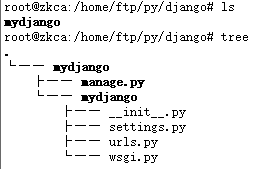
**1.安装**Django2.0.4

pip3 install django==2.0.4

2.创建项目

django-admin.py startproject mydjango



目录说明：

mydjango: 项目的容器。

manage.py: 一个实用的命令行工具，可让你以各种方式与该 Django 项目进行交互。

mydjango/\_\_init\_\_.py: 一个空文件，告诉 Python 该目录是一个 Python 包。

mydjango/settings.py: 该 Django 项目的设置/配置。

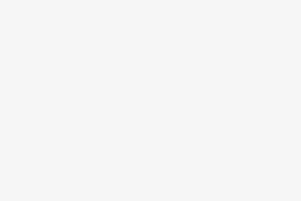
mydjango/urls.py: 该 Django 项目的 URL 声明; 一份由 Django 驱动的网站"目录"。

mydjango/wsgi.py: 一个 WSGI 兼容的 Web 服务器的入口，以便运行你的项目。

**在我们创建的项目里修改setting.py文件，ALLOWED\_HOSTS = ['\*'] ＃在这里请求的host添加了\***

启动:python manage.py runserver 0.0.0.0:8089 （注意服务器防火墙放开8089端口）

访问：<http://192.168.1.237:8089/>



2.项目上手：<http://blog.51cto.com/1576349/2064615>

<http://djangobook.py3k.cn/2.0/>

**1. 新建一个 django project**

|  |
| --- |
| django-admin.py startproject project\_name 特别是在 windows 上，如果报错，尝试用 django-admin 代替 django-admin.py 试试 |

注意 project\_name 是自己的项目名称，需要为合法的 Python 包名，如不能为 1a 或 a-b。

**2. 新建 app**

要先进入项目目录下，cd project\_name 然后执行下面的命令（下同，已经在项目目录下则不需要 cd project\_name）

|  |
| --- |
| python manage.py startapp app\_name 或 django-admin.py startapp app\_name |

一般一个项目有多个app, 当然通用的app也可以在多个项目中使用。

与项目名类似 app name 也需要为合法的 Python 包名，如 blog，news, aboutus 等都是合法的 app 名称。

**3. 创建数据库表 或 更改数据库表或字段**

|  |
| --- |
| Django 1.7.1及以上 用以下命令 # 1. 创建更改的文件 python manage.py makemigrations # 2. 将生成的py文件应用到数据库 python manage.py migrate 旧版本的Django 1.6及以下用 python manage.py syncdb |

这种方法可以在SQL等数据库中创建与models.py代码对应的表，不需要自己手动执行SQL。

备注：对已有的 models 进行修改，Django 1.7之前的版本的Django都是无法自动更改表结构的，不过有第三方工具 south,详见 [Django 数据库迁移](https://code.ziqiangxuetang.com/django/django-data-migration.html) 一节。

**4. 使用开发服务器**

开发服务器，即开发时使用，一般修改代码后会自动重启，方便调试和开发，但是由于性能问题，建议只用来测试，不要用在生产环境。

|  |
| --- |
| python manage.py runserver # 当提示端口被占用的时候，可以用其它端口： python manage.py runserver 8001 python manage.py runserver 9999 （当然也可以kill掉占用端口的进程，具体后面有讲，此处想知道的同学可查下 lsof 命令用法） # 监听机器所有可用 ip （电脑可能有多个内网ip或多个外网ip） python manage.py runserver 0.0.0.0:8000 # 如果是外网或者局域网电脑上可以用其它电脑查看开发服务器 # 访问对应的 ip加端口，比如 http://172.16.20.2:8000 |

**5. 清空数据库**

|  |
| --- |
| python manage.py flush |

此命令会询问是 yes 还是 no, 选择 yes 会把**数据全部清空掉**，只留下空表。

**6. 创建超级管理员**

|  |
| --- |
| python manage.py createsuperuser # 按照提示输入用户名和对应的密码就好了邮箱可以留空，用户名和密码必填 # 修改 用户密码可以用： python manage.py changepassword username |

