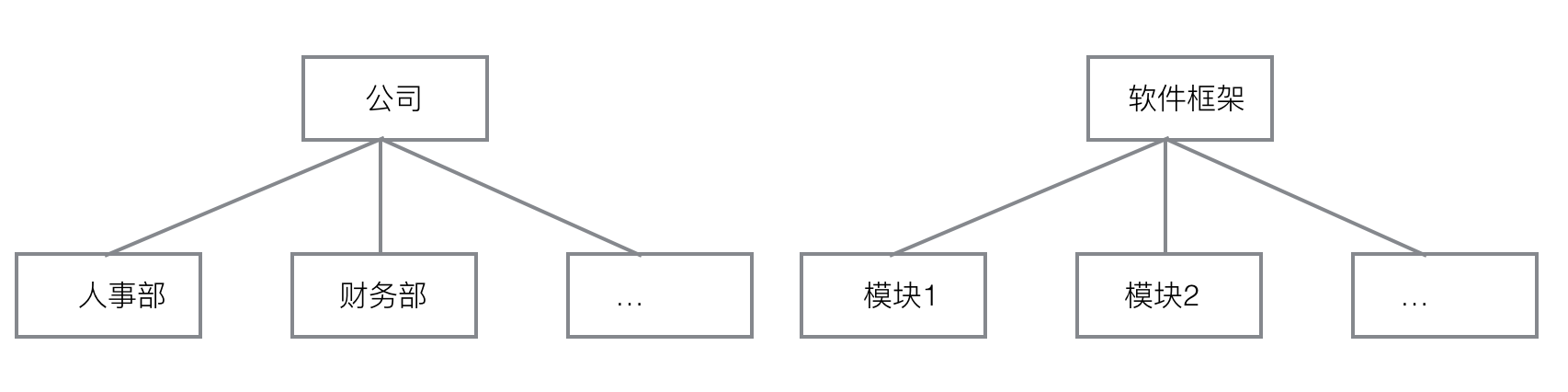
# Django

## 软件框架



一个公司是由公司中的各部部门来组成的，每一个部门拥有特定的职能，部门与部门之间通过相互的配合来完成让公司运转起来。

一个软件框架是由其中各个软件模块组成的，每一个模块都有特定的功能，模块与模块之间通过相互配合来完成软件的开发。

软件框架是针对某一类软件设计问题而产生的。

## MVC框架

### 2.1 MVC简介

MVC最初是由施乐公司旗下的帕罗奥多研究中心中的一位研究人员给 smalltalk语言发明的一中软件设计模式。

MVC的产生理念： **分工**。让专门的人去做专门的事。

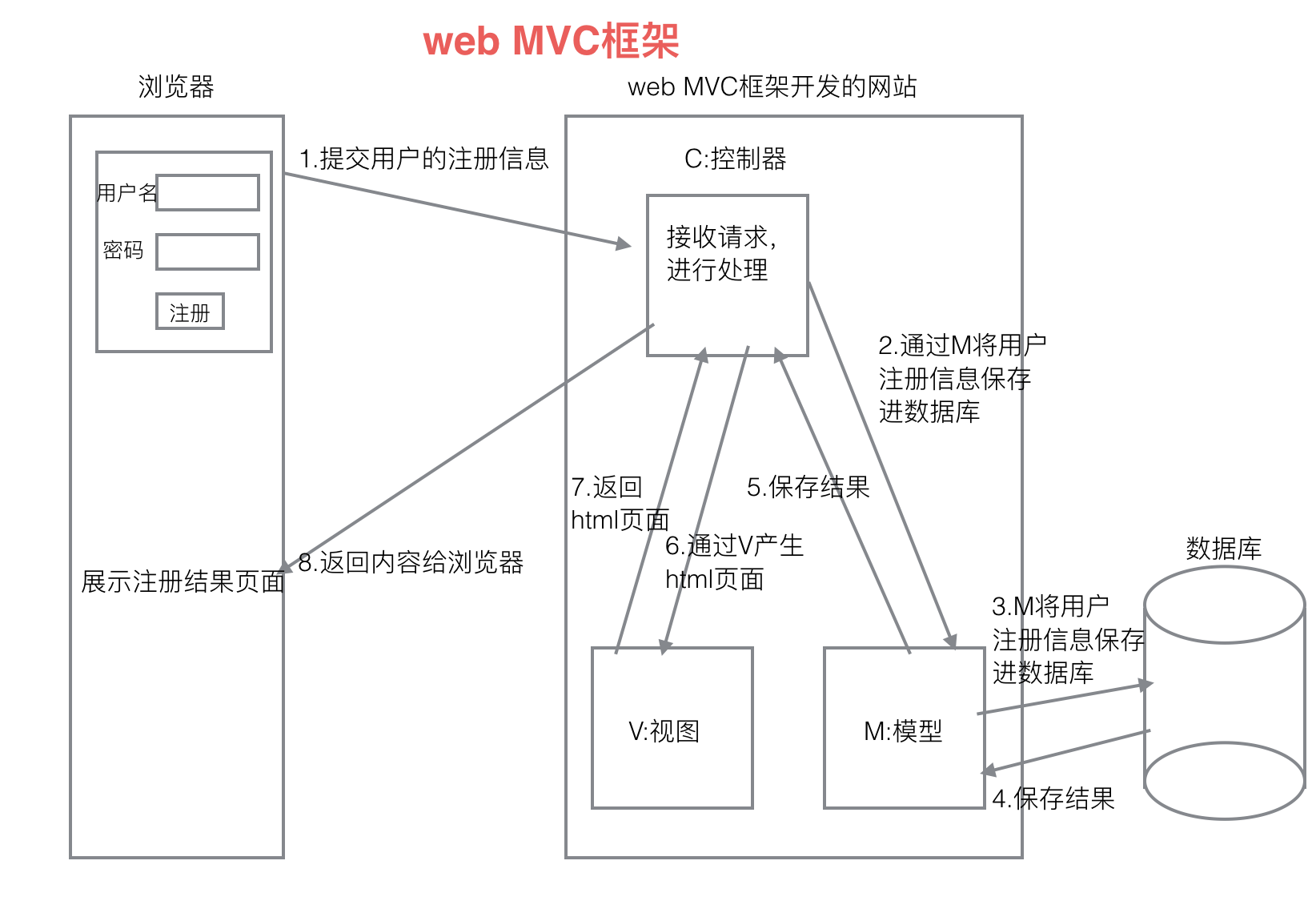
MVC的核心思想： **解耦**。

MVC的思想被应用在的web开发的方面，产生了**web MVC框架**。

### 2.2 Web MVC框架模块功能

通过浏览器注册用户信息。

****

M:Model,模型， 和数据库进行交互。

V:View,视图， 产生html页面。

C:Controller,控制器, 接收请求，进行处理，与M和V进行交互，返回应答。

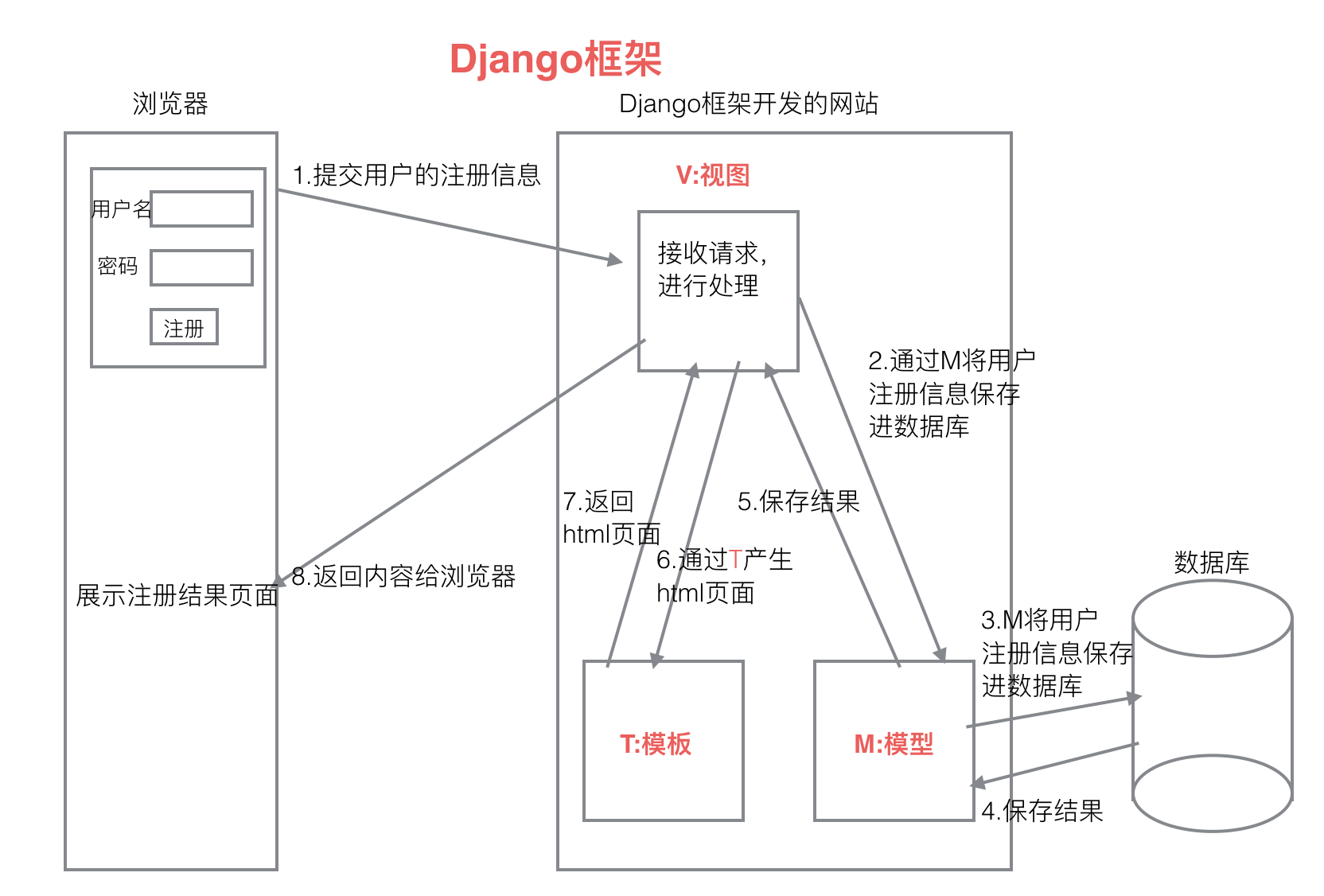
## Django框架

### 3.1 简介

Django是劳伦斯出版集团的开发人员为开发新闻内容网站而设计出来的一个软件，它遵循MVC思想，但是有自己的一个名词，叫做**MVT**。

Django遵循快速开发和DRY原则。Do not repeat yourself.不要自己去重复一些工作。

### 3.2 MVT各部分功能



M:Model,模型， 和MVC中M功能相同，和数据库进行交互。

V:View,视图， 和MVC中C功能相同，接收请求，进行处理，与M和T进行交互，返回应答。

T:Template,模板， 和MVC中V功能相同，产生html页面。

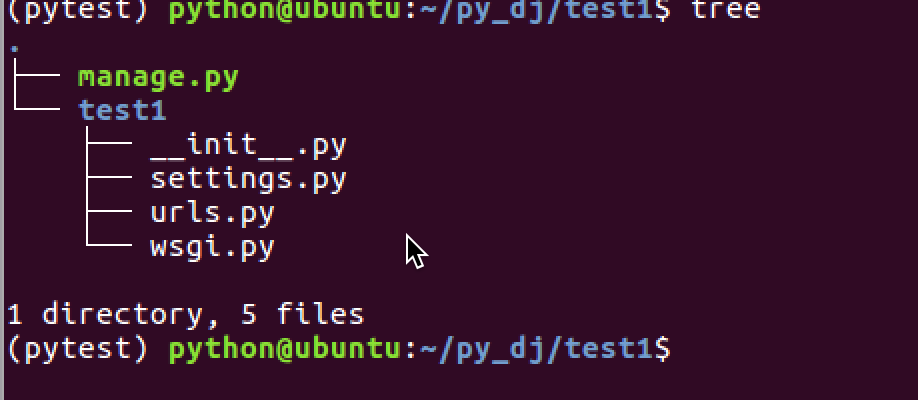
