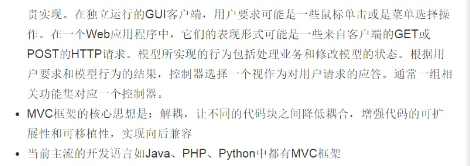
# 入门

## MVC简介



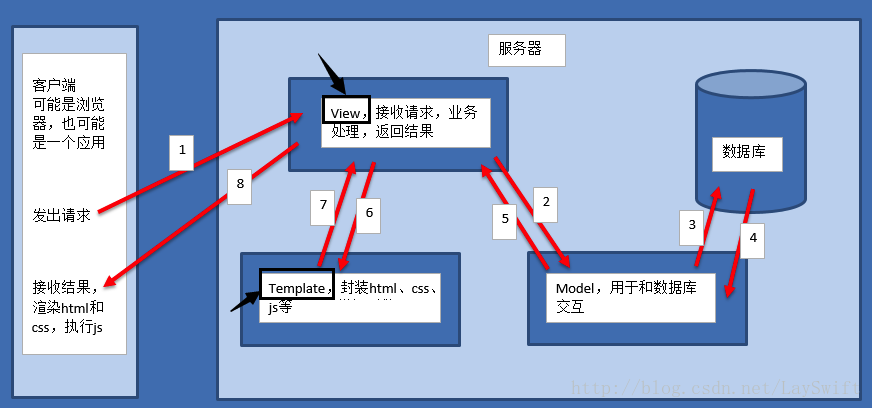


MVC的全拼为Model-View-Controller，最早由TrygveReenskaug在1978年提出，是施乐帕罗奥多研究中心(Xerox PARC)在20世纪80年代为程序语言Smalltalk发明的一种软件设计模式，是为了将传统的输入(input)、处理(processing)、输出(output)任务运用到图形化用户交互模型中而设计的。MVC结构提供了一种按功能对各种对象进行分割的方法(这些对象是用来维护和表现数据的)，其目的是为了将各对象间的耦合程度减至最小。

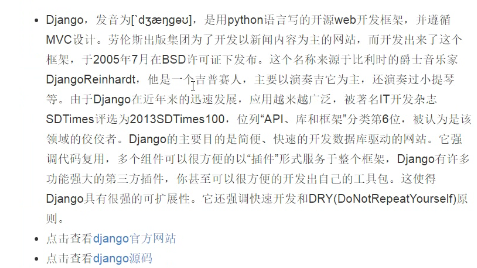
现在的MVC结构中，模型(Model)代表应用程序的数据(data)和用于控制访问和修改这些数据的业务规则(business rule)。通常模型被用来作为对现实世界中一个处理过程的软件近似，当定义一个模型时，可以采用一般的简单的建模技术。当模型发生变化时，它会通知视(View)，并且为视提供查询模型相关状态的能力。同时，它也为控制器(Controller)提供访问封装在模型内部的应用程序功能的能力。一个视(View)用来组织模型的内容。它从模型那里获得数据并指定这些数据如何表现。当模型变化时，视负责维持数据表现的一致性。视同时将用户要求告知控制器(Controller)。控制器(Controller)定义了应用程序的行为；它负责对来自视的用户要求进行解释，并把这些要求映射成相应的行为，这些行为由模型负责实现，在独立运行的GUI客户端，用户要求可能是一些鼠标单击或是菜单选择操作。在一个Web应用程序中，它们的表现形式可能是一些来自客户端的GET或POST的HTTP请求。模型所实现的行为包括处理业务和修改模型的状态。根据用户要求和模型行为的结果，控制器选择一个视作为对用户请求的应答。通常一组相关功能集对应一个控制器。

MVC框架的核心思想是：解耦，让不同的代码块之间降低耦合，增强代码的可扩展性和可移植性，实现向后兼容。

* M全拼为Model，主要封装对数据库层的访问，内嵌ORM框架，实现面向对象的编程来操作数据库，不用考虑数据库的差异性，简单配置就可以完成数据库切换。
* V全拼为View，用于封装结果，内嵌了模板引擎，实现动态展示数据。
* C全拼为Controller，用于接收GET或POST请求，处理业务逻辑，与Model和View交互，返回结果。