**7. 导出数据 导入数据**

|  |
| --- |
| python manage.py dumpdata appname > appname.json python manage.py loaddata appname.json |

关于数据操作 详见：[数据导入数据迁移](https://code.ziqiangxuetang.com/django/django-import-data.html)，现在了解有这个用法就可以了。

**8. Django 项目环境终端**

|  |
| --- |
| python manage.py shell |

如果你安装了 bpython 或 ipython 会自动用它们的界面，推荐安装 bpython。

这个命令和 直接运行 python 或 bpython 进入 shell 的区别是：你可以在这个 shell 里面调用当前项目的 models.py 中的 API，对于操作数据，还有一些小测试非常方便。

**9. 数据库命令行**

|  |
| --- |
| python manage.py dbshell |

Django 会自动进入在settings.py中设置的数据库，如果是 MySQL 或 postgreSQL,会要求输入数据库用户密码。

在这个终端可以执行数据库的SQL语句。如果您对SQL比较熟悉，可能喜欢这种方式。

**10. 更多命令**

终端上输入 python manage.py 可以看到详细的列表，在忘记子名称的时候特别有用。

1.编写模型类 models

2.生成模型迁移文件 python manage.py makemigrations 注意要把models注册到工程

python manage.py migrate

3.启动服务器 python manage.py runserver

4.添加django管理账户 python manage.py createsuperuser

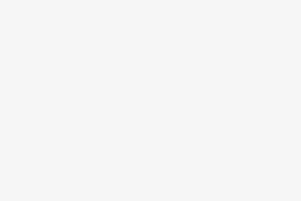
5.admin使用：在admin.py 注册，admin.site.register(模型类，admin类)

list\_display：显示字段，可以点击列头进行排序

list\_filter search\_fields list\_per\_page

fieldsets：属性分组

fieldsets = [ ('basic',{'fields': ['btitle']}), ('more', {'fields': ['bpub\_date']}), ]



开发流程：

step1:创建虚拟环境

step2：安装django包

step3：创建项目

step4：创建应用 需要将应用加入到 INSTALLED\_APPS

step5:根据表结构定义模型类 models.py 定义类属性 、

（定义admin）

step6：定义视图 视图需要接收request参数（必须）,必须返回结果

step7：配置url

step8：创建模板，模板一般放到项目路径下，建立template/projectname

**1.模型**

**1.1字段类型**

* AutoField：一个根据实际ID自动增长的IntegerField，通常不指定
  + 如果不指定，一个主键字段将自动添加到模型中
* BooleanField：true/false 字段，此字段的默认表单控制是CheckboxInput
* NullBooleanField：支持null、true、false三种值
* CharField(max\_length=字符长度)：字符串，默认的表单样式是 TextInput
* TextField：大文本字段，一般超过4000使用，默认的表单控件是Textarea
* IntegerField：整数
* DecimalField(max\_digits=None, decimal\_places=None)：使用python的Decimal实例表示的十进制浮点数
  + DecimalField.max\_digits：位数总数
  + DecimalField.decimal\_places：小数点后的数字位数
* FloatField：用Python的float实例来表示的浮点数
* DateField[auto\_now=False, auto\_now\_add=False])：使用Python的datetime.date实例表示的日期
  + 参数DateField.auto\_now：每次保存对象时，自动设置该字段为当前时间，用于"最后一次修改"的时间戳，它总是使用当前日期，默认为false
  + 参数DateField.auto\_now\_add：当对象第一次被创建时自动设置当前时间，用于创建的时间戳，它总是使用当前日期，默认为false
  + 该字段默认对应的表单控件是一个TextInput. 在管理员站点添加了一个JavaScript写的日历控件，和一个“Today"的快捷按钮，包含了一个额外的invalid\_date错误消息键
  + auto\_now\_add, auto\_now, and default 这些设置是相互排斥的，他们之间的任何组合将会发生错误的结果
* TimeField：使用Python的datetime.time实例表示的时间，参数同DateField
* DateTimeField：使用Python的datetime.datetime实例表示的日期和时间，参数同DateField
* FileField：一个上传文件的字段
* ImageField：继承了FileField的所有属性和方法，但对上传的对象进行校验，确保它是个有效的image