## 4．项目创建

### 4.1 创建Django项目

命令：django-admin startproject 项目名

注意：创建应用必须先进入虚拟环境。

项目目录如下：



\_\_init\_\_.py: 说明test1是一个python包。

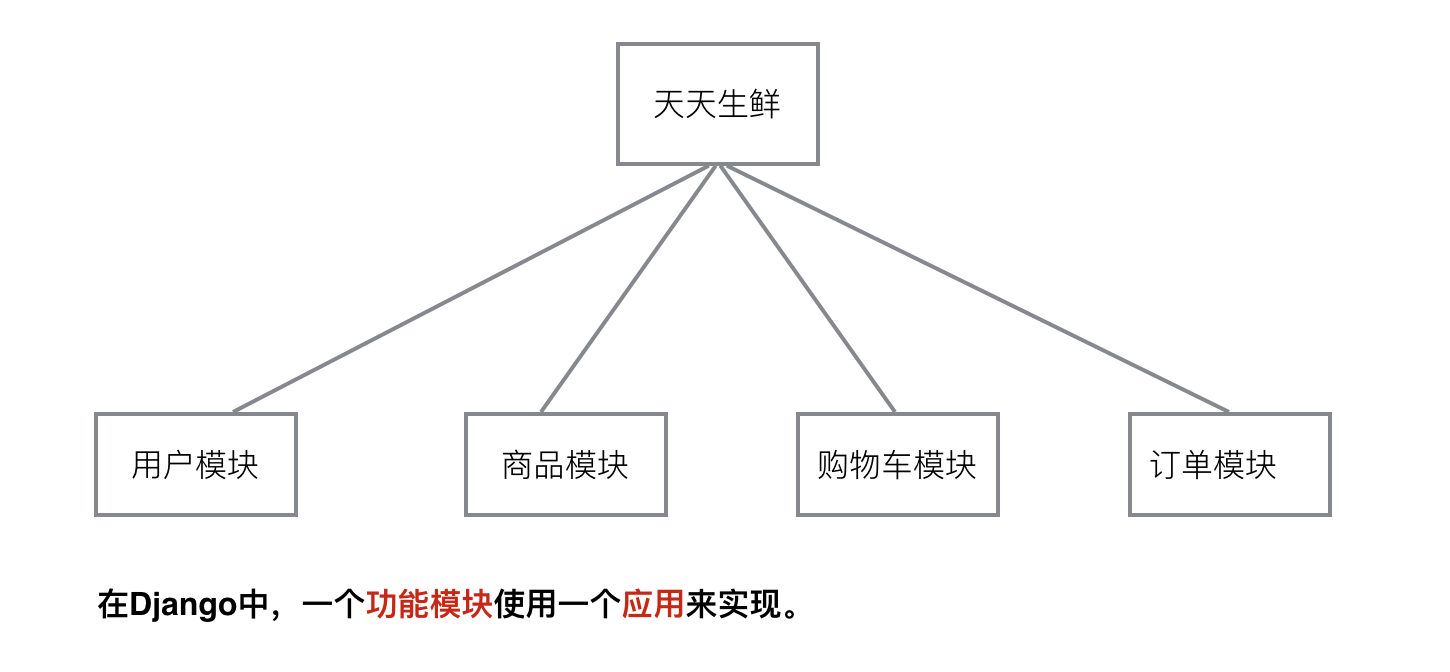
settings.py: 项目的配置文件。

urls.py: 进行url路由的配置。

wsgi.py: web服务器和Django交互的入口。

manage.py: 项目的管理文件。

### 4.2 创建Django应用



一个项目由很多个应用组成的，每一个应用完成一个功能模块。

创建应用的命令如下：

python manage.py startapp 应用名

注意：创建应用时需要先进入项目目录。

应用目录如下：



\_\_init\_\_.py: 说明目录是一个Python模块。

models.py: 写和数据库项目的内容, 设计模型类。

views.py: ，接收请求，进行处理，与M和T进行交互，返回应答。

定义处理函数，视图函数。

tests.py: 写测试代码的文件。

admin.py: 网站后台管理相关的文件。

### 4.3 应用注册

建立应用和项目之间的联系，需要对应用进行注册。

修改settings.py中的INSTALLED\_APPS配置项。

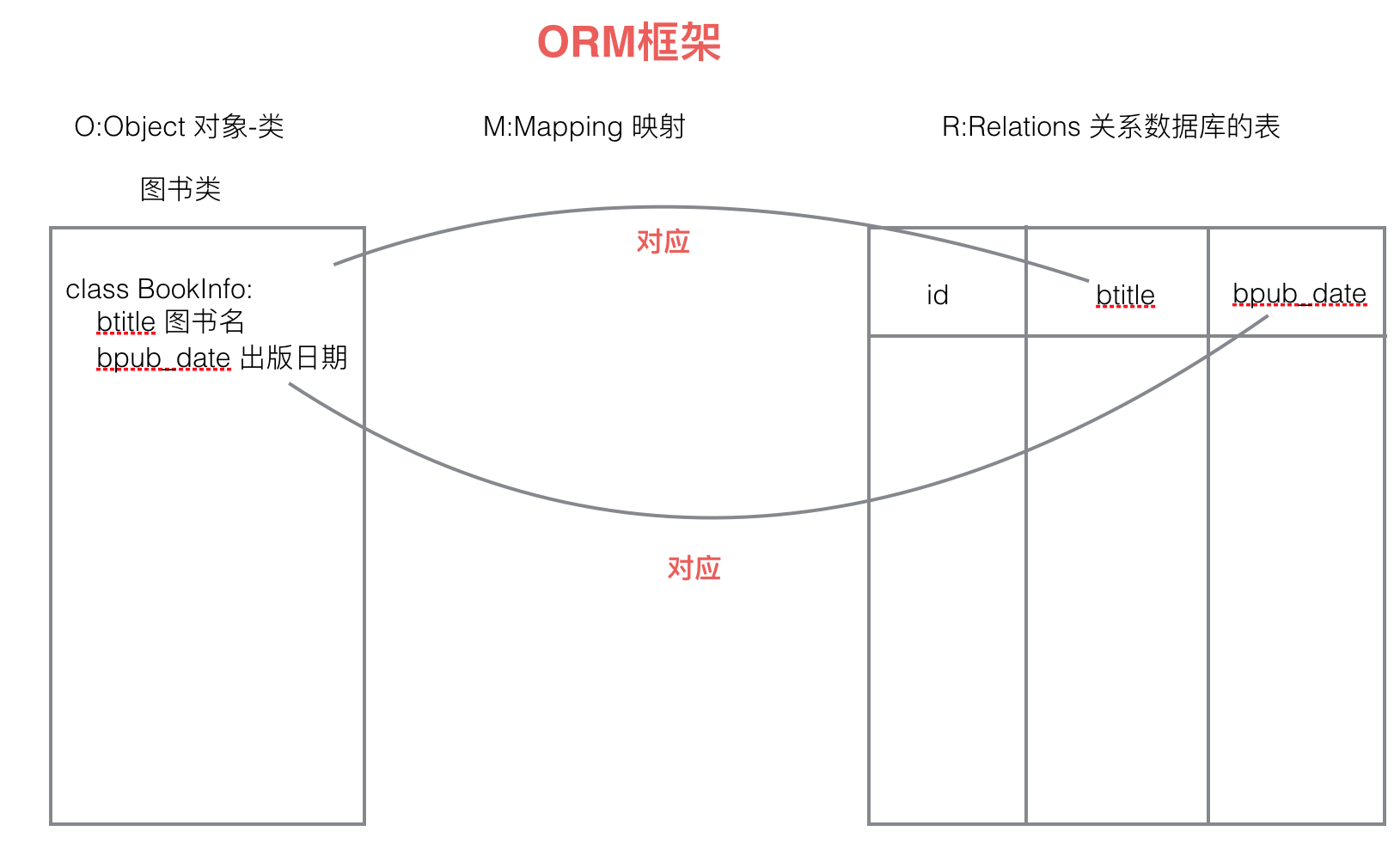
### 4.4 启动项目

运行开发web服务器命令：

python manage.py runserver

## 5.模型类

### 5.1 ORM

****

django中内嵌了ORM框架，ORM框架可以将类和数据表进行对应起来，只需要通过类和对象就可以对数据表进行操作。

在Django中主要是设计类：模型类。

ORM另外一个作用：根据设计的类生成数据库中的表。

### 5.2 模型类设计

在应用models.py中设计模型类。

必须继承与models.Model类。

1. 设计BookInfo类。
2. 设计HeroInfo类。