## Django简介



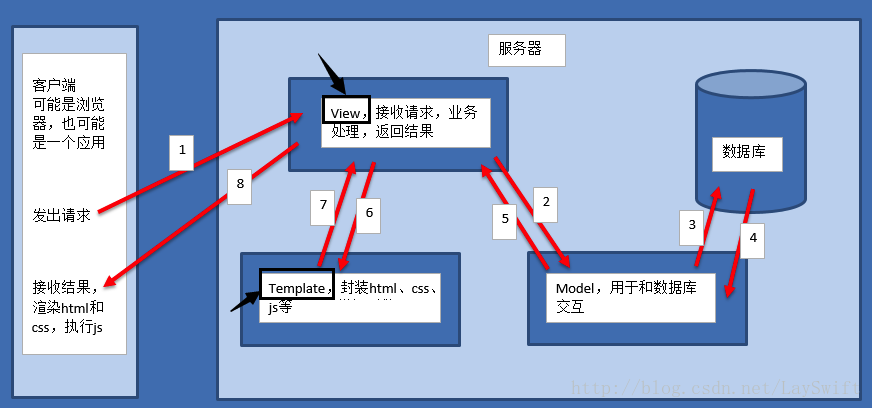
django官方网站：<https://www.djangoproject.com/>

django源码：<https://github.com/django/django>

django文档：<https://docs.djangoproject.com/zh-hans/2.1/_modules/django/>

<https://docs.djangoproject.com/en/3.0/contents/>





# 入门项目1

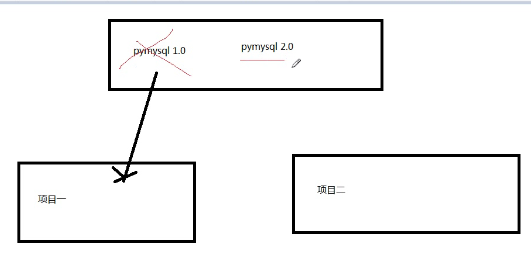
## 安装django环境

### 问题

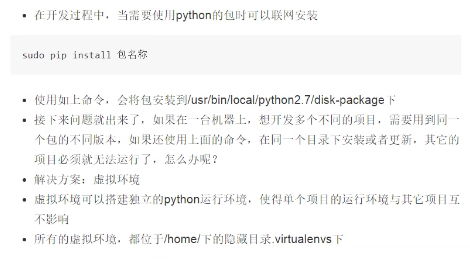
sudo pip install django

会将包安装到/usr/bin/local/python2.7/disk-package下。

出现的问题：



不同项目需要包的不同的版本，



### 创建

hlx2@NLP:~$ sudo apt-get install python-virtualenv

hlx2@NLP:~$ sudo easy\_install virtualenvwrapper

* 安装命令：hlx2@NLP:~$ pip install --ignore-installed virtualenv

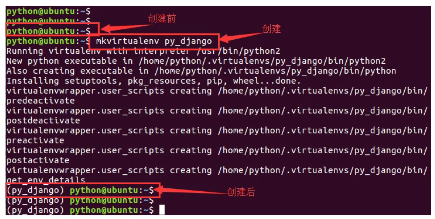
安装出现的错误：

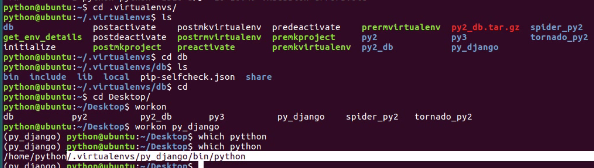
ERROR: Cannot uninstall 'filelock'. It is a distutils installed project and thus we cannot accurately determine which files belong to it which would lead to only a partial uninstall.

