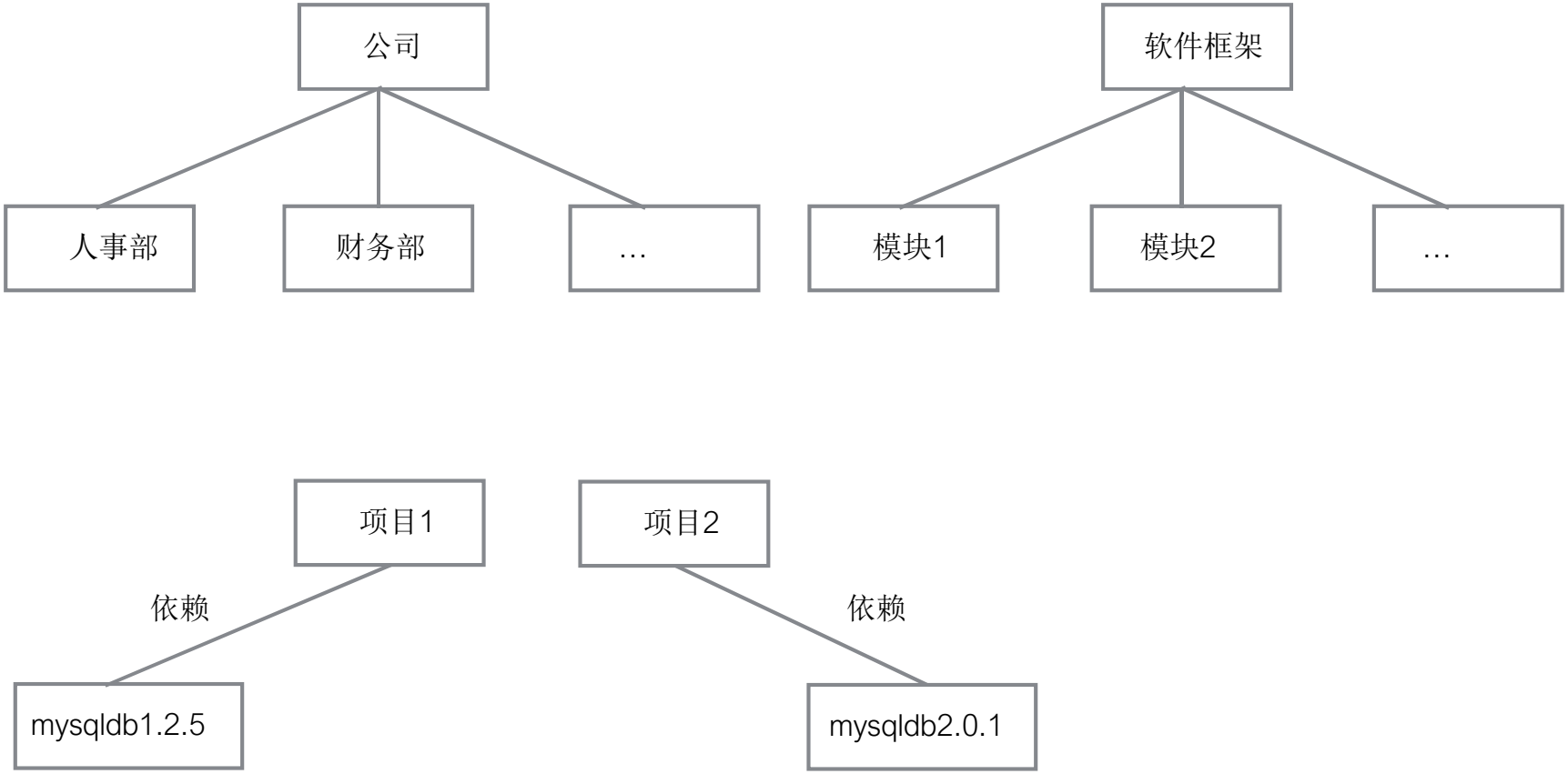
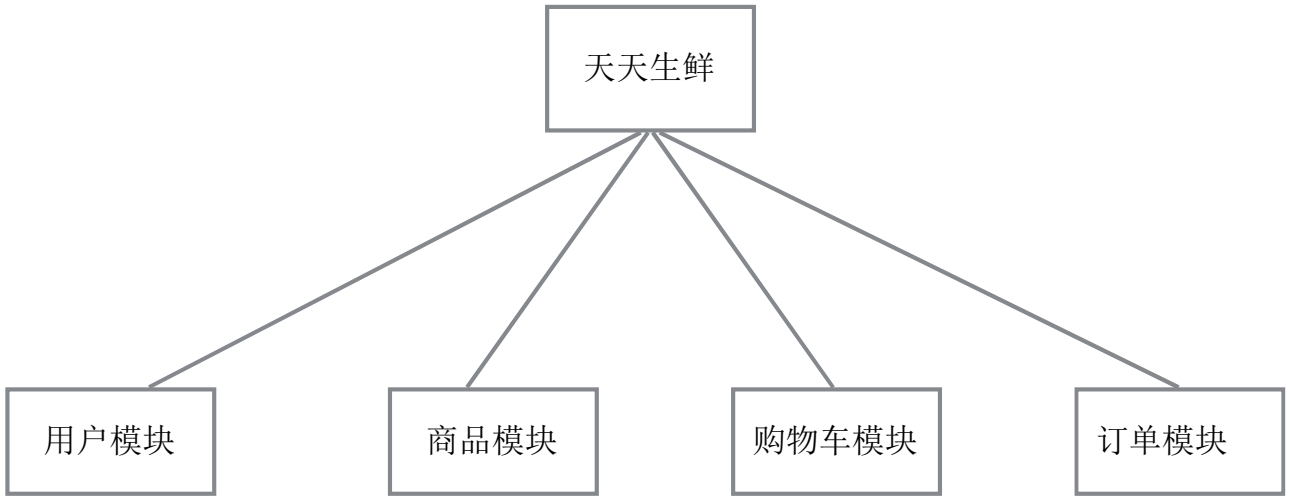
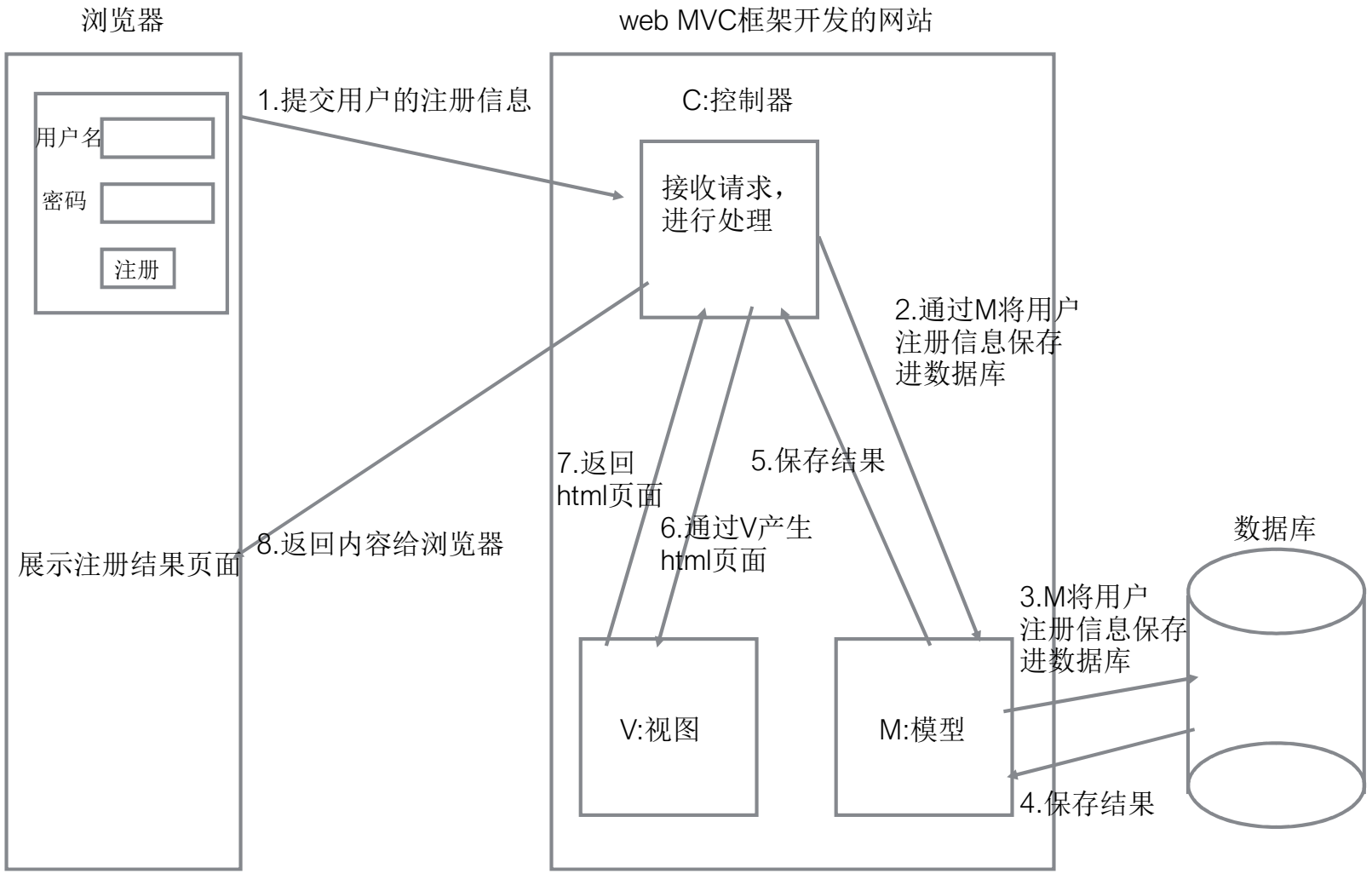