**1.2字段选项**

* 通过字段选项，可以实现对字段的约束
* 在字段对象时通过关键字参数指定
* null：如果为True，Django 将空值以NULL 存储到数据库中，默认值是 False
* blank：如果为True，则该字段允许为空白，默认值是 False
* **对比：null是数据库范畴的概念，blank是表单验证证范畴的**
* db\_column：字段的名称，如果未指定，则使用属性的名称
* db\_index：若值为 True, 则在表中会为此字段创建索引
* default：默认值
* primary\_key：若为 True, 则该字段会成为模型的主键字段
* unique：如果为 True, 这个字段在表中必须有唯一值

**1.3关系**

* 关系的类型包括
  + ForeignKey：一对多，将字段定义在多的端中
  + ManyToManyField：多对多，将字段定义在两端中
  + OneToOneField：一对一，将字段定义在任意一端中
* 可以维护递归的关联关系，使用'self'指定，详见“自关联”
* 用一访问多：对象.模型类小写\_set

bookinfo.heroinfo\_set

* 用一访问一：对象.模型类小写

heroinfo.bookinfo

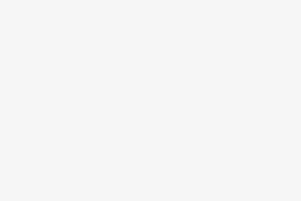
* 访问id：对象.属性\_id

heroinfo.book\_id

**1.4查询**

自定义管理器： 1， 可以更改查询级 2. 增加模型类的创建方法

管理器是模型的属性，用于对象与数据库的映射



返回单个值的方法

* get()：返回单个满足条件的对象
  + 如果未找到会引发"模型类.DoesNotExist"异常
  + 如果多条被返回，会引发"模型类.MultipleObjectsReturned"异常
* count()：返回当前查询的总条数
* first()：返回第一个对象
* last()：返回最后一个对象
* exists()：判断查询集中是否有数据，如果有则返回True

**元选项**

* 在模型类中定义类Meta，用于设置元信息
* 元信息db\_table：定义数据表名称，推荐使用小写字母，数据表的默认名称

<app\_name>\_<model\_name>

* ordering：对象的默认排序字段，获取对象的列表时使用，接收属性构成的列表

class BookInfo(models.Model): ... class Meta(): ordering = ['id']

* 字符串前加-表示倒序，不加-表示正序

class BookInfo(models.Model): ... class Meta(): ordering = ['-id']

* 排序会增加数据库的开销

**2.视图**

**2.1 URL配置**

urls.py中urlpatterns 配置： <https://blog.csdn.net/qq_40272386/article/details/78800507>

path('booktest/<int:id>',booktest\_views.show,name = 'booktest'),

基本规则：

* 使用尖括号(<>)从url中捕获值。
* 捕获值中可以包含一个转化器类型（converter type），比如使用 <int:name> 捕获一个整数变量。若果没有转化器，将匹配任何字符串，当然也包括了 / 字符。
* 无需添加前导斜杠。

Django默认支持以下5个转化器： 文档原文是Path converters，暂且翻译为转化器。

* str,匹配除了路径分隔符（/）之外的非空字符串，这是默认的形式
* int,匹配正整数，包含0。 自动转换为整型而不是字符串。
* slug,匹配字母、数字以及横杠、下划线组成的字符串。
* uuid,匹配格式化的uuid，如 075194d3-6885-417e-a8a8-6c931e272f00。
* path,匹配任何非空字符串，包含了路径分隔符

urlpatterns = [

path('articles/2003/', views.special\_case\_2003),

path('articles/<int:year>/', views.year\_archive),

path('articles/<int:year>/<int:month>/', views.month\_archive),

path('articles/<int:year>/<int:month>/<slug>/', views.article\_detail),

]