Models.ForeignKey可以建立两个模型类之间一对多的关系，django在生成表的时候，就会在多端的表中创建一列作为外键，建立两个表之间一对多的关系。

### 5.3 模型类生成表

1. 生成迁移文件

命令：python manage.py makemigrations



迁移文件是根据模型类生成的。

1. 执行迁移生成表

命令：python mange.py migrate

根据迁移文件生成表。

生成表名的默认格式：

应用名\_模型类名小写

### 5.4 通过模型类操作数据表

进入项目shell的命令:

python manage.py shell

以下为在相互shell终端中演示的例子：

首先导入模型类：

from booktest.models import BookInfo,HeroInfo

1. **向booktest\_bookinfo表中插入一条数据。**

b = BookInfo() #定义一个BookInfo类的对象

b.btitle ='天龙八部' #定义b对象的属性并赋值

b.bpub\_date = date(1990,10,11)

b.save() #才会将数据保存进数据库

**2) 查询出booktest\_bookinfo表中id为1的数据。**

b = BookInfo.objects.get(id=1)

**3) 在上一步的基础上改变b对应图书的出版日期。**

b.bpub\_date = date(1989,10,21)

b.save() #才会更新表格中的数据

**4) 紧接上一步，删除b对应的图书的数据。**

b.delete() #才会删除

**5) 向booktest\_heroInfo表中插入一条数据。**

h = HeroInfo()

h.hname = '郭靖'  
 h.hgender = False

h.hcomment = ‘降龙十八掌’

b2 = BookInfo.objects.get(id=2)

h.hbook = b2 #给关系属性赋值，英雄对象所属的图书对象

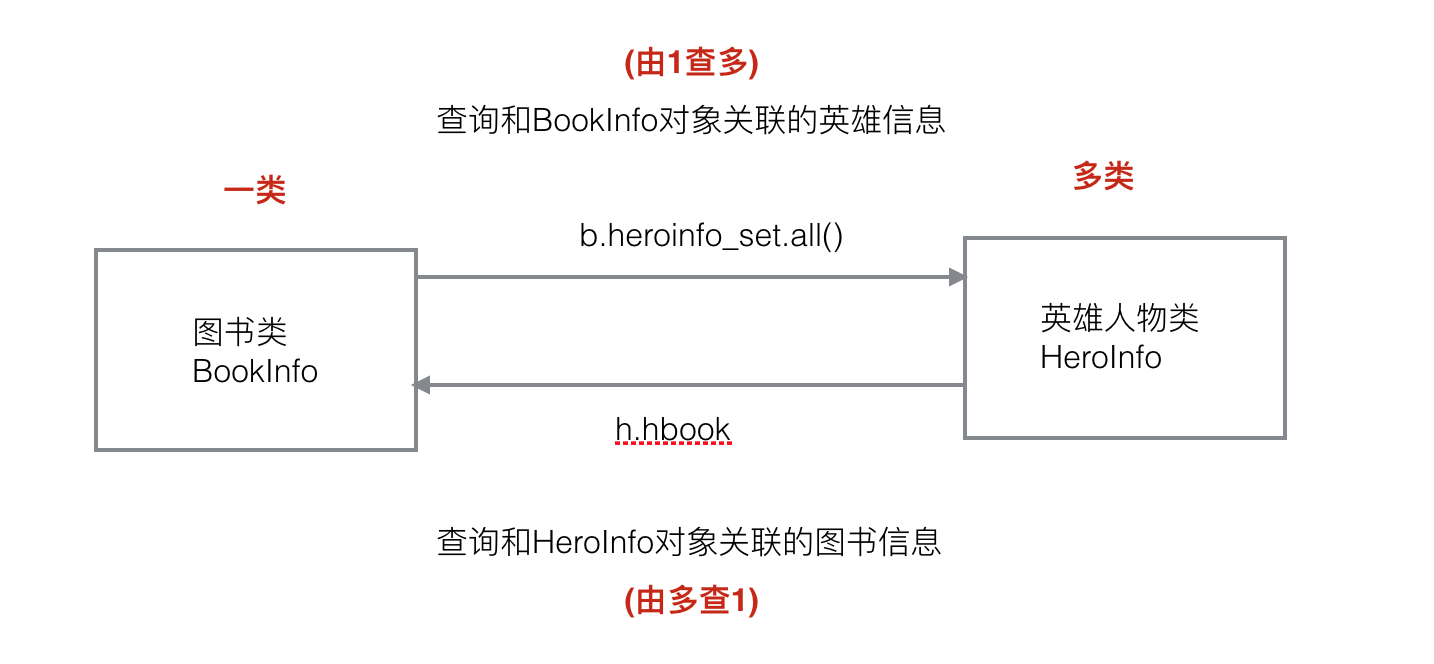
h.save()