公司
人事部
财务部
…
软件框架
模块1
模块2
…
项目1
依赖
mysqldb1.2.5
项目2
依赖
mysqldb2.0.1

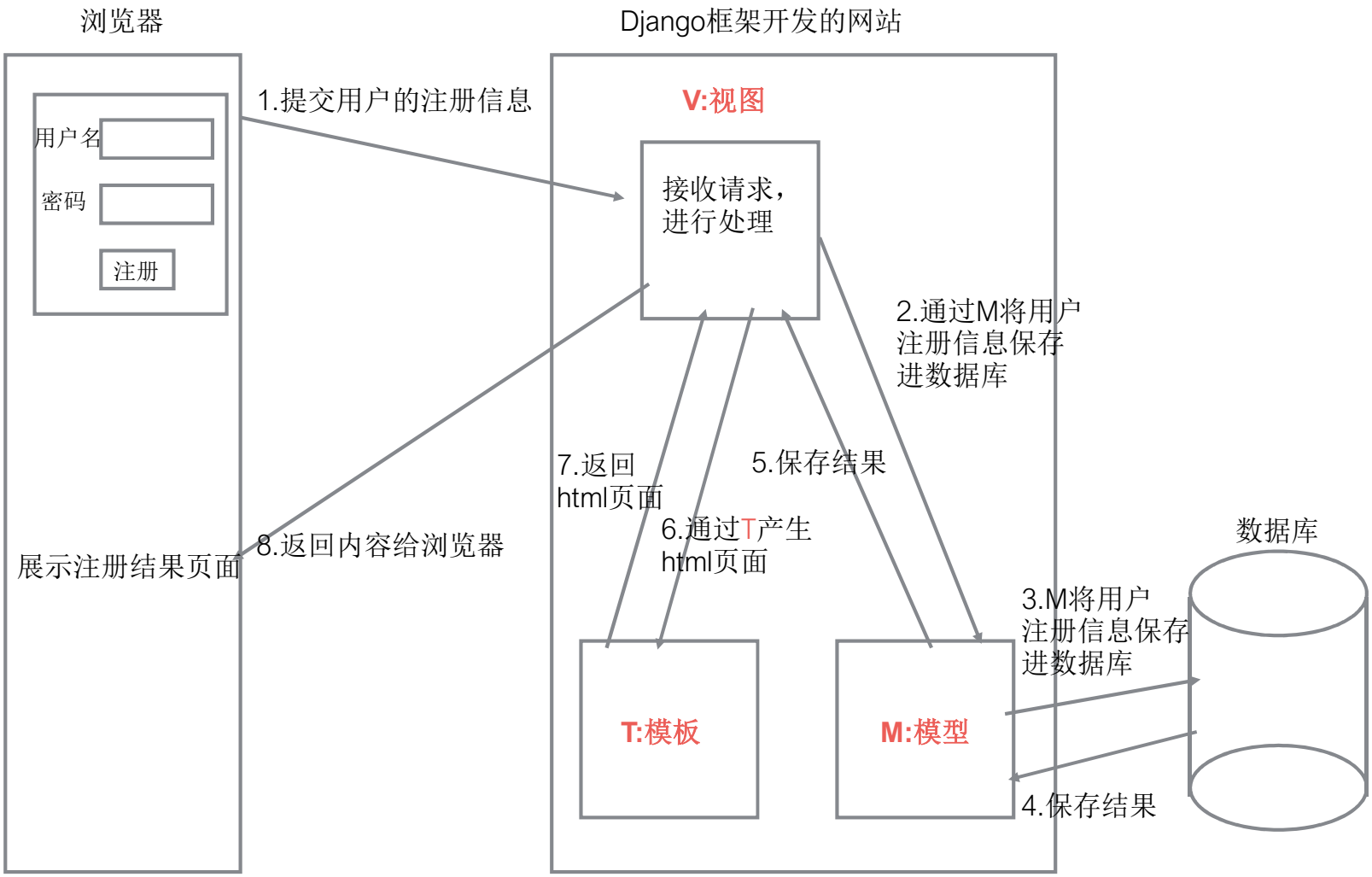


在Django中，一个功能模块使用一个应用来实现。

web MVC框架


浏览器
web MVC框架开发的网站
用户名
密码
注册
1.提交用户的注册信息
C:控制器
接收请求，
进行处理
2.通过M将用户
注册信息保存
进数据库
7.返回
html页面
5.保存结果
6.通过V产生
html页面
数据库
展示注册结果页面
8.返回内容给浏览器
3.M将用户
注册信息保存
进数据库
V:视图
M:模型
4.保存结果

Django框架


浏览器
Django框架开发的网站
用户名
密码
注册
1.提交用户的注册信息
V:视图
接收请求，
进行处理
2.通过M将用户
注册信息保存
进数据库
7.返回
html页面
5.保存结果
6.通过T产生
html页面
8.返回内容给浏览器
展示注册结果页面
数据库
3.M将用户
注册信息保存
进数据库
T:模板
M:模型
4.保存结果