* hlx2@NLP:~$ pip install virtualenvwrapper



mkvirtualenv py\_djan





hlx2@NLP:~$ mkvirtualenv py\_djan

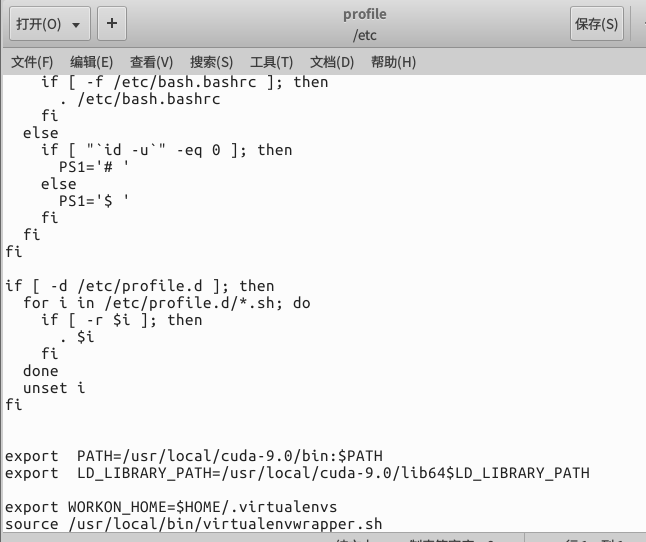
mkvirtualenv：未找到命令

### 设置环境变量

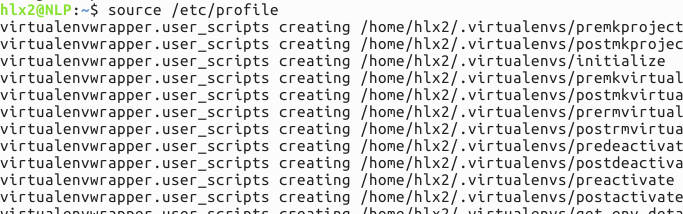
hlx2@NLP:~$ sudo gedit /etc/profile

export WORKON\_HOME=$HOME/.virtualenvs

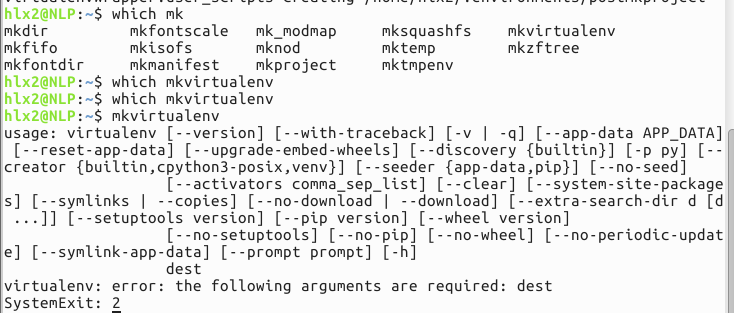
source /usr/local/bin/virtualenvwrapper.sh



