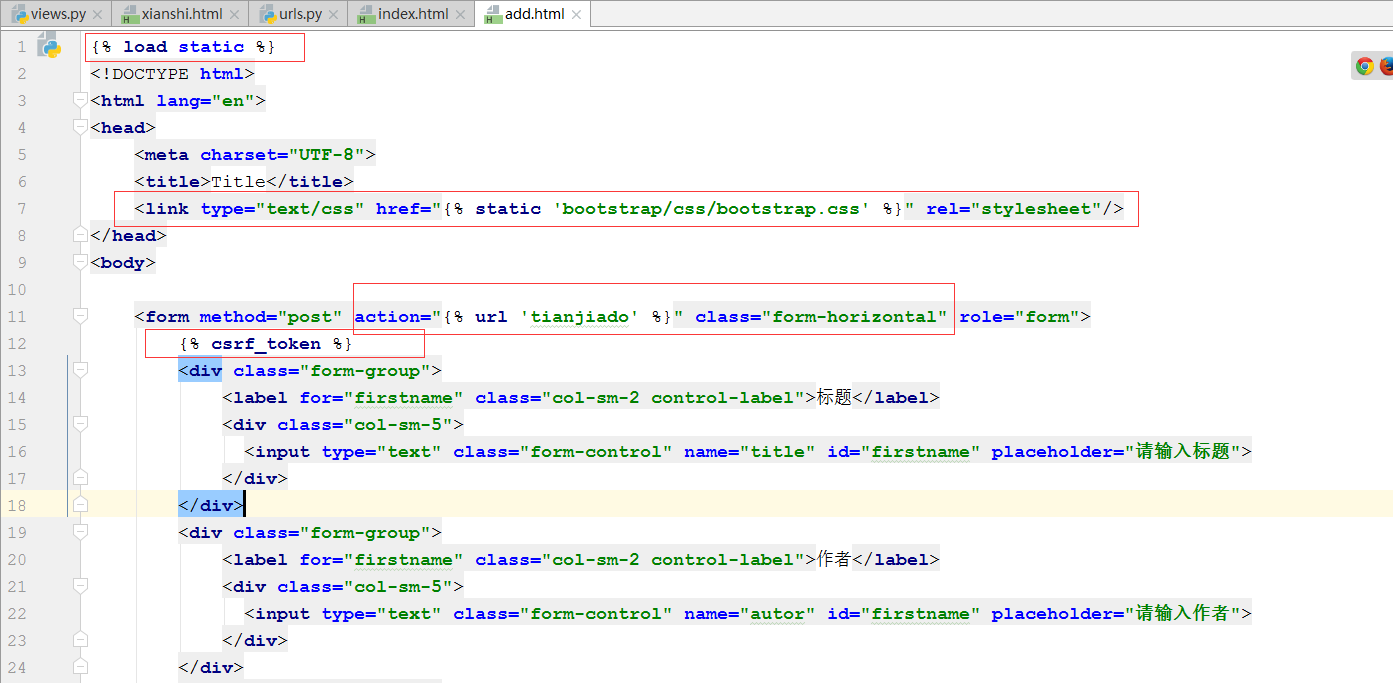
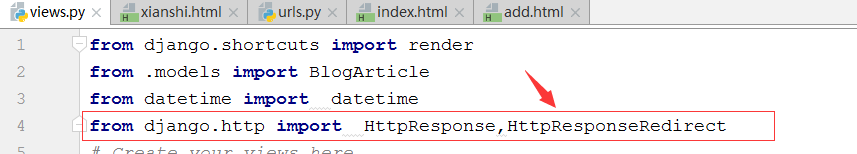
1. 在控制器文件中创建处理表单数据的函数