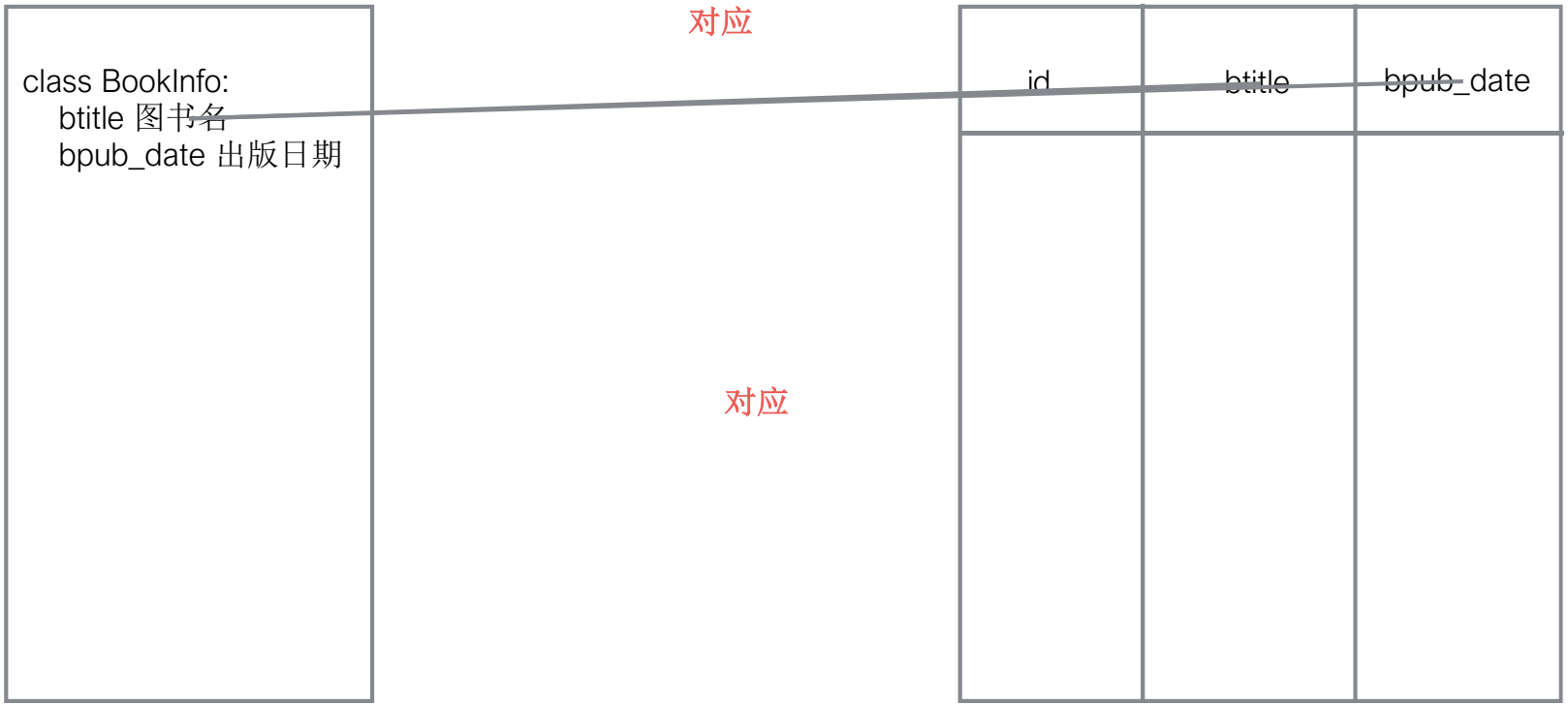
ORM框架

O:Object 对象-类

M:Mapping 映射

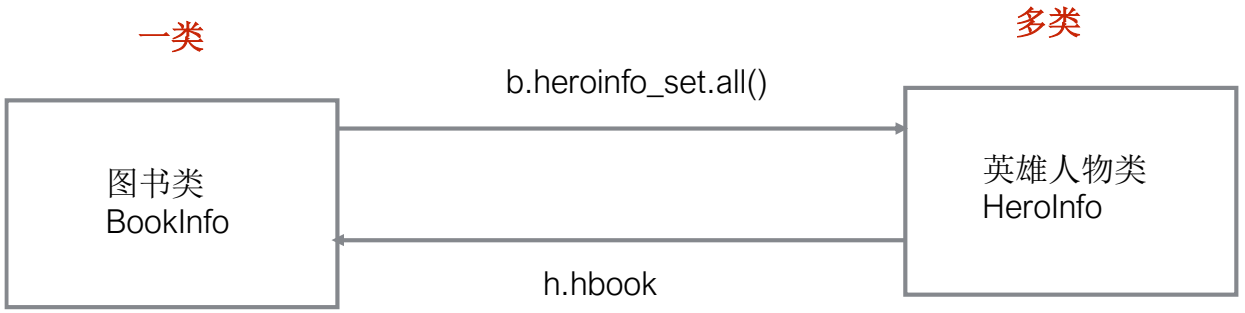
R:Relations 关系数据库的表

图书类



(由1查多)

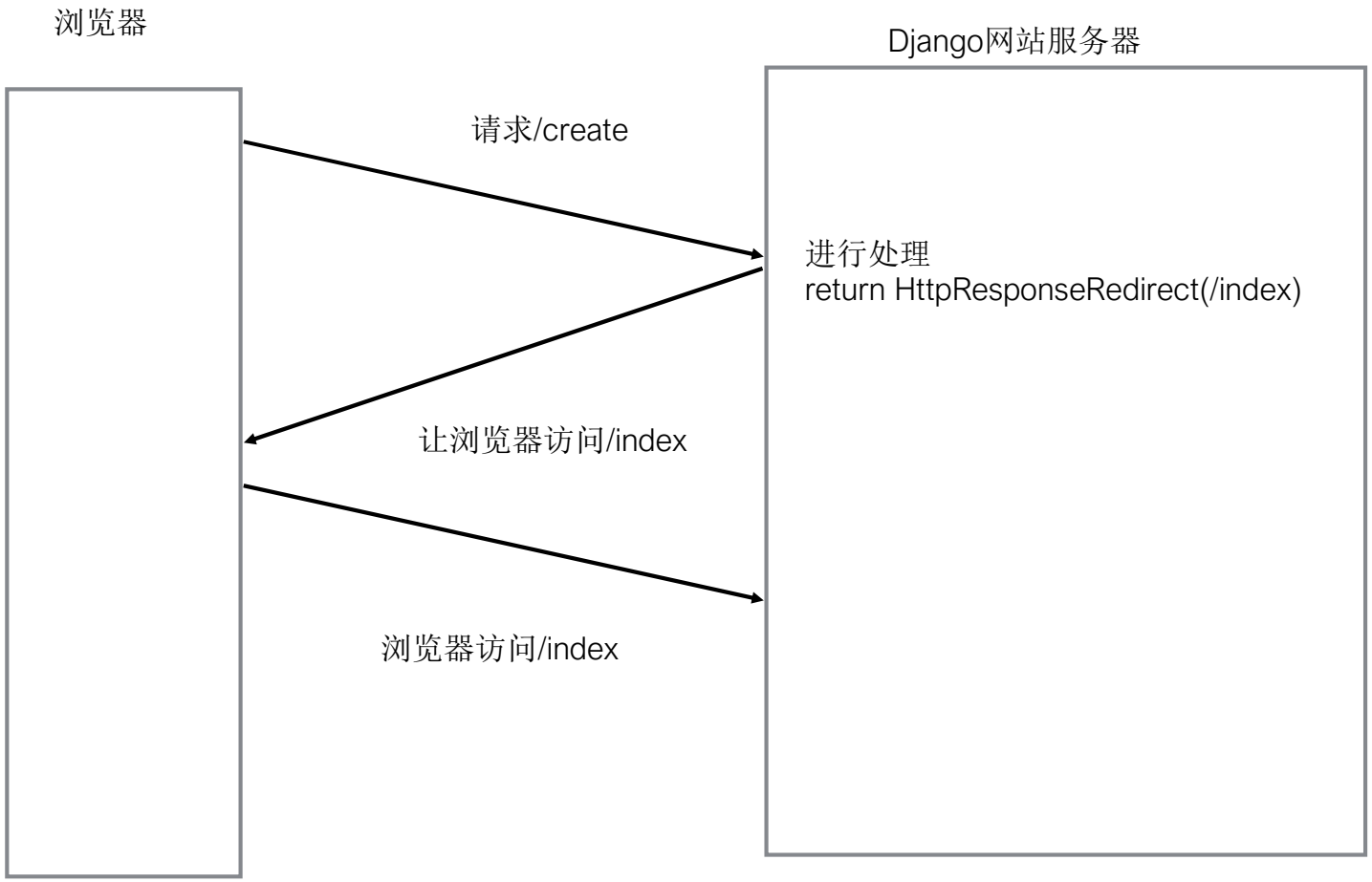
查询和BookInfo对象关联的英雄信息



查询和HeroInfo对象关联的图书信息

(由多查1)

重定向


浏览器
Django网站服务器
请求/create
进行处理
return HttpResponseRedirect(/index)
让浏览器访问/index
浏览器访问/index

查询相关函数

get:返回一条且只能有一条数据，返回值是一个对象，参数可以写查询条件。
all:返回模型类对应表的所有数据，返回值是QuerySet。
filter:返回满足条件的数据，返回值是QuerySet，参数可以写查询条件。
exclude:返回不满足条件的数据，返回值是QuerySet，参数可以写查询条件。
order_by:对查询结果进行排序，返回值是QuerySet，参数中写排序的字段。

注意：

1.get, filter,exclude函数中可以写查询条件，如果传多个参数，条件之间代表且的关系。
2.all，filter，exclude，order_by函数的返回值是QuerySet类的实例对象，叫做查询集。