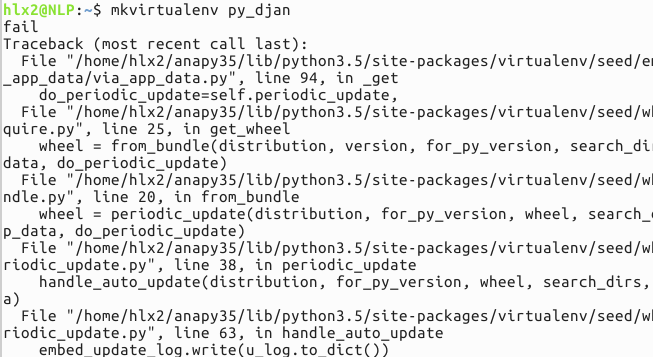
hlx2@NLP:~$ source /etc/profile

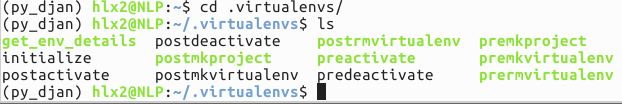


在hlx2@NLP:~$ gedit .bashrc修改可以成功。



再次运行：hlx2@NLP:~$ mkvirtualenv py\_djan





使用虚拟环境：

workon py\_djan

hlx2@NLP:~/.virtualenvs$ workon py\_djan

(py\_djan) hlx2@NLP:~/.virtualenvs$ which python

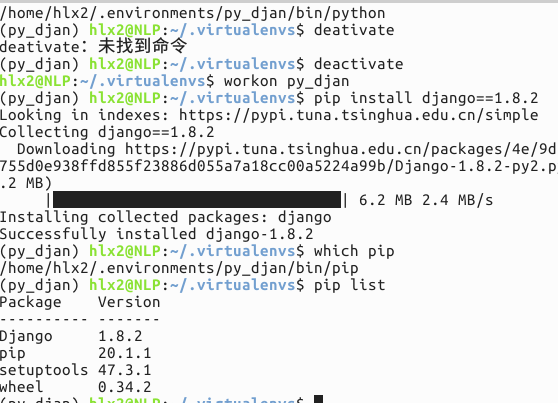
/home/hlx2/.environments/py\_djan/bin/python

退出：

(py\_djan) hlx2@NLP:~/.virtualenvs$ deactivate

安装包：

(py\_djan) hlx2@NLP:~/.virtualenvs$ pip install django==1.8.2



## 创建项目

示例：图书-英雄项目

一本图书对多个英雄，一个英雄对应一本图书

django-admin startproject test1

(py\_djan) hlx2@NLP:~/.virtualenvs$ django-admin startproject test1

(py\_djan) hlx2@NLP:~/.virtualenvs$ django-admin startproject test1

(py\_djan) hlx2@NLP:~/.virtualenvs$ ls

get\_env\_details postdeactivate postrmvirtualenv premkproject test1

initialize postmkproject preactivate premkvirtualenv

postactivate postmkvirtualenv predeactivate prermvirtualenv

(py\_djan) hlx2@NLP:~/.virtualenvs$ cd test

bash: cd: test: 没有那个文件或目录

(py\_djan) hlx2@NLP:~/.virtualenvs$ cd test1

(py\_djan) hlx2@NLP:~/.virtualenvs/test1$ ls

manage.py test1

(py\_djan) hlx2@NLP:~/.virtualenvs/test1$ cd ..

(py\_djan) hlx2@NLP:~/.virtualenvs$ cd ..

(py\_djan) hlx2@NLP:~$ mkdir py\_djan

(py\_djan) hlx2@NLP:~$ cd py\_djan/

(py\_djan) hlx2@NLP:~/py\_djan$ django-admin startproject test1

# 创建的项目在当前路径下。

(py\_djan) hlx2@NLP:~/py\_djan$ ls

test1

(py\_djan) hlx2@NLP:~/py\_djan$ cd test1/

(py\_djan) hlx2@NLP:~/py\_djan/test1$ ls

manage.py test1

(py\_djan) hlx2@NLP:~/py\_djan/test1$ ls

manage.py test1

(py\_djan) hlx2@NLP:~/py\_djan/test1$

(py\_djan) hlx2@NLP:~/py\_djan/test1$ tree

.

├── manage.py # 项目入口

└── test1 # 与项目名称同名的目录

├── \_\_init\_\_.py #

├── settings.py # 设置

├── urls.py # url的配置

└── wsgi.py # 发布使用

1 directory, 5 files

urls.py：

url: <http://itcast.cn/1/>

进行页面的匹配，

url(正则表达式)，拿1/和正则表达式匹配，获得具体页面。

进入项目目录下，

创建应用名称，booktest，python manage.py startapp booktest

一个项目有多个应用

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py startapp booktest

(py\_djan) hlx2@NLP:~/py\_djan/test1$ tree

.

├── booktest

│   ├── admin.py # 后台u管理

│   ├── \_\_init\_\_.py

│   ├── migrations # 数据的迁移

│   │   └── \_\_init\_\_.py

│   ├── models.py # 模型的开发

│   ├── tests.py # 测试

│   └── views.py #

├── manage.py

└── test1

├── \_\_init\_\_.py

├── \_\_pycache\_\_

│   ├── \_\_init\_\_.cpython-35.pyc

│   └── settings.cpython-35.pyc

├── settings.py

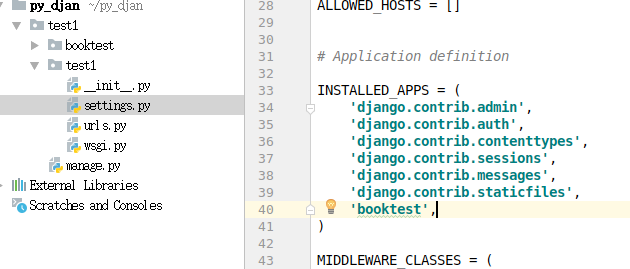
├── urls.py

└── wsgi.py

4 directories, 13 files

(py\_djan) hlx2@NLP:~/py\_djan/test1$

将应用加到settings.py



## 定义模型类

manage.py的

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "test1.settings")