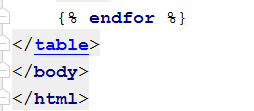
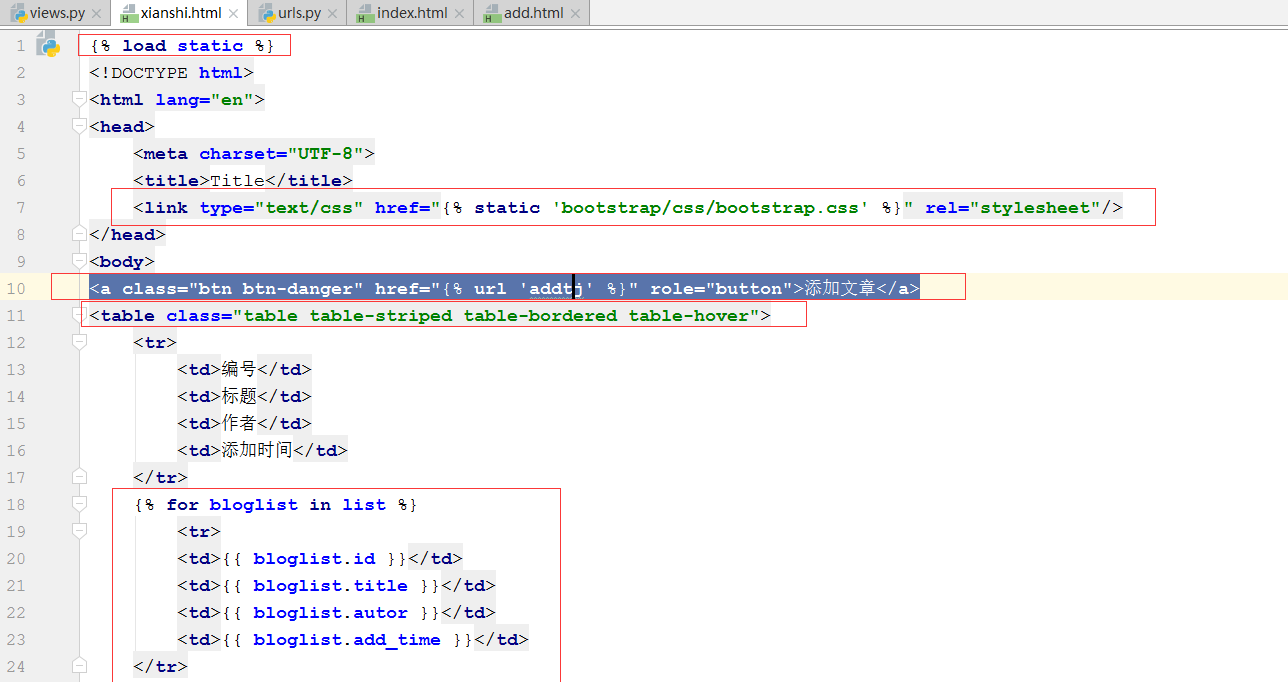
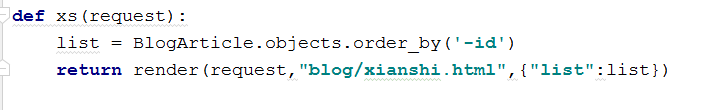
python manage.py runserver 127.0.0.1:8080 运行多个项目的时候切换端口号