**使用正则表达式**

如果上述的paths和converters还是无法满足需求，也可以使用正则表达式，这时应当使用 django.urls.re\_path函数。

在Python正则表达式中，命名式分组语法为 (?P<name>pattern) ，其中name为名称， pattern为待匹配的模式。

之前的示例代码也可以写为:

urlpatterns = [

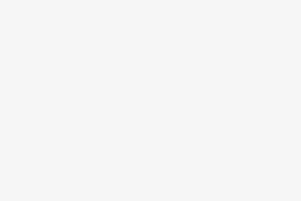
path('articles/2003/', views.special\_case\_2003),

re\_path('articles/(?P<year>[0-9]{4})/', views.year\_archive),

re\_path('articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/', views.month\_archive),

re\_path('articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>[^/]+)/', views.article\_detail),

]



**2.2 Request对象**

**属性**

* content：表示返回的内容，字符串类型
* charset：表示response采用的编码字符集，字符串类型
* status\_code：响应的HTTP响应状态码
* content-type：指定输出的MIME类型

**方法**

* init ：使用页内容实例化HttpResponse对象
* write(content)：以文件的方式写
* flush()：以文件的方式输出缓存区
* set\_cookie(key, value='', max\_age=None, expires=None)：设置Cookie
  + key、value都是字符串类型
  + max\_age是一个整数，表示在指定秒数后过期
  + expires是一个datetime或timedelta对象，会话将在这个指定的日期/时间过期，注意datetime和timedelta值只有在使用PickleSerializer时才可序列化
  + max\_age与expires二选一
  + 如果不指定过期时间，则两个星期后过期

**2.3状态保持**

存储方式包括cookie、session，会话一般指session对象

使用cookie，所有数据存储在客户端，注意不要存储敏感信息

推荐使用sesison方式，所有数据存储在服务器端，在客户端cookie中存储session\_id

* session：

**启用session**

* 使用django-admin startproject创建的项目默认启用
* 在settings.py文件中

项INSTALLED\_APPS列表中添加： 'django.contrib.sessions', 项MIDDLEWARE\_CLASSES列表中添加： 'django.contrib.sessions.middleware.SessionMiddleware',

* 禁用会话：删除上面指定的两个值，禁用会话将节省一些性能消耗

**使用session**

* 启用会话后，每个HttpRequest对象将具有一个session属性，它是一个类字典对象
* get(key, default=None)：根据键获取会话的值
* clear()：清除所有会话
* flush()：删除当前的会话数据并删除会话的Cookie
* del request.session['member\_id']：删除会话

在views.py文件中创建视图

配置url

创建模板

**会话过期时间**

* set\_expiry(value)：设置会话的超时时间
* 如果没有指定，则两个星期后过期
* 如果value是一个整数，会话将在values秒没有活动后过期
* 若果value是一个imedelta对象，会话将在当前时间加上这个指定的日期/时间过期
* 如果value为0，那么用户会话的Cookie将在用户的浏览器关闭时过期
* 如果value为None，那么会话永不过期
* 修改视图中login\_handle函数，查看效果

**存储session**

* 使用存储会话的方式，可以使用settings.py的SESSION\_ENGINE项指定
* 基于数据库的会话：这是django默认的会话存储方式，需要添加django.contrib.sessions到的INSTALLED\_APPS设置中，运行manage.py migrate在数据库中安装会话表，可显示指定为

SESSION\_ENGINE='django.contrib.sessions.backends.db'

* 基于缓存的会话：只存在本地内在中，如果丢失则不能找回，比数据库的方式读写更快

SESSION\_ENGINE='django.contrib.sessions.backends.cache'

* 可以将缓存和数据库同时使用：优先从本地缓存中获取，如果没有则从数据库中获取

SESSION\_ENGINE='django.contrib.sessions.backends.cached\_db'