test1.settings：引用，联系。

定义书的类：

booktest.models.py



生成迁移文件，根据类和表结构.

python manage.py makemigrations

- 生成迁移文件，根据模型类生成创建表的语句

- 执行迁移，根据第一步生成的语句在数据库总创建表

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py makemigrations

Migrations for 'booktest':

0001\_initial.py:

- Create model BookInfo

- Create model HeroInfo

进行迁移

python manage.py migrate

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py migrate

Operations to perform:

Synchronize unmigrated apps: staticfiles, messages

Apply all migrations: contenttypes, sessions, admin, auth, booktest

Synchronizing apps without migrations:

Creating tables...

Running deferred SQL...

Installing custom SQL...

Running migrations:

Rendering model states... DONE

Applying contenttypes.0001\_initial... OK

Applying auth.0001\_initial... OK

Applying admin.0001\_initial... 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

Applying auth.0004\_alter\_user\_username\_opts... OK

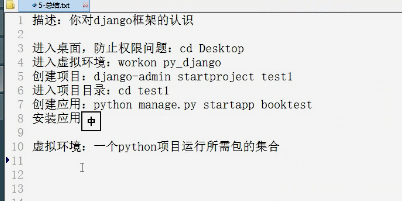
Applying auth.0005\_alter\_user\_last\_login\_null... OK

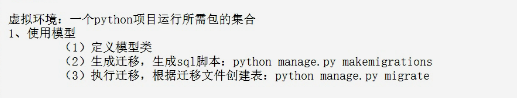
Applying auth.0006\_require\_contenttypes\_0002... OK

Applying booktest.0001\_initial... OK

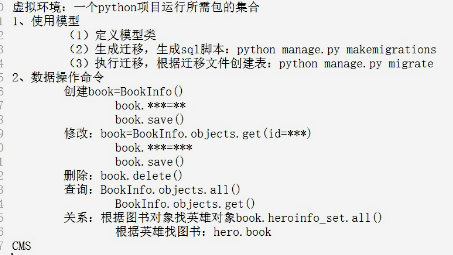
Applying sessions.0001\_initial... OK

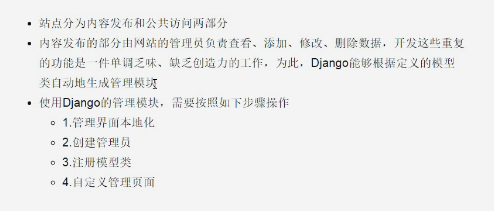
(py\_djan) hlx2@NLP:~/py\_djan/test1$

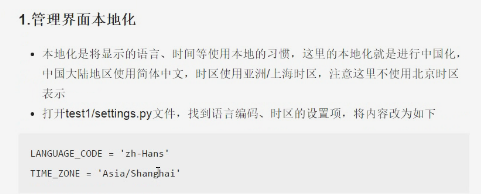




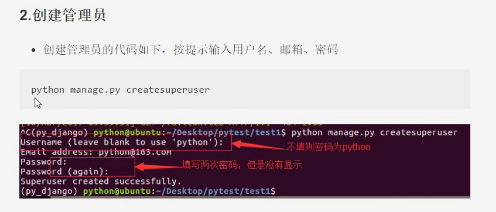
## 使用后台管理数据







python manage.py createsuperuser



(py\_djan) hlx2@NLP:~/py\_djan/test1$ ls

booktest db.sqlite3 manage.py test1

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py createsuperuser

Username (leave blank to use 'hlx2'): hlx2

Email address: luanxin\_huang@qq.com

Password:

Password (again):

Superuser created successfully.

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py runserver

Performing system checks...