**6) 查询图书表里面的所有内容。**

BookInfo.objects.all()

HeroInfo.objects.all()

### 5.5 关联操作

1) 查询出id为2的图书中所有英雄人物的信息。

b = BookInfo.objects.get(id=2)

b.heroinfo\_set.all() #查询出b图书中所有英雄人物的信息

## 6. 后台管理

1. 本地化

语言和时区的本地化。

修改settings.py文件。

1. 创建管理员

命令：python manage.py createsuperuser

1. 注册模型类

在应用下的admin.py中注册模型类。

告诉djang框架根据注册的模型类来生成对应表管理页面。

b = BookInfo()

str(b) \_\_str\_\_

1. 自定义管理页面

自定义模型管理类。模型管理类就是告诉django在生成的管理页面上显示哪些内容。

## 7.视图

在Django中，通过浏览器去请求一个页面时，使用视图函数来处理这个请求的，视图函数处理之后，要给浏览器返回页面内容。

### 7.1视图函数的使用

1) 定义视图函数

视图函数定义在views.py中。

例：

def index(request):

#进行处理。。。

return HttpResponse('hello python')

视图函数必须有一个参数request，进行处理之后，需要返回一个HttpResponse的类对象，hello python就是返回给浏览器显示的内容。

1. 进行url配置



url配置的目的是让建立url和视图函数的对应关系。url配置项定义在urlpatterns的列表中，每一个配置项都调用url函数。

url函数有两个参数，第一个参数是一个正则表达式，第二个是对应的处理动作。

配置url时，有两种语法格式：

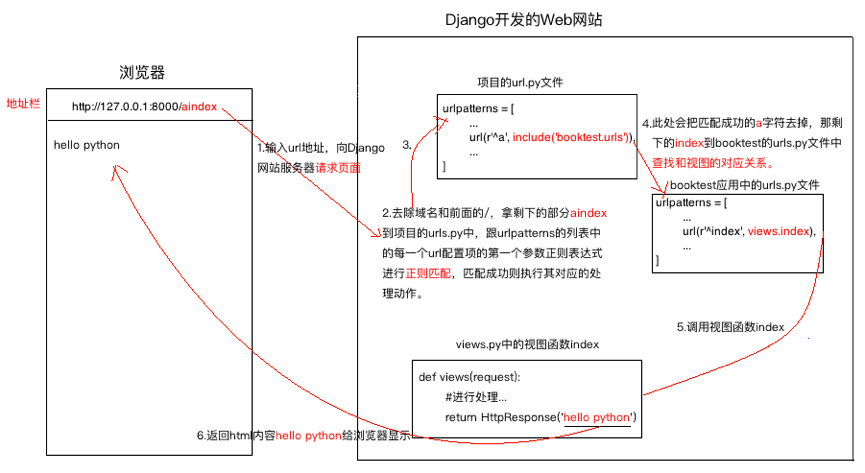
a) url(正则表达式，视图函数名)

b) url(正则表达式，include(应用中的urls文件))

工作中在配置url时，首先在项目的urls.py文件中添加配置项时，并不写具体的url和视图函数之间的对应关系，而是包含具体应用的urls.py文件，在应用的urls.py文件中写url和视图函数的对应关系。

### 7.2 url匹配的过程

在项目的urls.py文件中包含具体应用的urls.py文件，应用的urls.py文件中写url和视图函数的对应关系。



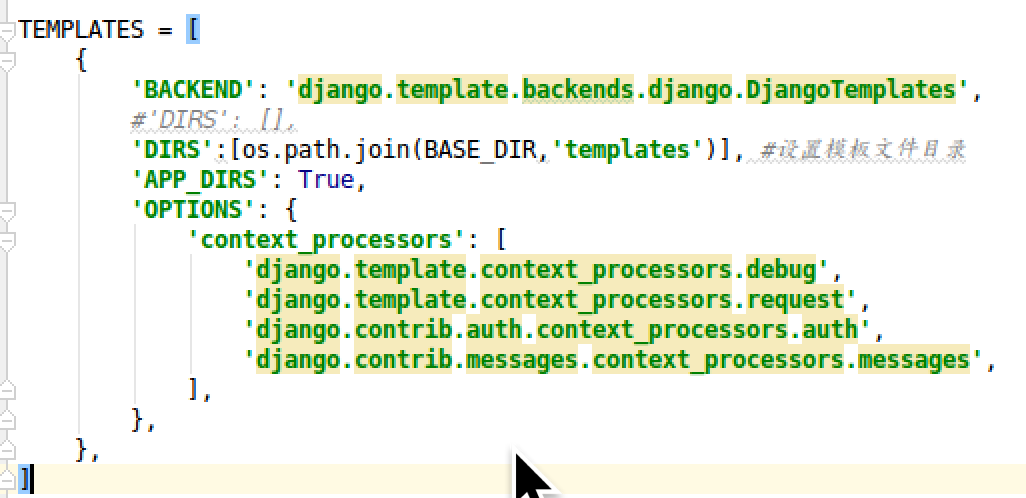
当用户输入如<http://127.0.0.1:8000/aindex>时，去除域名和最前面的/，剩下aindex，拿aindex字符串到项目的urls文件中进行匹配，配置成功之后，去除匹配的a字符，那剩下的index字符串继续到应用的urls文件中进行正则匹配，匹配成功之后执行视图函数index，index视图函数返回内容hello python给浏览器来显示。

## 模板

模板不仅仅是一个html文件。

### 模板文件的使用

1. 创建模板文件夹
2. 配置模板目录

3) 使用模板文件

1. 加载模板文件

去模板目录下面获取html文件的内容，得到一个模板对象。

1. 定义模板上下文

向模板文件传递数据。

1. 模板渲染

得到一个标准的html内容。

### 8.2 给模板文件传递数据

模板变量使用：{{ 模板变量名 }}

模板代码段：{%代码段%}

for循环：

{% for i in list %}

list不为空时执行的逻辑

{% empty %}

list为空时执行的逻辑

{% endfor %}

## 案例完成

编码之前的准备工作：

1. 设计出访问页面的url和对应的视图函数的名字，确定视图函数的功能。
2. 设计模板文件的名字。

**以下为案例中的简单设计过程：**

1. 完成图书信息的展示：
2. 设计url，通过浏览器访问 [http://127.0.0.1:8000/books 时显示图书信息页面](http://127.0.0.1:8000/books/%20时显示图书信息页面)。
3. 设计url对应的视图函数show\_books。

查询出所有图书的信息，将这些信息传递给模板文件。

1. 编写模板文件show\_books.html。

遍历显示出每一本图书的信息。

1. 完成点击某本图书时，显示出图书里所有英雄信息的页面。
2. 设计url,通过访问http://127.0.0.1:8000/books/数字时显示对应的英雄信息页面。