**使用Redis缓存session**

* 会话还支持文件、纯cookie、Memcached、Redis等方式存储，下面演示使用redis存储
* 安装包

pip install django-redis-sessions

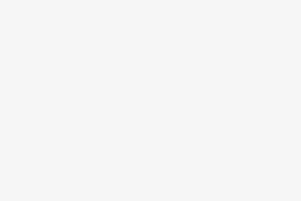
* 修改settings中的配置，增加如下项

SESSION\_ENGINE = 'redis\_sessions.session' SESSION\_REDIS\_HOST = 'localhost' SESSION\_REDIS\_PORT = 6379 SESSION\_REDIS\_DB = 0 SESSION\_REDIS\_PASSWORD = '' SESSION\_REDIS\_PREFIX = 'session'

* 管理redis的命令

启动：sudo redis-server /etc/redis/redis.conf 停止：sudo redis-server stop 重启：sudo redis-server restart redis-cli：使用客户端连接服务器 keys \*：查看所有的键 get name：获取指定键的值 del name：删除指定名称的键

session依赖cookie：



sessionid=bmjyaxu8bj9tr44jhn40wd62wp5c47es

bmjyaxu8bj9tr44jhn40wd62wp5c47es

**3.模板**

DTL:django template language

**3.1模板语言包括**

* 变量 {{ variable }} （点：1.字典查询，例如：foo["bar"] 2.属性或方法查询，例如：foo.bar 3.数字索引查询，例如：foo[bar]） ~~在模板中调用方法时不能传递参数~~
* 标签 { % 代码块 % } ： 作用：在输出中创建文本；控制循环或逻辑；加载外部信息到模板中供以后的变量使用

{% for hero in list %}

<li>{{ hero.hname }}</li>

{% endfor %}

* 过滤器
* 注释{# 代码或html #}
* for标签

{ %for ... in ...%} 循环逻辑 {{forloop.counter}}表示当前是第几次循环 { %empty%} 给出的列表为或列表不存在时，执行此处 { %endfor%}

* if标签

{ %if ...%} 逻辑1 { %elif ...%} 逻辑2 { %else%} 逻辑3 { %endif%}

* comment标签

{ % comment % } 多行注释 { % endcomment % }

* include：加载模板并以标签内的参数渲染

{ %include "foo/bar.html" % }

* url：反向解析

{ % url 'name' p1 p2 %}

* csrf\_token：这个标签用于跨站请求伪造保护

{ % csrf\_token %}

* 布尔标签：and、or，and比or的优先级高
* block、extends：详见“模板继承”
* autoescape：详见“HTML转义”

**过滤器**

* 语法：{ { 变量|过滤器 }}，例如{ { name|lower }}，表示将变量name的值变为小写输出
* 使用管道符号 (|)来应用过滤器
* 通过使用过滤器来改变变量的计算结果
* 可以在if标签中使用过滤器结合运算符

if list1|length > 1

* 过滤器能够被“串联”，构成过滤器链

name|lower|upper

* 过滤器可以传递参数，参数使用引号包起来 {% if forloop.counter|divisibleby:"2" %}

list|join:", "

* default：如果一个变量没有被提供，或者值为false或空，则使用默认值，否则使用变量的值

value|default:"什么也没有"

* date：根据给定格式对一个date变量格式化

value|date:'Y-m-d'

* escape：详见“HTML转义”
* 点击查看详细的过滤器

**注释**

* 单行注释

{#...#}

* 注释可以包含任何模版代码，有效的或者无效的都可以

{# { % if foo % }bar{ % else % } #}

* 使用comment标签注释模版中的多行内容

**示例**

* 查询所有英雄信息显示出来，要求奇数行显示为红色，偶数行显示为蓝色

**html转义：**

{{ t1|safe }} {# 过滤器关闭转义 #}

<hr>

{% autoescape off %} {# 标签关闭转义 #}

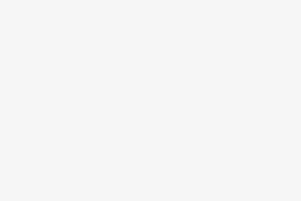
{{ t1 }}

{% endautoescape %}

**CSRF：**

**验证码**

**静态文件：动态加载路径**



注意配置url时候，结尾的 ‘/’ 一定要带上！！！