from django.db.models import F,Q,Sum,Count,Avg,Max,Min

F对象: 用于类属性之间的比较。
Q对象: 用于条件之间的逻辑关系。

aggregate:进行聚合操作，返回值是一个字典，进行聚合的时候需要先导入聚合类。
count:返回结果集中数据的数目，返回值是一个数字。

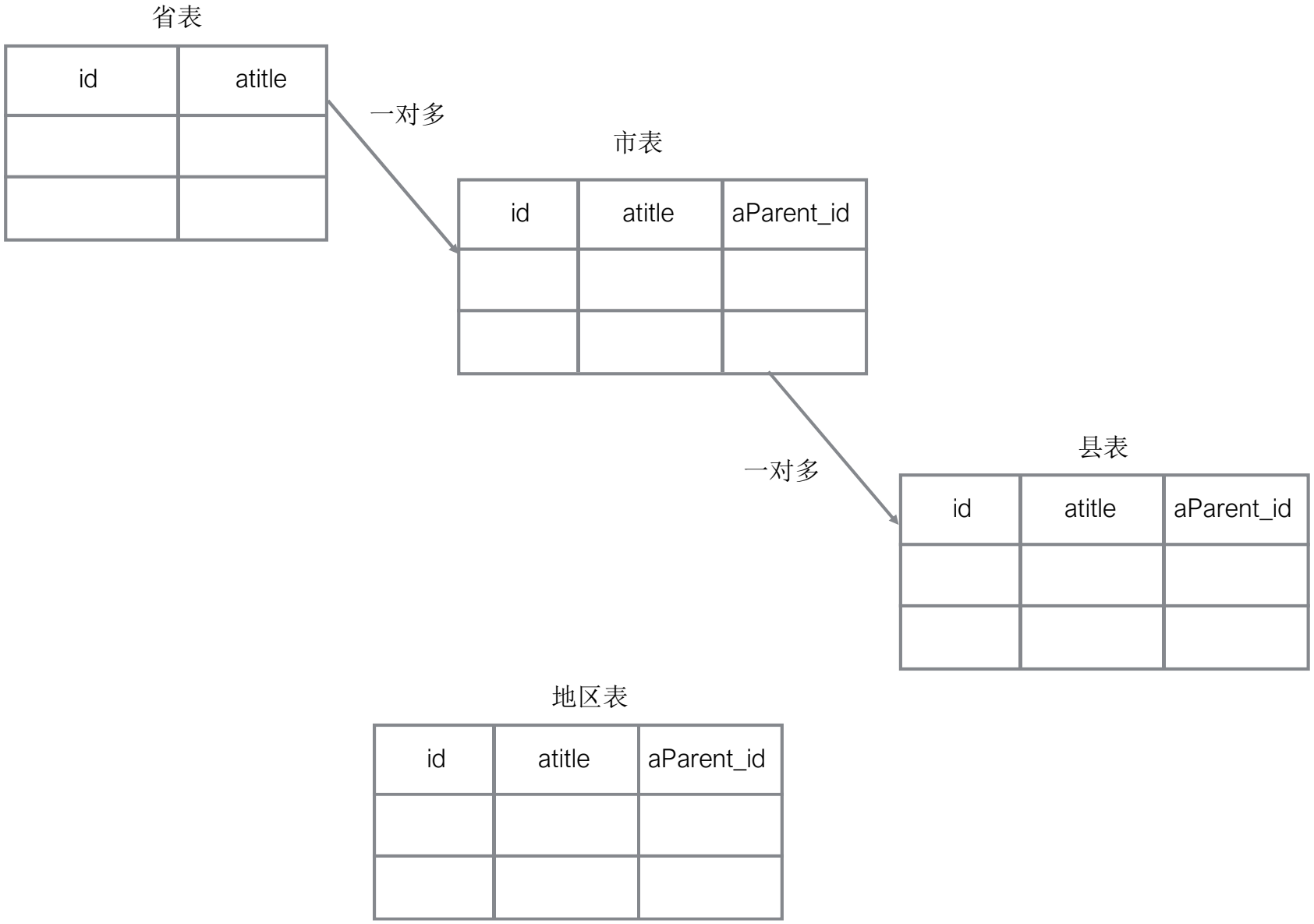
注意:

对一个QuerySet实例对象，可以继续调用上面的所有函数。

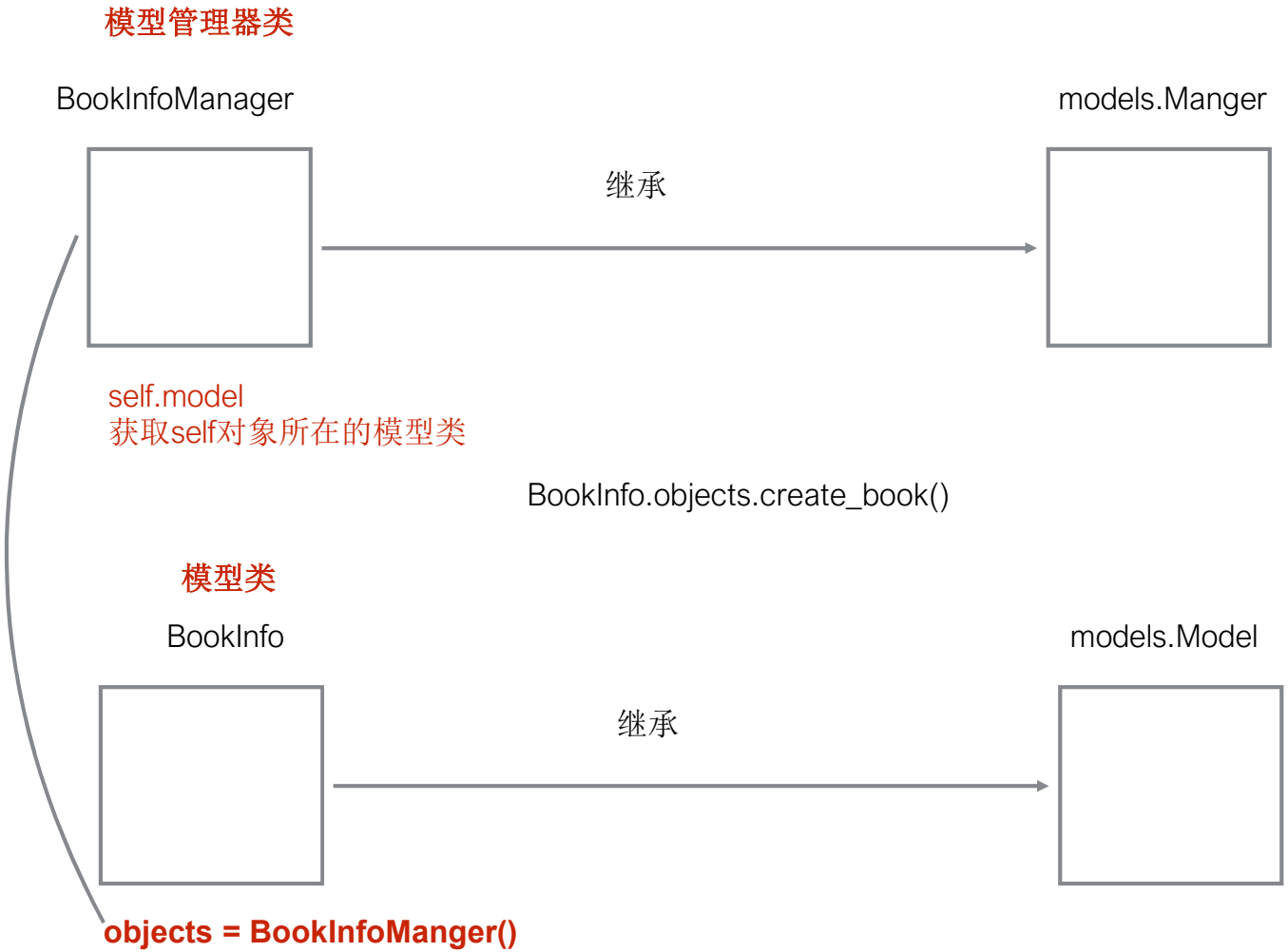
关联查询

注意：

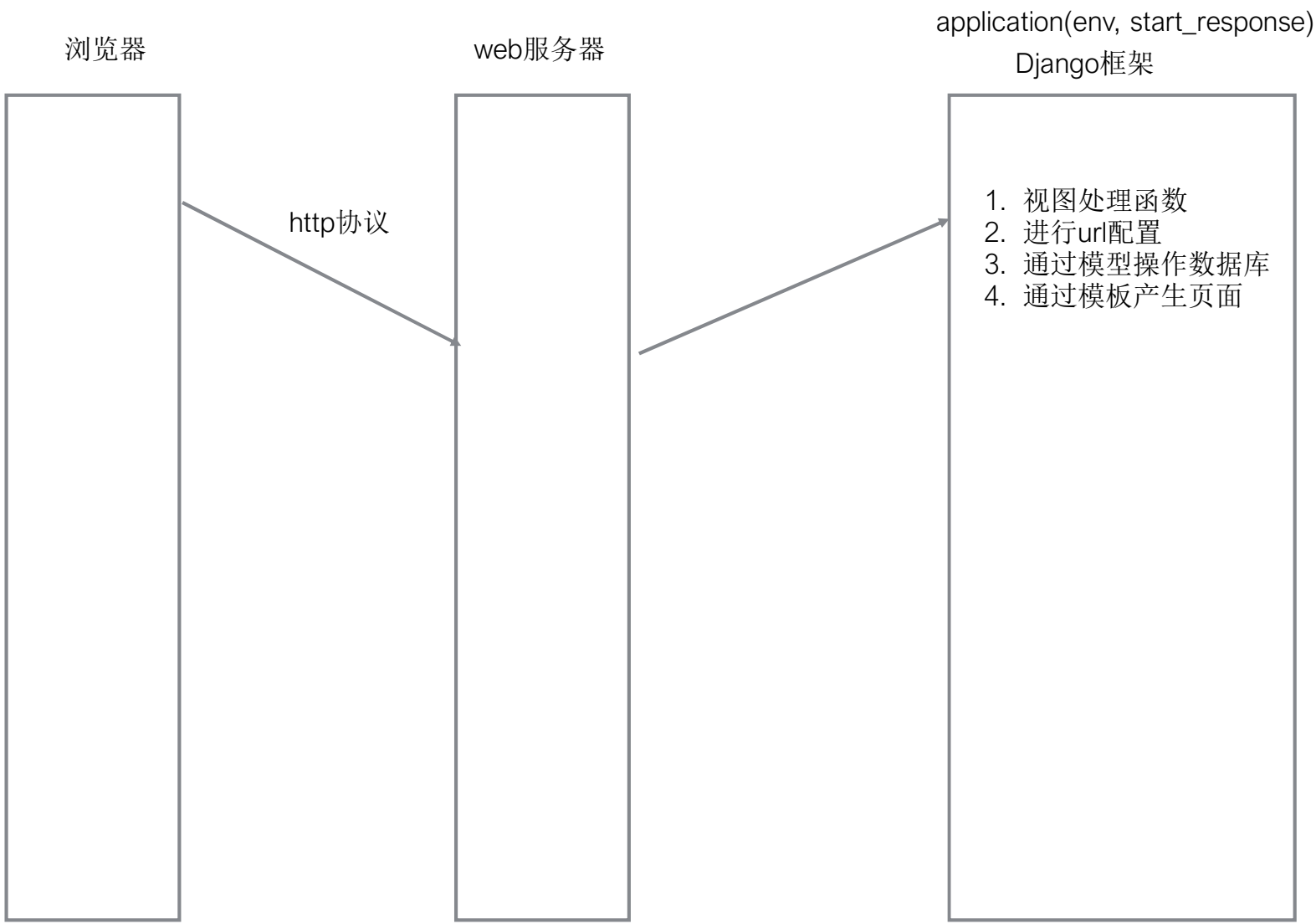
1. 通过模型类实现关联查询时，要查哪个表中的数据，就需要通过哪个类来查。
2. 写关联查询条件的时候，如果类中没有关系属性，条件需要些对应类的名，
如果类中有关系属性，直接写关系属性。