这里数字指点击的图书的id。

1. 设计对应的视图函数detail。

接收图书的id,根据id查询出相应的图书信息，然后查询出图书中的所有英雄信息。

1. 编写模板文件detail.html。

# 模型

## Django ORM



O(objects):类和对象。

R(Relation):关系，关系数据库中的表格。

M(Mapping):映射。

**Django ORM框架的功能：**

1. 建立模型类和表之间的对应关系，允许我们通过面向对象的方式来操作数据库。
2. 根据设计的模型类生成数据库中的表格。
3. 通过方便的配置就可以进行数据库的切换。

## Django数据库配置

### 2.1 mysql命令回顾

登录mysql数据库:mysql –uroot –p

查看有哪些数据库:show databases

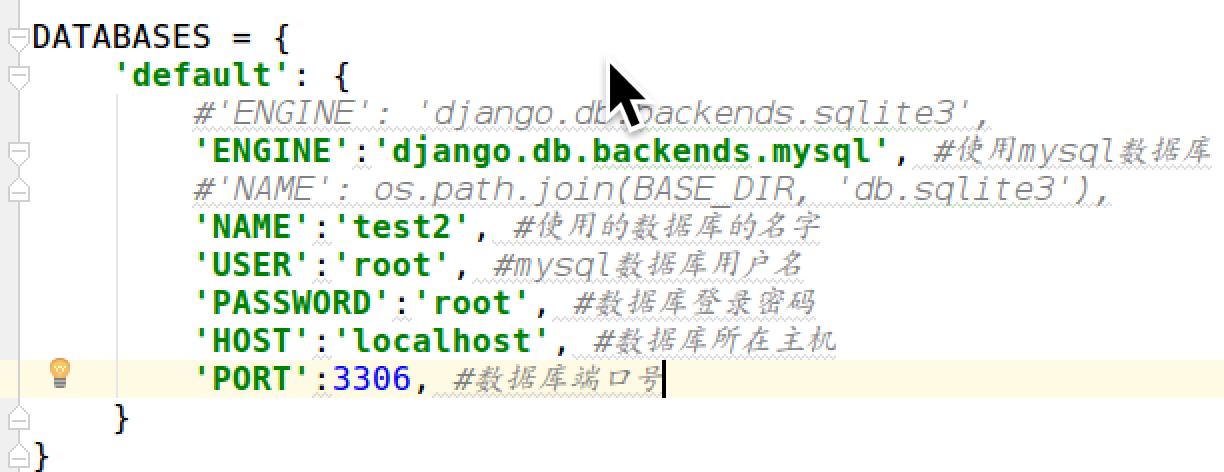
创建数据库:create database test2 **charset=utf8;** **#切记:指定编码**

使用数据库：use test2;

查看数据库中的表：show tables;

### 2.2 Django配置使用mysql数据库

修改settings.py中的DATABASES。



注意：django框架不会自动帮我们生成mysql数据库，所以我们需要自己去创建。

### 2.3 切换mysql数据库之后不能启动服务器

需要安装操作mysql数据库的包，python2环境和python3环境有以下区别。

1. **python2需要安装mysql-python:**

pip install mysql-python

1. **python3需要安装pymysql**:

pip install pymysql

python3中安装好pymysql，需要在test2/\_\_init\_\_.py中加如下内容：

import pymysql

pymysql.install\_as\_MySQLdb()

## 复习案例

1. 设计模型类并生成表
2. 设计BookInfo,增加属性bread和bcomment，另外设置软删除标记属性isDelete。
3. 设计HeroInfo类，增加软删除标记属性isDelete。

软删除标记：删除数据时不做真正的删除，而是把标记数据设置为1表示删除，目的是防止重要的数据丢失。

1. 编写视图函数并配置URL。

3）创建模板文件。

**拆解功能：**

1. 信息展示页。
2. 设计url，通过浏览器访问 <http://127.0.0.1:8000/massag时显示图书信息页面>。
3. 设计url对应的视图函数massage。

查询出所有图书的信息，将这些信息传递给模板文件。

1. 编写模板文件massage.html。

遍历显示出每一本图书的信息并增加新建和删除超链接。

## 字段属性和选项

### 4.1 模型类属性命名限制

1）不能是python的保留关键字。

2）不允许使用连续的下划线，这是由django的查询方式决定的。

3）定义属性时需要指定字段类型，通过字段类型的参数指定选项，语法如下：

属性名=models.字段类型(选项)

### 4.2 字段类型

使用时需要引入django.db.models包，字段类型如下：

|  |  |
| --- | --- |
| 类型 | 描述 |
| **AutoField** | 自动增长的IntegerField，通常不用指定，不指定时Django会自动创建属性名为id的自动增长属性。 |
| **BooleanField** | 布尔字段，值为True或False。 |
| **NullBooleanField** | 支持Null、True、False三种值。 |
| **CharField**(max\_length=最大长度) | 字符串。参数max\_length表示最大字符个数。 |
| **TextField** | 大文本字段，一般超过4000个字符时使用。 |
| **IntegerField** | 整数 |
| **DecimalField**(max\_digits=None, decimal\_places=None) | 十进制浮点数。参数max\_digits表示总位。参数decimal\_places表示小数位数。 |
| **FloatField** | 浮点数。参数同上 |
| **DateField：**([auto\_now=False, auto\_now\_add=False]) | 日期。  1)参数auto\_now表示每次保存对象时，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。  2) 参数auto\_now\_add表示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。  3)参数auto\_now\_add和auto\_now是相互排斥的，组合将会发生错误。 |
| **TimeField** | 时间，参数同DateField。 |
| **DateTimeField** | 日期时间，参数同DateField。 |
| **FileField** | 上传文件字段。 |
| **ImageField** | 继承于FileField，对上传的内容进行校验，确保是有效的图片。 |

### 4.3 选项

通过选项实现对字段的约束，选项如下：

|  |  |
| --- | --- |
| 选项名 | 描述 |
| **default** | 默认值。设置默认值。 |
| **primary\_key** | 若为True，则该字段会成为模型的主键字段，默认值是False，一般作为AutoField的选项使用。 |
| **unique** | 如果为True, 这个字段在表中必须有唯一值，默认值是False。 |
| **db\_index** | 若值为True, 则在表中会为此字段创建索引，默认值是False。 |
| **db\_column** | 字段的名称，如果未指定，则使用属性的名称。 |
| **null** | 如果为True，表示允许为空，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是后台管理页面表单验证范畴的。

**经验:**

当修改模型类之后，如果添加的选项不影响表的结构，则不需要重新做迁移，商品的选项中default和blank不影响表结构。

**参考文档:**

**http://python.usyiyi.cn/translate/django\_182/index.html**

## 查询

### 5.1 mysql的日志文件

mysql.log是mysql的日志文件，里面记录的对MySQL数据库的操作记录。默认情况下mysql的日志文件没有产生，需要修改mysql的配置文件，步骤如下：

1. 使用下面的命令打开mysql的配置文件，去除68,69行的注释，然后保存。

sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

2）重启mysql服务，就会产生mysql日志文件。

sudo service mysql restart

3）打开MySQL的日志文件。

/var/log/mysql/mysql.log 是mysql日志文件所在的位置。

使用下面的命令可以实时查看mysql的日志文件:

sudo tail -f /var/log/mysql/mysql.log

## 对象

**使用之前需要先导入**：

from django.db.models import 对象

## 模型类关系

1. 一对多关系

例：图书类-英雄类

models.ForeignKey() 定义在多的类中。

1. 多对多关系

例：新闻类-新闻类型类 体育新闻 国际新闻

models.ManyToManyField() 定义在哪个类中都可以。

1. 一对一关系

例：员工基本信息类-员工详细信息类. 员工工号

models.OneToOneField定义在哪个类中都可以。

# 视图

## 视图的功能

接收请求，进行处理，与M和T进行交互，返回应答。

返回html内容 HttpResponse，也可能重定向 redirect,还可以返回json数据。

## 视图函数使用

### 2.1 使用

1. 定义视图函数

request参数必须有。是一个HttpRequest类型的对象。参数名可以变化，

但不要更改。

1. 配置url

建立url和视图函数之间的对应关系。

### url配置的过程

1. 在项目的urls文件中包含具体应用的urls文件，在具体应用的urls文件中包含具体url和视图的对应关系。
2. url配置项是定义在一个名叫urlpatterns的列表中，其中的每一个元素就是一个配置项，每一个配置项都调用url函数。