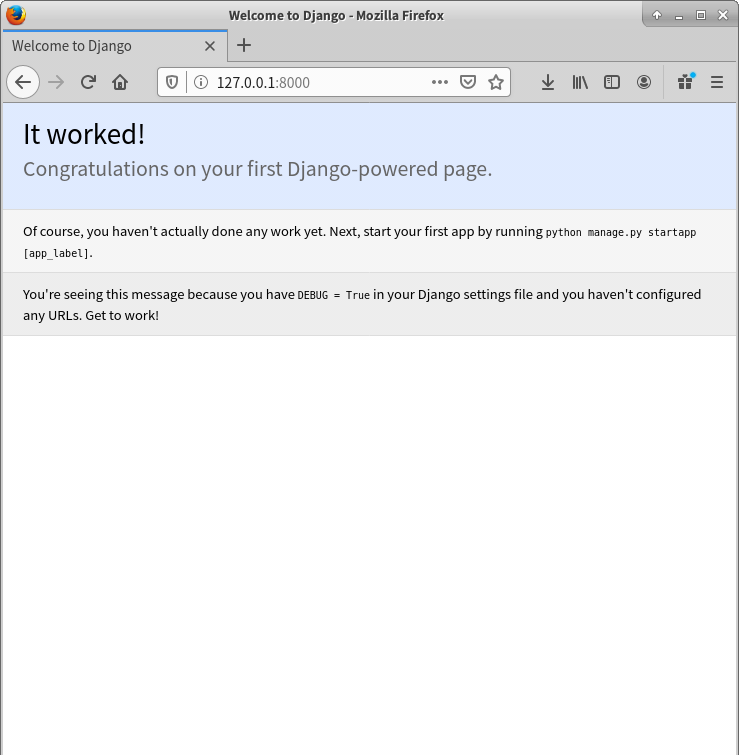
System check identified no issues (0 silenced).

July 13, 2020 - 01:08:28

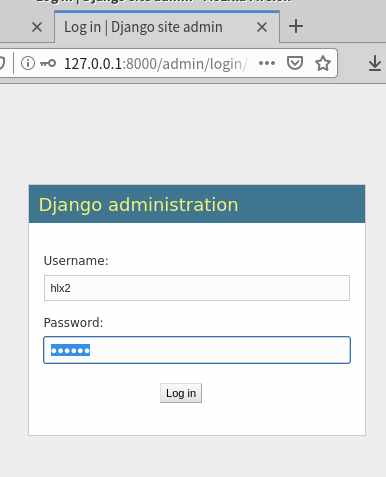
Django version 1.8.2, using settings 'test1.settings'

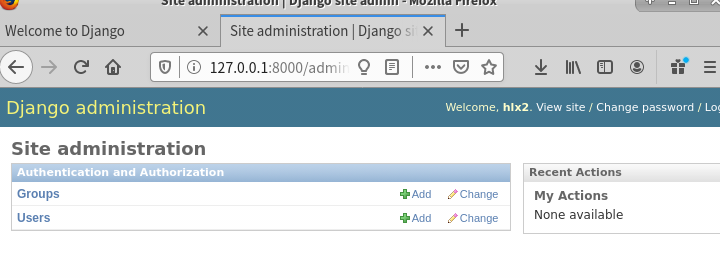
Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.



<http://127.0.0.1:8000/admin>





修改语言为中文：

Py\_dian/test1/test1/settings.py

# Internationalization

# https://docs.djangoproject.com/en/1.8/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

==>

LANGUAGE\_CODE = 'zh-Hanz'

/home/hlx2/.environments/py\_djan/lib/python3.5/site-packages/django/contrib/admin/locale/



TIME\_ZONE = 'UTC'