省表
id
atitle
一对多
市表
id
atitle
aParent_id
一对多
县表
id
atitle
aParent_id
地区表
id
atitle
aParent_id

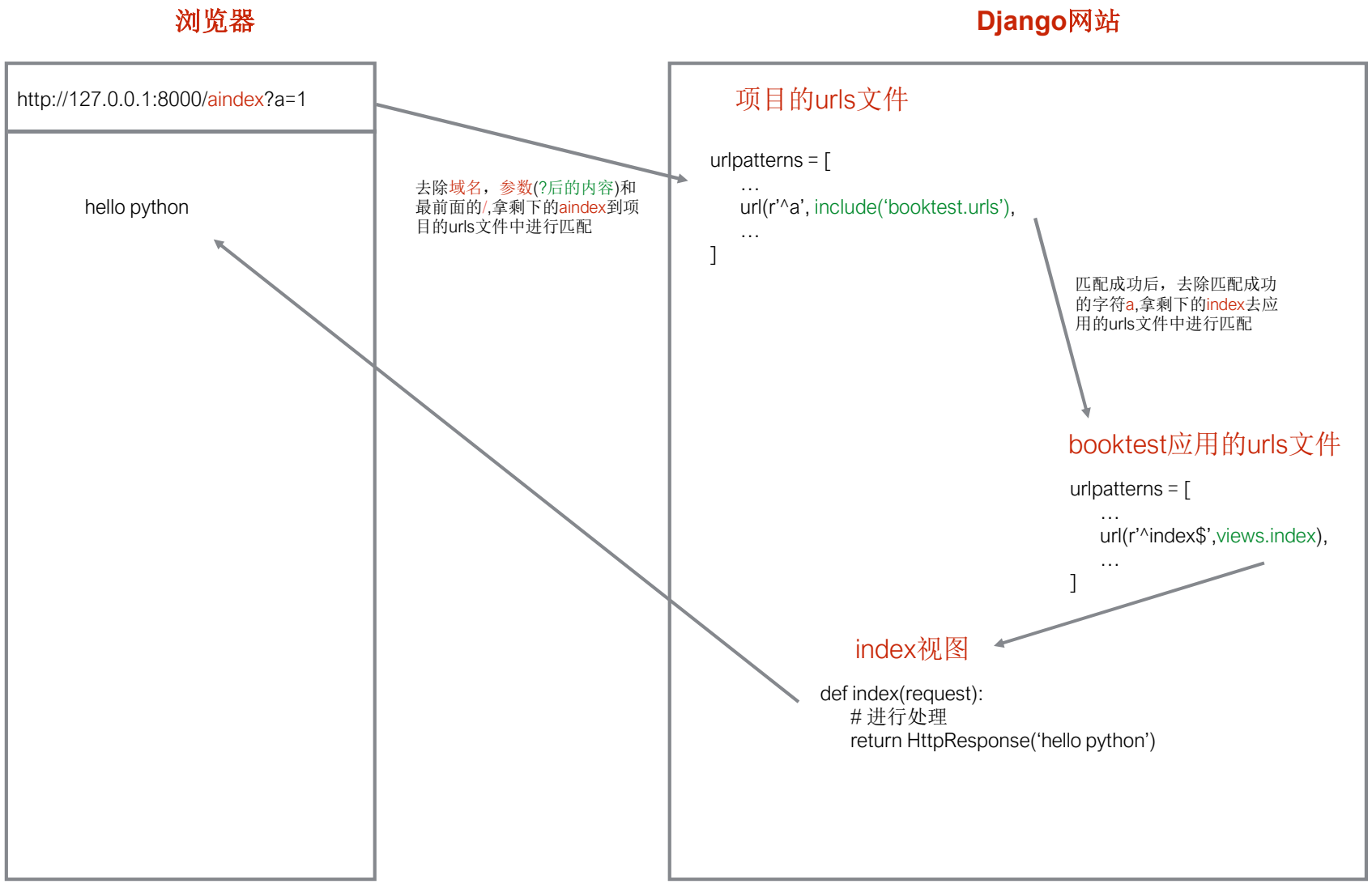
模型管理器类



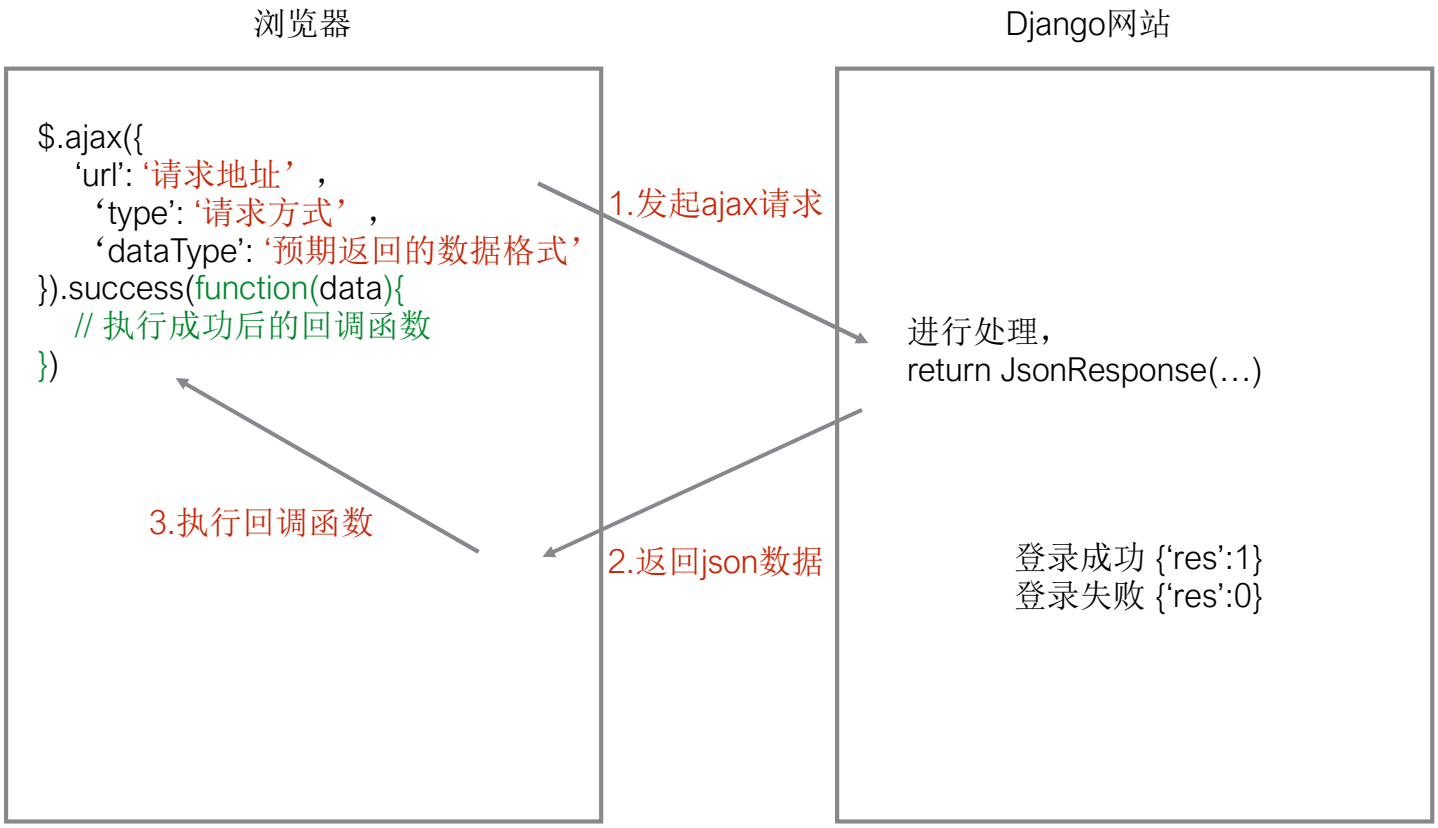
1. 改变原有查询的结果集。
2. 封装方法，用户操作管理器对象所在模型类对应的数据表。