## url匹配的过程



**url:**http://127.0.0.1:8000/aindex?a=1

1) 去除域名和后面的参数，剩下/aindex,再把前面的/去掉，剩下aindex

2) 拿aindex先到项目的url.py文件中进行从上到下的匹配，匹配成功之后执行后面对应的处理动作，就是把匹配成功的部分a字符去除，然后拿剩下的部分index到应用的urls.py文件中再进行从上到下的匹配。

3) 如果匹配成功则调用相应的视图产生内容返回给客户端。如果匹配失败则产生404错误。

## 错误视图

404：找不到页面，关闭调试模式之后，默认会显示一个标准的错误页面，如果要显示自定义的页面，则需要的templates目录下面自定义一个404.html文件。

1. url没有配置
2. url配置错误

500: 服务器端的错误。

1. 视图出错

网站开发完成需要关闭调试模式，在settings.py文件中：

DEBUG=False

ALLOWED\_HOST=[ '\*']

## 捕获url参数

进行url匹配时，把所需要的捕获的部分设置成一个正则表达式组，这样django框架就会自动把匹配成功后相应组的内容作为参数传递给视图函数。

1. 位置参数

位置参数，参数名可以随意指定

1. 关键字参数：在位置参数的基础上给正则表达式组命名即可。

?P<组名>

关键字参数，视图中参数名必须和正则表达式组名一致.

## 普通登录案例

1. **显示出登录页面**
2. 设计url，通过浏览器访问 [http://127.0.0.1:8000/login 时显示登录页面](http://127.0.0.1:8000/login/%20时显示登录页面)。
3. 设计url对应的视图函数login。
4. 编写模板文件login.html。

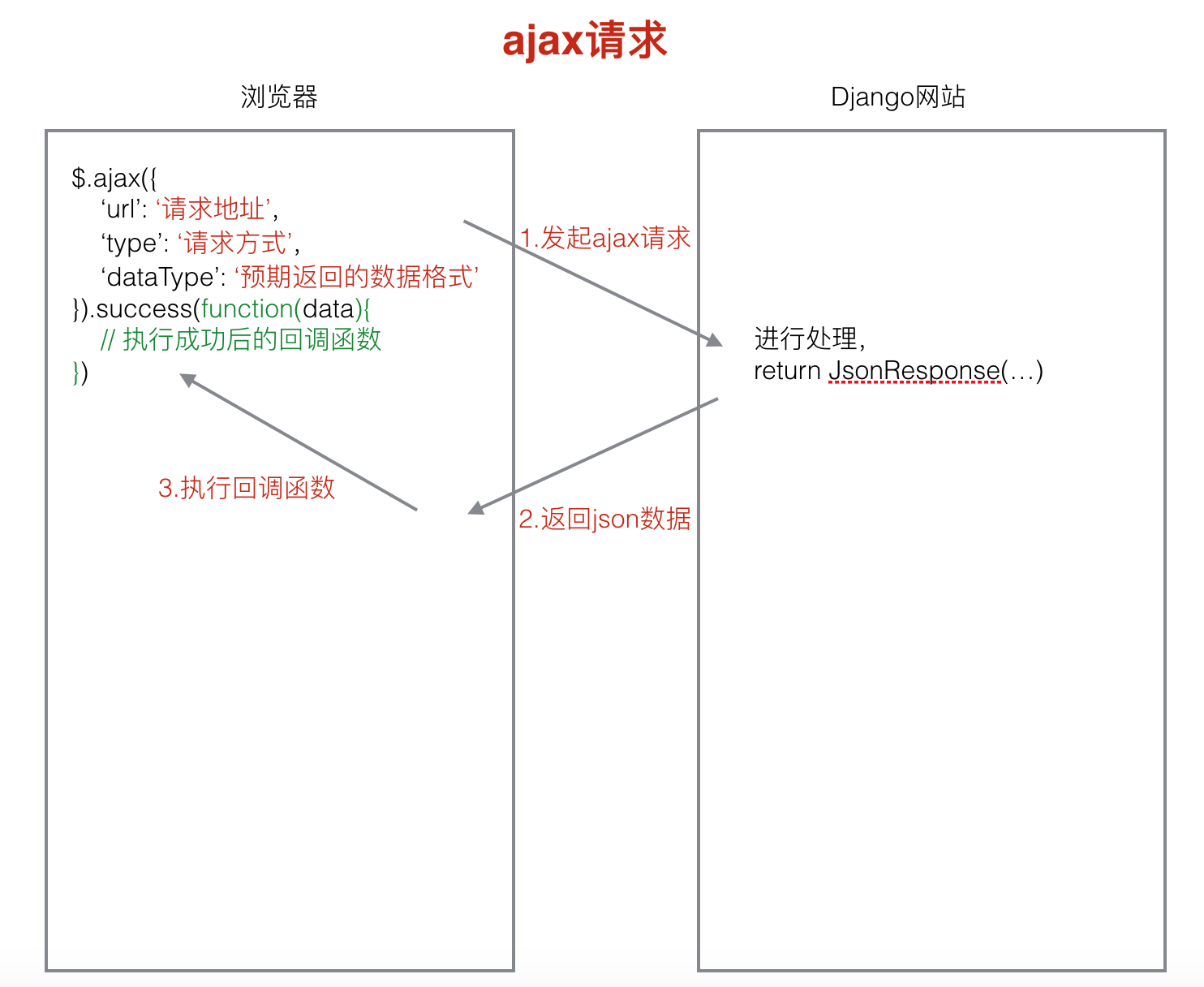
|  |  |  |
| --- | --- | --- |
| **url** | **视图** | **模板文件** |
| **/login** | **login** | **login.html** |