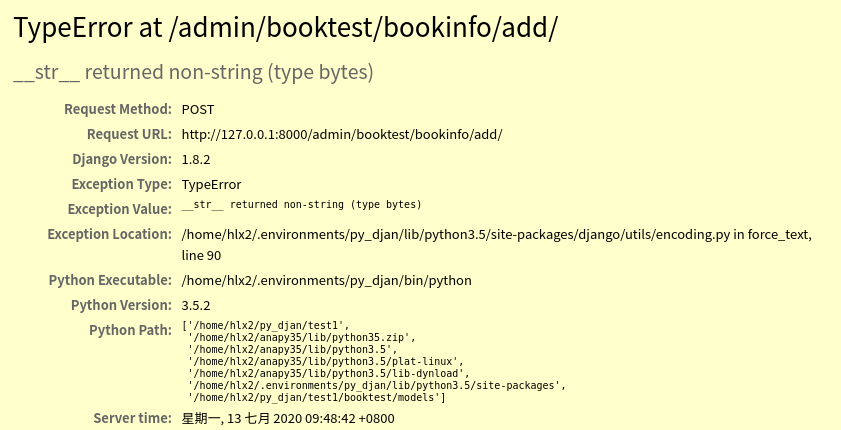
===>

TIME\_ZONE = 'Asia/Shanghai'

重启服务器：

(py\_djan) hlx2@NLP:~/py\_djan/test1$ python manage.py runserver









添加hero info例子，修改HeroInfo object这个名字。

/home/hlx2/py\_djan/test1/booktest/admin.py

# 修改增加数据的名称改变

class BookInfoAdmin(admin.ModelAdmin):

# 模型类的字段

list\_display = ['id','title','pub\_date']

admin.site.register(BookInfo, BookInfoAdmin)



## 编写视图接受请求，并调用模型类获得数据

在view写视图，返回请求，要在test1/urls加入映射

/home/hlx2/py\_djan/test1/booktest/views.py

# HttpRequest,请求作为视图，必须返回响应

def index(request,):

# HttpResponse

return HttpResponse('<h1>hello word</h1>')

/home/hlx2/py\_djan/test1/test1/urls.py

也可以在/home/hlx2/py\_djan/test1/booktest下添加url。

/home/hlx2/py\_djan/test1/booktest/urls.py

**from** django.conf.urls **import** url  
**from** .views **import** \*  
*# import views*urlpatterns = [  
 url(**'^index/$'**, index),  
]

访问：

<http://127.0.0.1:8000/index/>

'^/$'==》http://127.0.0.1:8000

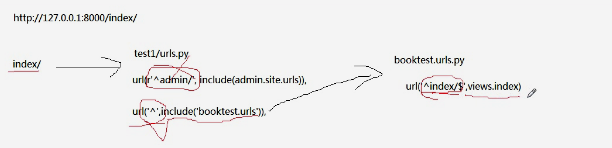
程序的入口manage.py ==》test1.settings

settings.py ROOT\_URL\_CONF=’test1.urls’

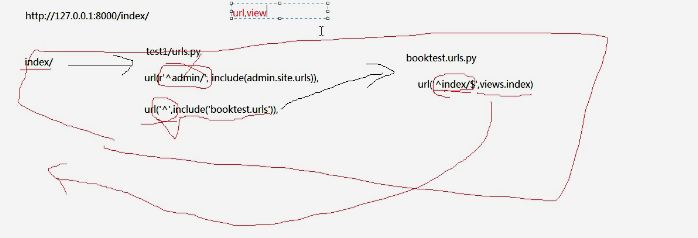
从’test1.urls’开始匹配

url(**'^'**, include(**'booktest.urls'**)),

从**booktest.urls匹配**



匹配url，获得视图，返回视图render，渲染，显示。

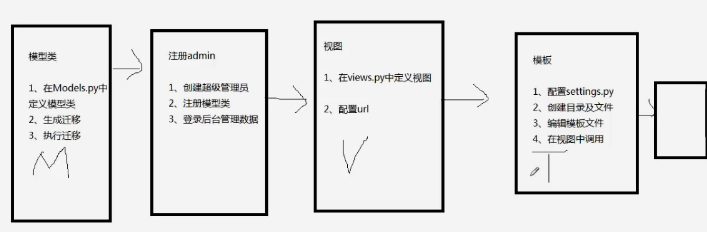


view的使用，取决于浏览器的请求。

## 定义模板，用于展示获取到的数据

在test1增加template

在template新建和应用重名的目录，booktest，在booktest目录下增加index.html



## 效果