浏览器
web服务器
application(env, start_response)
Django框架
http协议
1. 视图处理函数
2. 进行url配置
3. 通过模型操作数据库
4. 通过模板产生页面

URL匹配的过程


浏览器
Django网站
http://127.0.0.1:8000/aindex?a=1
hello python
去除域名，参数(?后的内容)和最前面的/,拿剩下的aindex到项目的urls文件中进行匹配
项目的urls文件
urlpatterns = [
…
url(r'^a', include('booktest.urls'),
…
]
匹配成功后，去除匹配成功的字符a,拿剩下的index去应用的urls文件中进行匹配
booktest应用的urls文件
urlpatterns = [
…
url(r'^index$',views.index),
…
]
index视图
def index(request):
# 进行处理
return HttpResponse('hello python')

ajax请求



1. 点击普通的超链接和手动输入地址。
2. 表单提交。
3. 通过ajax请求。

请求->响应

处理函数
1. 返回页面 HttpResponse
2. 重定向 HttpResponseRedirect
3. 返回json数据 JsonResponse

ajax:异步的javascript

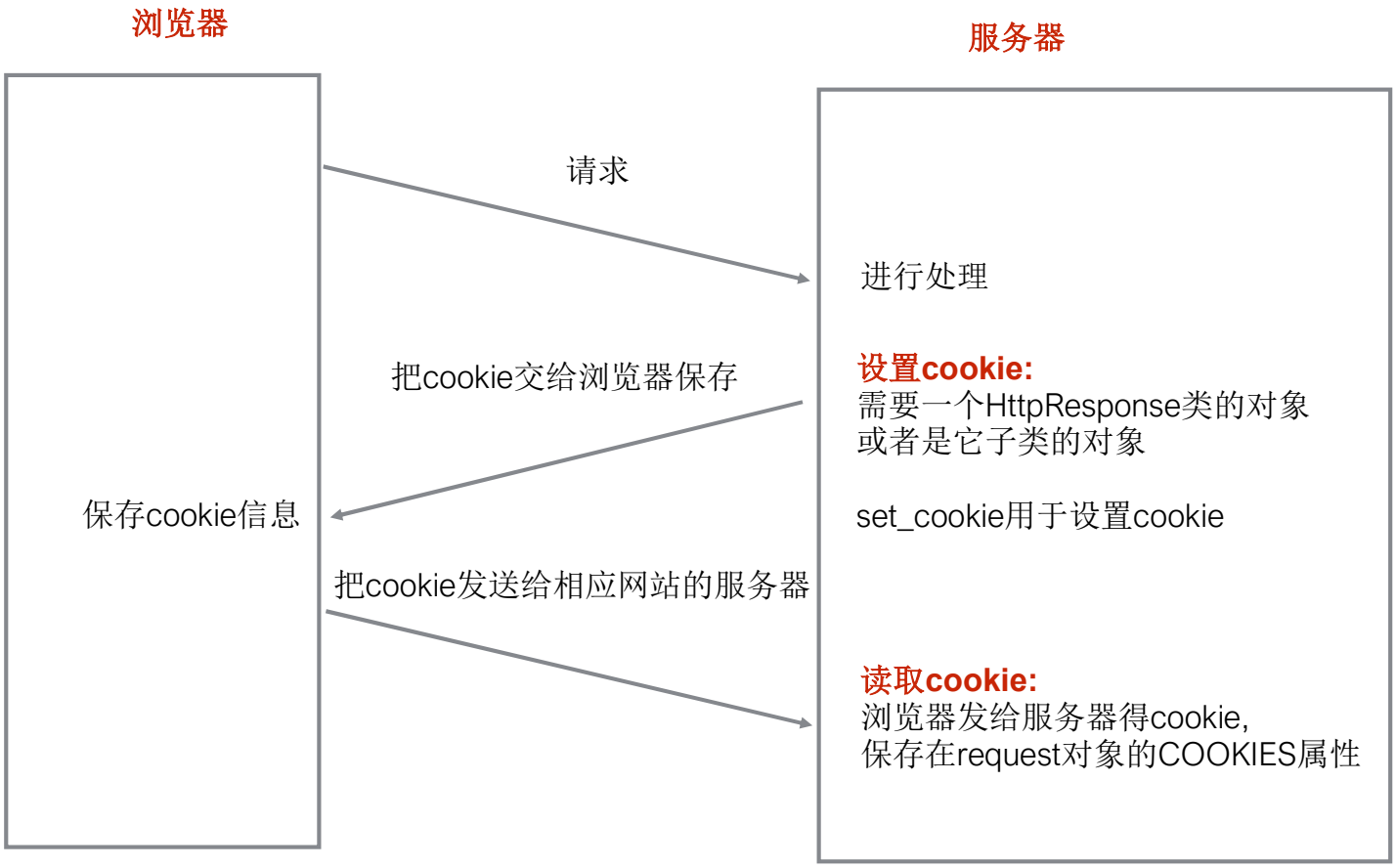
在不重新加载页面的情况下，对页面进行局部的刷新。

```
$.ajax({
    ‘url’ : 请求地址,
    ‘type’: 请求方式，
      ‘dataType’: 预期返回的数据格式
      ‘data’: 参数
}).success(function(data){
    // 回调函数
})
```

request对象的属性:
POST:保存post提交提交的参数。
GET:保存get提交的参数。
method:获取请求的方式(GET和POST)
path:获取请求的路径，不包括域名和参数。
COOKIES:保存浏览器发过来的cookie信息，字典类。
session:设置或读取session信息，类似于字典。

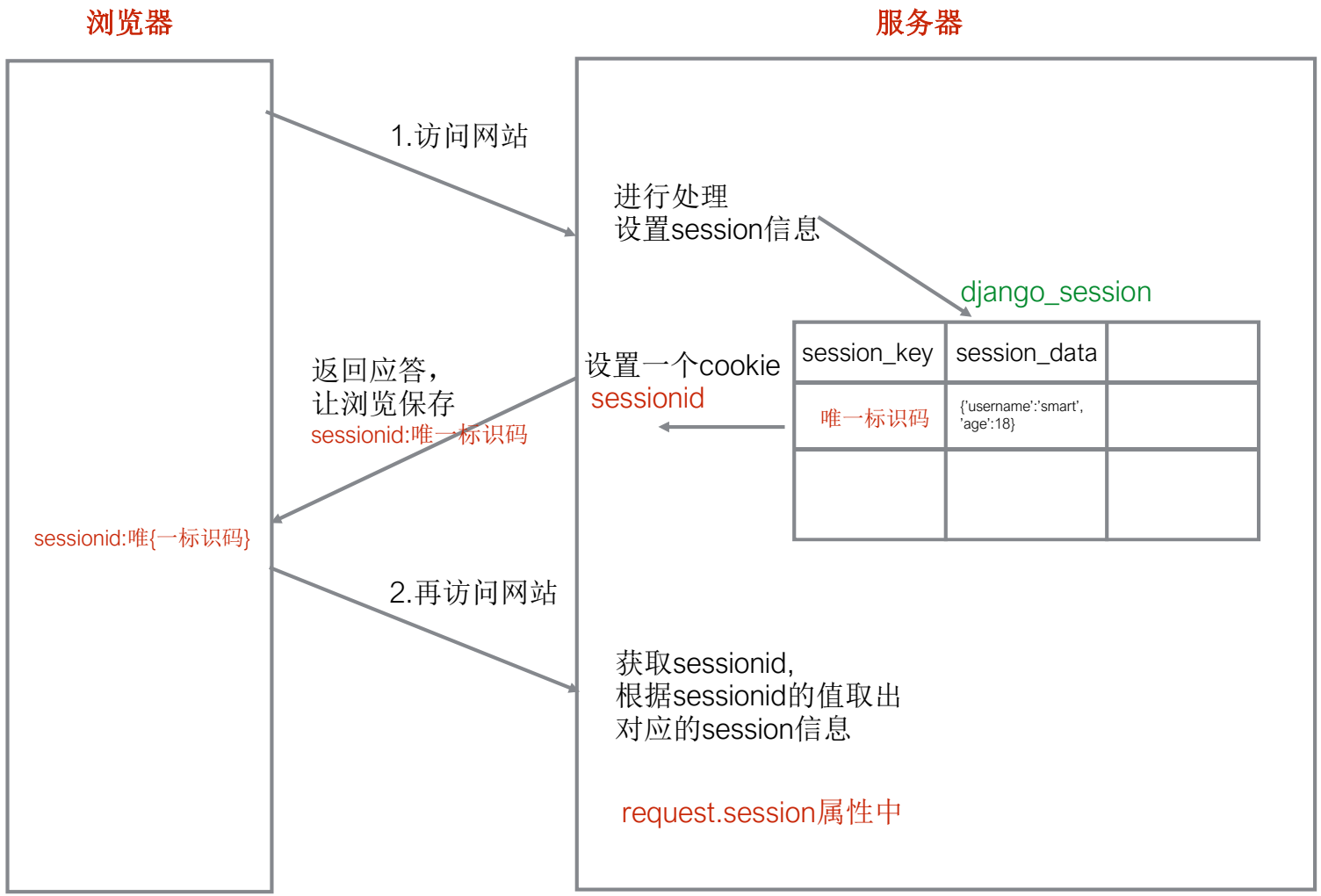
Cookie

你->老板 买豆浆
老板给你一个单子 设置cookie
拿单子找老板要豆浆 读取cookie



Session

你去办健身卡，你的信息都保存健身俱乐部的电脑中，给你卡号 cookie sessionid



设置session: request.session['username']='smart'
获取session:request.session['username']