## Ajax

### 7.1 基本概念

异步的javascript。在不全部加载某一个页面部的情况下，对页面进行局的刷新，ajax请求都在后台。

图片，css文件，js文件都是静态文件。



1. 发起ajax请求：jquery发起
2. 执行相应的视图函数，返回json内容
3. 执行相应的回调函数。通过判断json内容，进行相应处理。

### Ajax登录案例

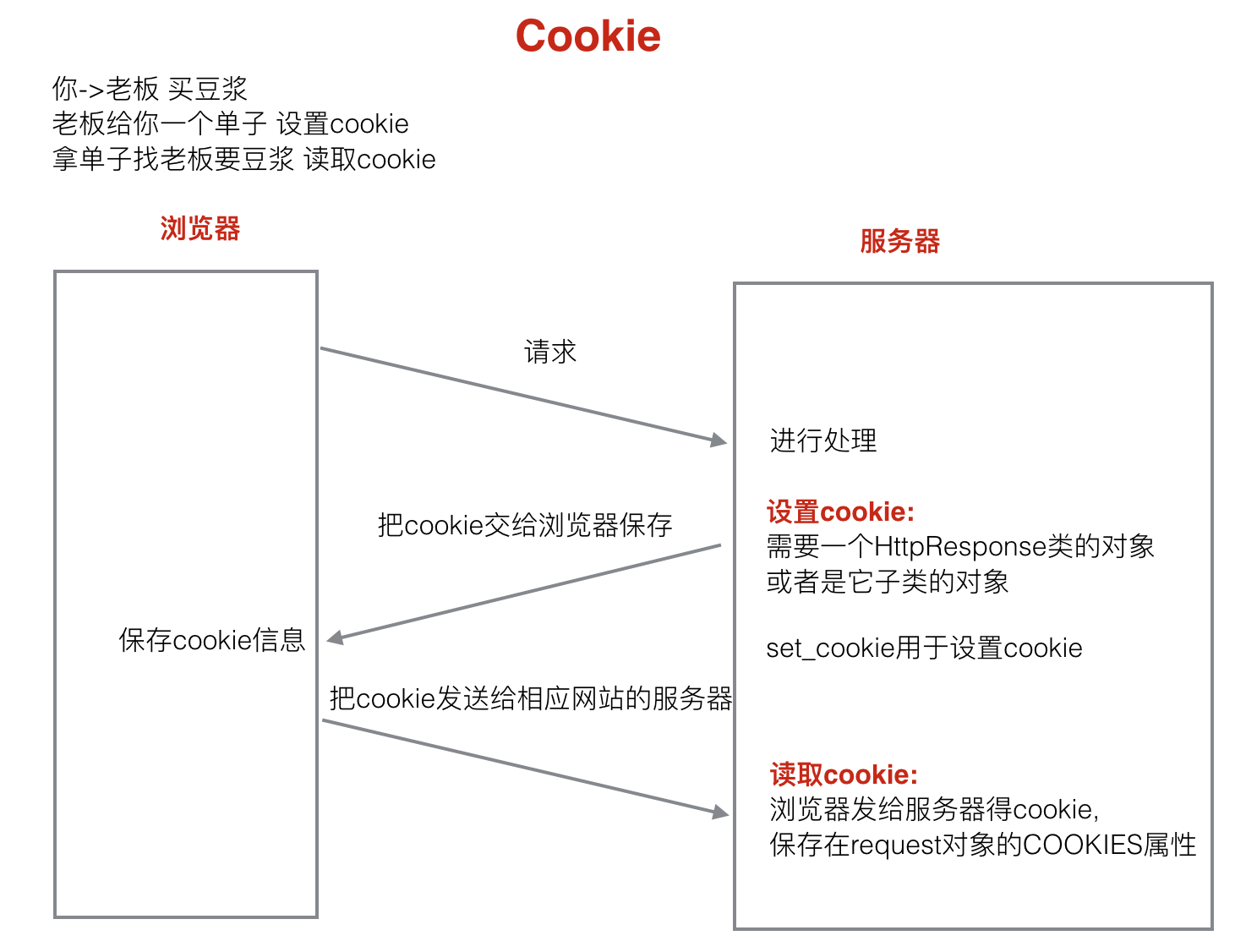
1. 首先分析出请求地址时需要携带的参数。
2. 视图函数处理完成之后，所返回的json的格式。
3. **显示出登录页面**
4. 设计url，通过浏览器访问 [http://127.0.0.1:8000/login/ 时显示登录页面](http://127.0.0.1:8000/login/%20时显示登录页面)。
5. 设计url对应的视图函数login/。
6. 编写模板文件login.html。

在里面写jquery代码发起ajax请求。

## 状态保持

http协议是无状态的。下一次去访问一个页面时并不知道上一次对这个页面做了什么。

### 8.1 Cookie



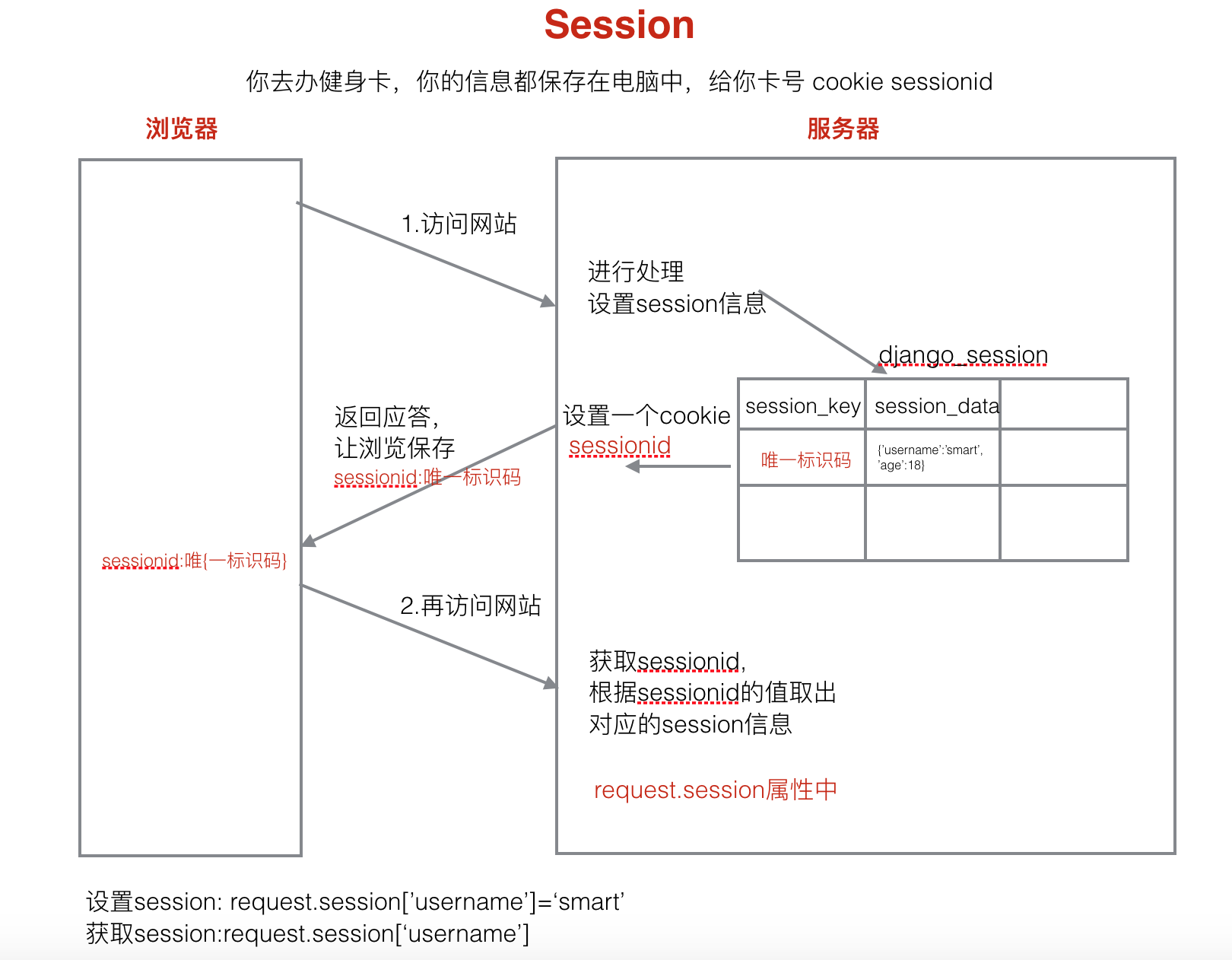
cookie是由服务器生成，存储在浏览器端的一小段文本信息。

cookie的特点：

1. 以键值对方式进行存储。
2. 通过浏览器访问一个网站时，会将浏览器存储的跟网站相关的所有cookie信息发送给该网站的服务器。request.COOKIES
3. cookie是基于域名安全的。[www.baidu.com](http://www.baidu.com) www.tudou.com
4. cookie是有过期时间的，如果不指定，默认关闭浏览器之后cookie就会过期。

**记住用户名案例。**

### 8.2 Session



session存储在服务器端。

session的特点：

1) session是以键值对进行存储的。

2) session依赖于cookie。唯一的标识码保存在sessionid cookie中。

3) session也是有过期时间，如果不指定，默认两周就会过期。

**记住用户登录状态案例。**

### 8.3 cookie和session的应用场景

cookie:记住用户名。安全性要求不高。

session:涉及到安全性要求比较高的数据。银行卡账户,密码

# 模板

## 模板的功能

产生html，控制页面上展示的内容。模板文件不仅仅是一个html文件。

模板文件包含两部分内容：

1. 静态内容：css,js,html。
2. 动态内容：用于动态去产生一些网页内容。通过模板语言来产生。

## 模板文件的使用

通常是在视图函数中使用模板产生html内容返回给客户端。

1. 加载模板文件 loader.get\_template

获取模板文件的内容，产生一个模板对象。

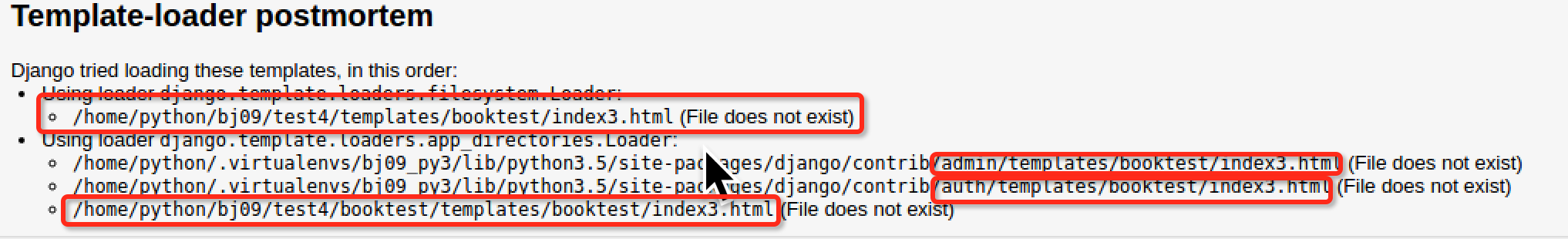
1. 定义模板上下文 RequeseContext

给模板文件传递数据。

1. 模板渲染产生html页面内容 render

用传递的数据替换相应的变量，产生一个替换后的标准的html内容。

## 模板文件加载顺序



1. 首先去配置的模板目录下面去找模板文件。
2. 去INSTALLED\_APPS下面的每个应用的templates去找模板文件，前提是应用中必须有templates文件夹。