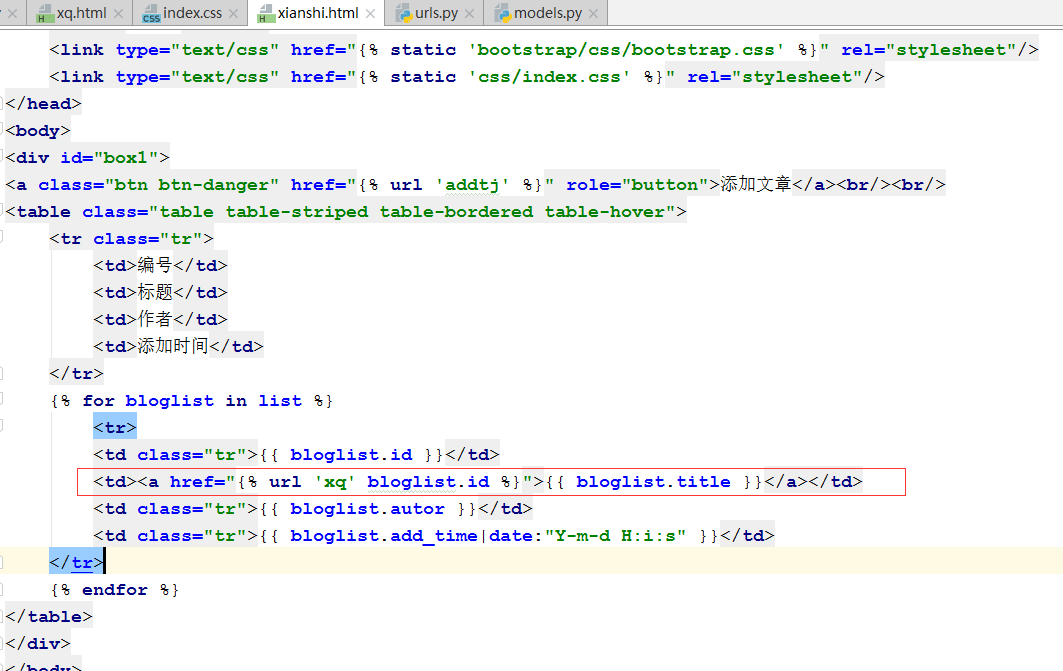
1. 修改添加处理方法以及美化添加表单页面



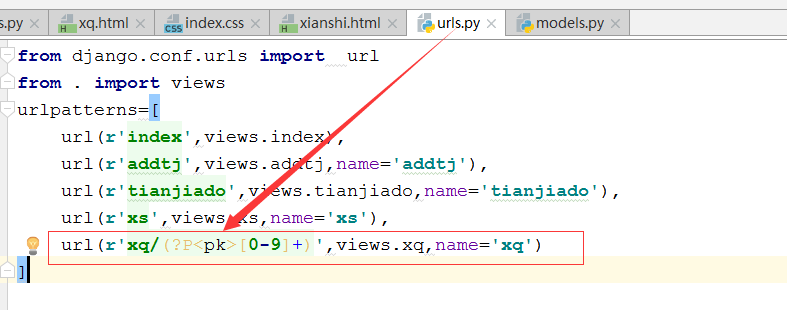
1. 创建一个显示数据的方法以及对应模版界面



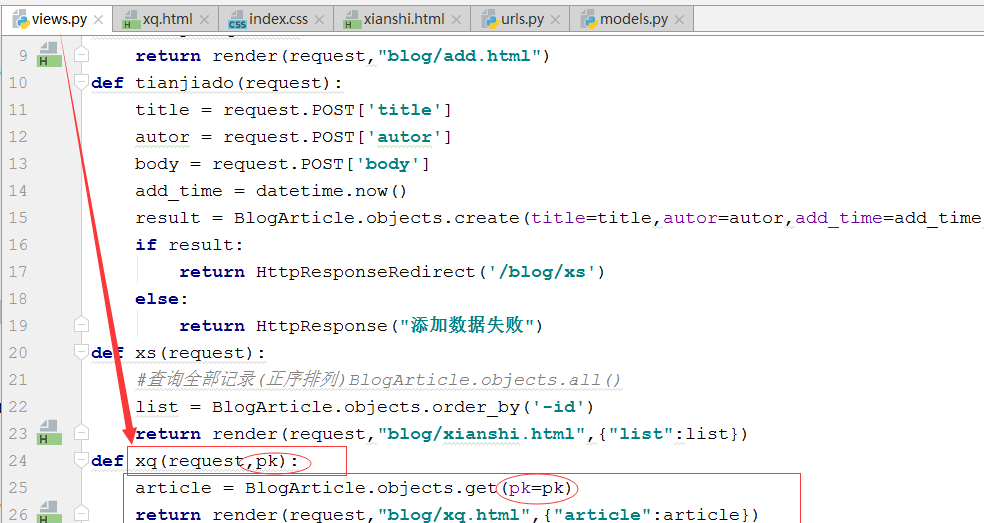
1. 处理显示页面给标题加超链接



定义xq路由



创建xq函数并接受路由中传递的参数



模板页面