昨日问题

1.redirect和render算是HttpResponse的对象或者子对象吗？设置cookie的时候连写为什么报错啊？
比如 return.set_cookie('num',1,max_age(7*24*3600))？
redirect->HttpResponseRedirect render->HttpResponse
response = HttpResponse('设置cookie')
response.set_cookie('num',1, max_age=7*24*3600)
return response

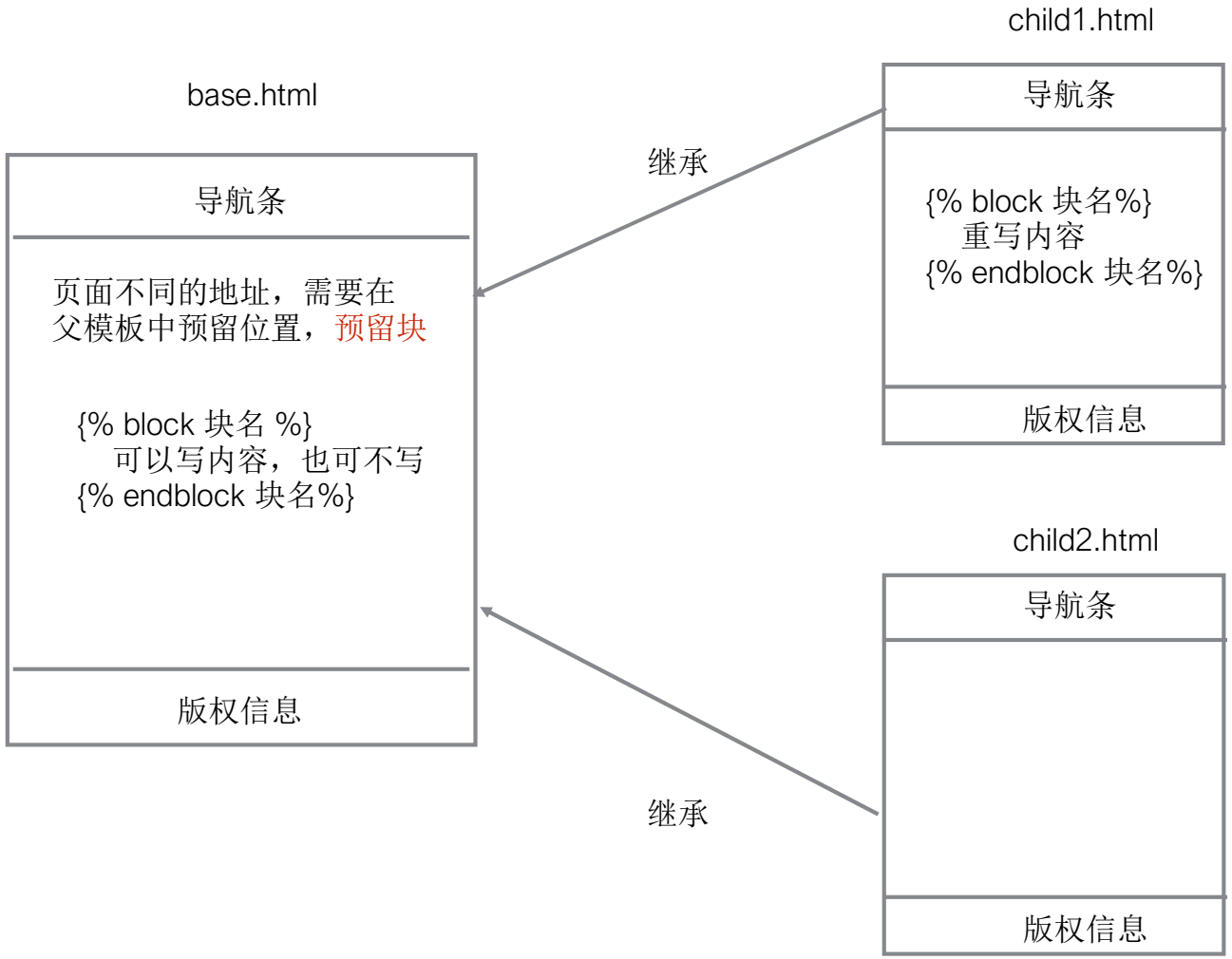
2. 对于模型中已有数据的表进行结构修改，只能删除表然后重建吗?

3. 复选框不指定value值得情况下勾选不勾选val()都是on ，是不是应该判断复选框的prop("checked")值
勾选true 不勾选false
通过ajax进行提交数据，$('input[name="remember"]').prop('checked') true false

4. 勾选 checkbox 跟不勾选 一样值都是 on 怎么解决？

5. ajax里面不能设置 session 吗？

模板继承



把所有页面相同的内容放到父模板文件中，不需要放在块中。
有些位置页面内容不同，需要在父模板中预留块。

登录装饰器

1.在进行网站开发的时候，有些页面是用户登录之后才能访问的，假如用户访问了这个地址，
需要进行登录的判断，如果用户登录的话，可以进行后续的操作，如果没有登录，跳转到登录页。

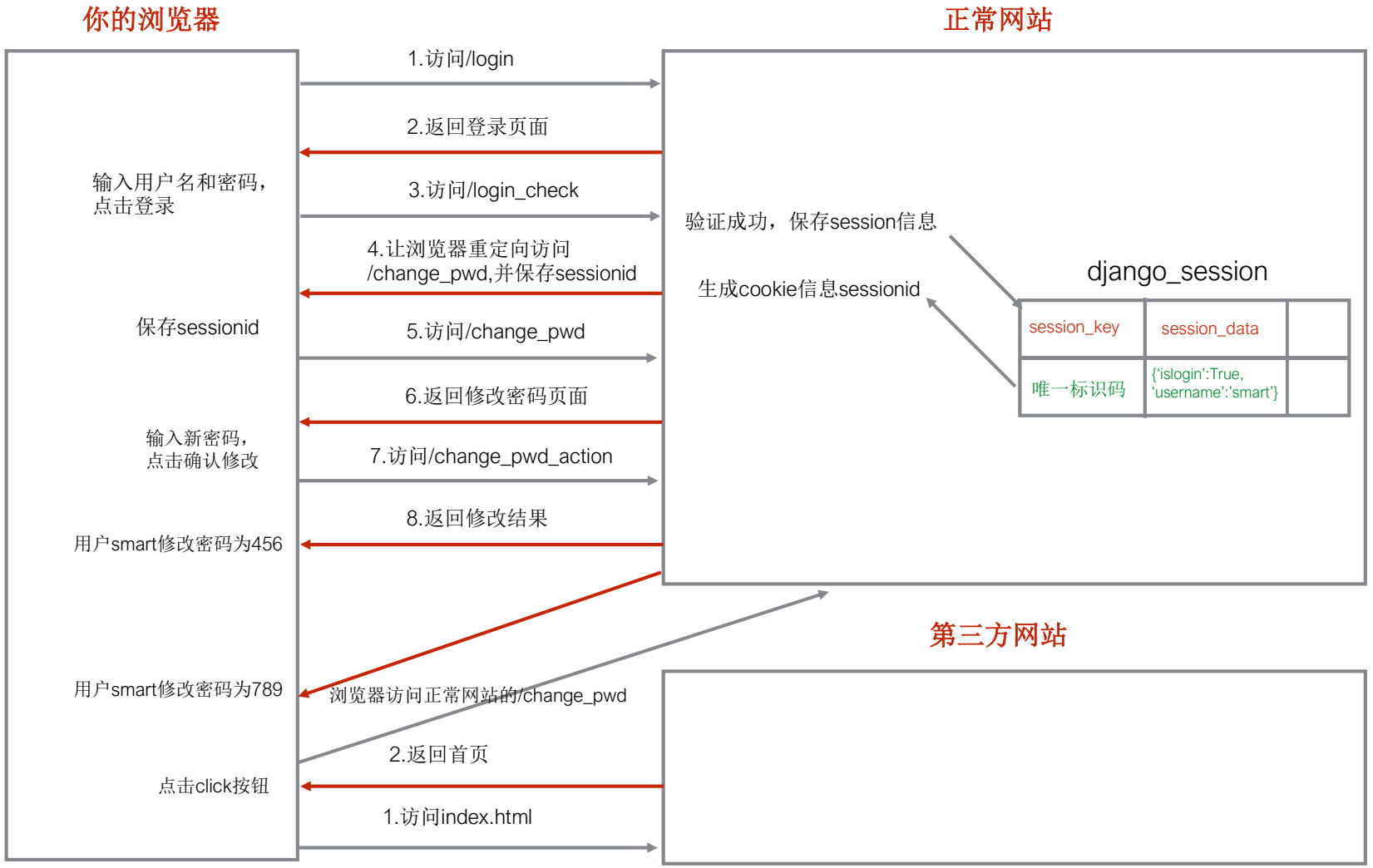
```
def login_required(view_func):
    def wrapper(request, *args, **kwargs):
        if request.session.has_key('islogin'):
            # 用户登录
            return view_func(request, *args, **kwargs)
        else:
            # 用户未登录
            return redirect('/login')
    return wrapper
```

```
@login_required
def index(request):
    return HttpResponse('index')
```

```
url(r'^index$', views.index)
```