## 模板语言

模板语言简称为DTL。（Django Template Language）

### 4.1 模板变量

模板变量名是由数字，字母，下划线和点组成的，不能以下划线开头。

使用模板变量：{{模板变量名}}

模板变量的解析顺序：

例如：{{ book.btitle }}

1. 首先把book当成一个字典，把btitle当成键名，进行取值book['btitle']
2. 把book当成一个对象，把btitle当成属性，进行取值book.btitle
3. 把book当成一个对象，把btitle当成对象的方法，进行取值book.btitle

例如：{{book.0}}

1. 首先把book当成一个字典，把0当成键名，进行取值book[0]
2. 把book当成一个列表，把0当成下标，进行取值book[0]

如果解析失败，则产生内容时用空字符串填充模板变量。

**使用模板变量时，.前面的可能是一个字典，可能是一个对象，还可能是一个列表。**

### 4.2 模板标签

{% 代码段 %}

for循环：

{% for x in 列表 %}

# 列表不为空时执行

{% empty %}

# 列表为空时执行

{% endfor %}

可以通过**{{ forloop.counter }}**得到for循环遍历到了第几次。

{% if 条件 %}

{% elif 条件 %}

{% else %}

{% endif %}

关系比较操作符：> < >= <= == !=

注意：进行比较操作时，比较操作符两边必须有空格。

逻辑运算：not and or

### 4.3 过滤器

过滤器用于对模板变量进行操作。

date：改变日期的显示格式。

length:求长度。字符串，列表.

default:设置模板变量的默认值。

格式：模板变量｜过滤器：参数

自定义过滤器。

**自定义的过滤器函数，至少有一个参数，最多两个**

**参考资料：(模板标签和内置过滤器)**

http://python.usyiyi.cn/documents/django\_182/ref/templates/builtins.html

### 4.4 模板注释

单行注释：{# 注释内容 #}

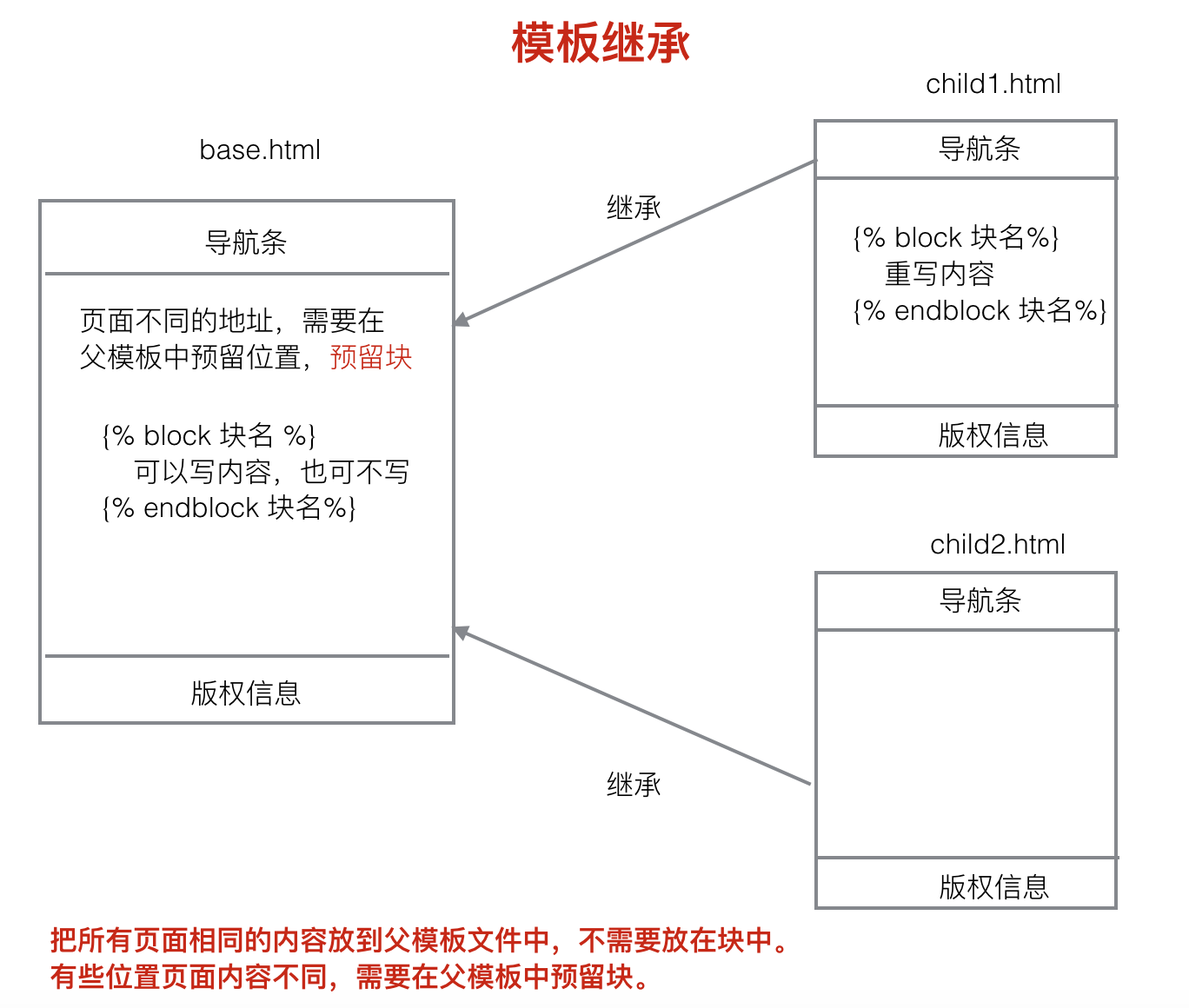
多行注释：{% comment %}

注释内容

{% endcomment %}

## 模板继承

模板继承也是为了重用html页面内容。



**在父模板里可以定义块，使用标签：**

{% block 块名 %}

块中间可以写内容，也可以不写

{% endblock 块名%}

子模板去继承父模板之后，可以重写父模板中的某一块的内容。

**继承格式：{% extends 父模板文件路径%}**

{% block 块名 %}

{{ block.super}} #获取父模板中块的默认内容

重写的内容

{% endblock 块名%}

# 其他技术

## 静态文件

### 1.1 使用

在 网页使用的css文件，js文件和图片叫做静态文件。

1. 新建静态文件夹 static。



2) 配置静态文件所在的物理目录。Settings.py



STATIC\_URL设置访问静态文件对应的url。

STATICFILES\_DIRS设置静态文件所在的物理目录。

**动态生成静态文件的路径。**

### 1.2 加载目录

STATICFILES\_FINDERS=('django.contrib.staticfiles.finders.FileSystemFinder', 'django.contrib.staticfiles.finders.AppDirectoriesFinder')

## Admin后台管理

### 3.1 使用

1) 本地化。语言和时区本地化。

2) 创建超级管理员。

python mange.py createsuperuser

3) 注册模型类。

4) 自定义管理页面。

自定义模型管理类。

注册模型类的时候给register函数添加第二个参数，就是自定义模型管理类的名字。

### 3.2 模型管理类相关属性

1) 列表页相关的选项。

2) 编辑页相关的选项。