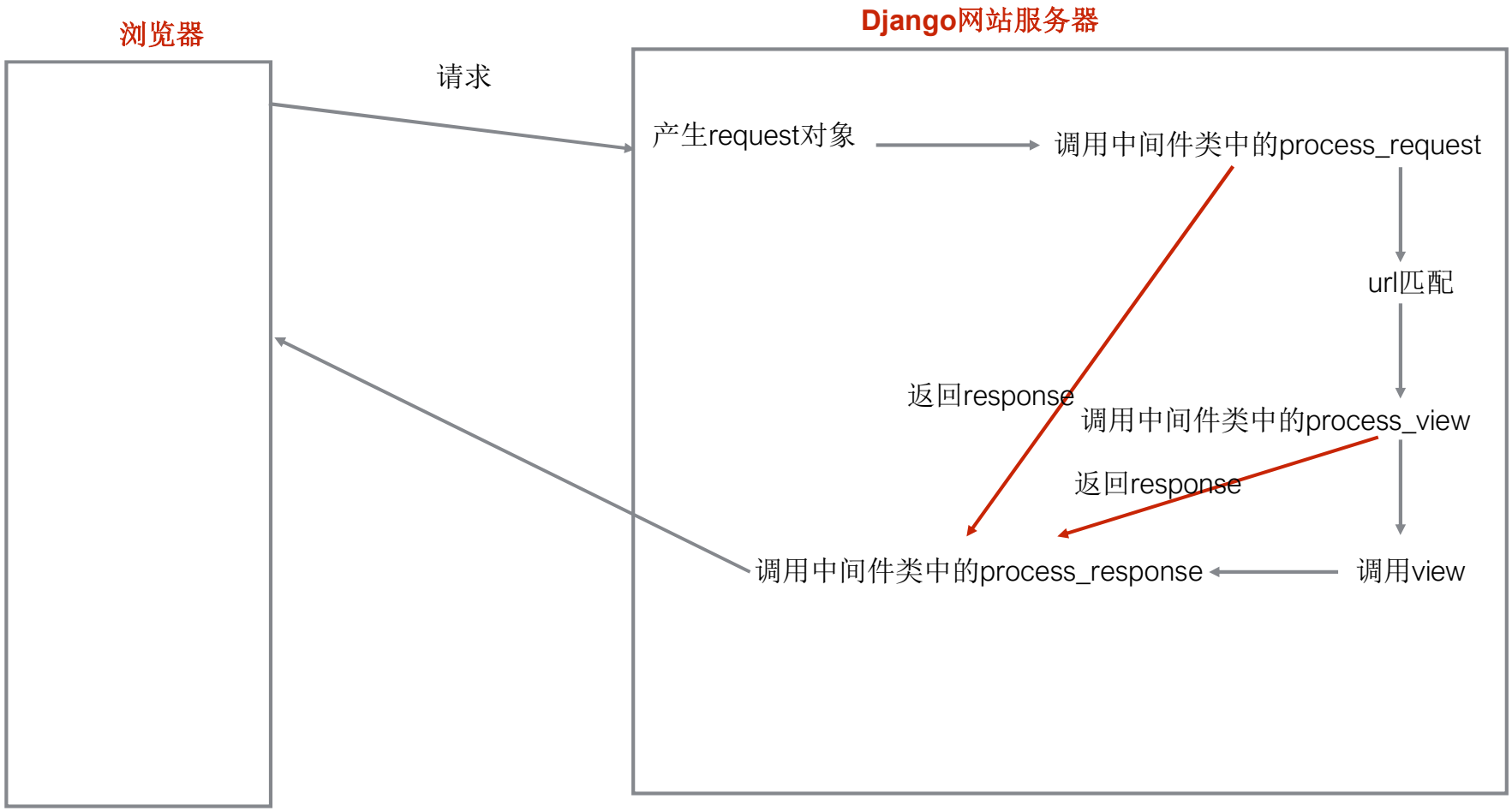
```
return login_required(index)(request, *args, **kwargs)
```

csrf伪造

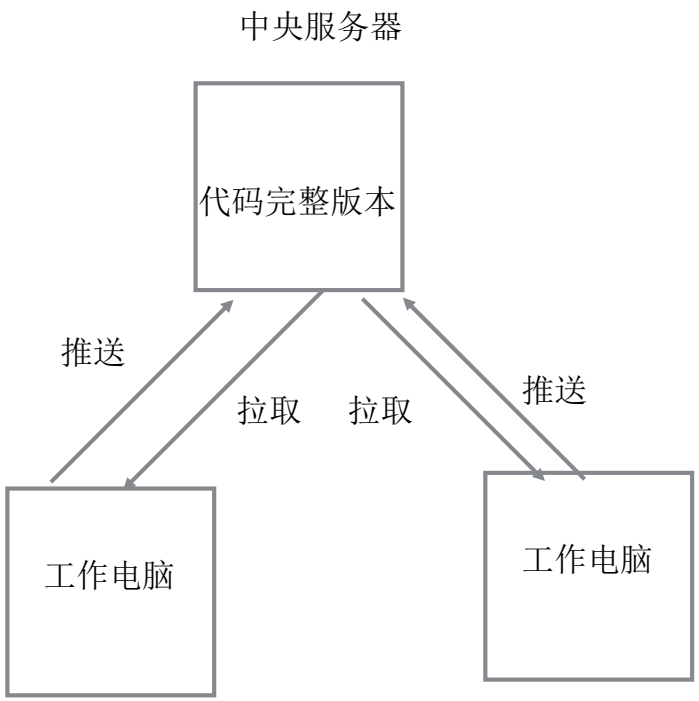


1. 登录正常网站之后，你的浏览器保存的sessionid, 你没有退出。
2. 你不小心访问了另外一个网站，并且你点击了页面上的按钮。。。

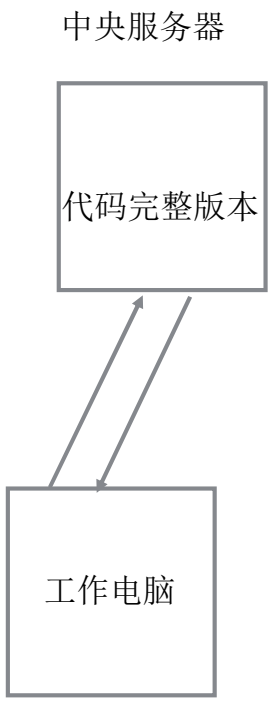
中间件


浏览器
Django网站服务器
请求
产生request对象
调用中间件类中的process_request
url匹配
返回response
调用中间件类中的process_view
返回response
调用中间件类中的process_response
调用view

分布式



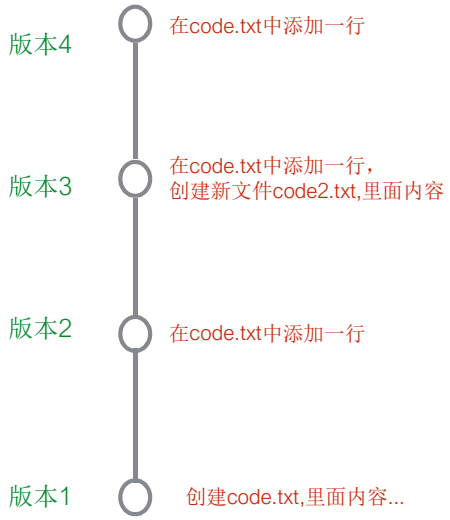
集中式

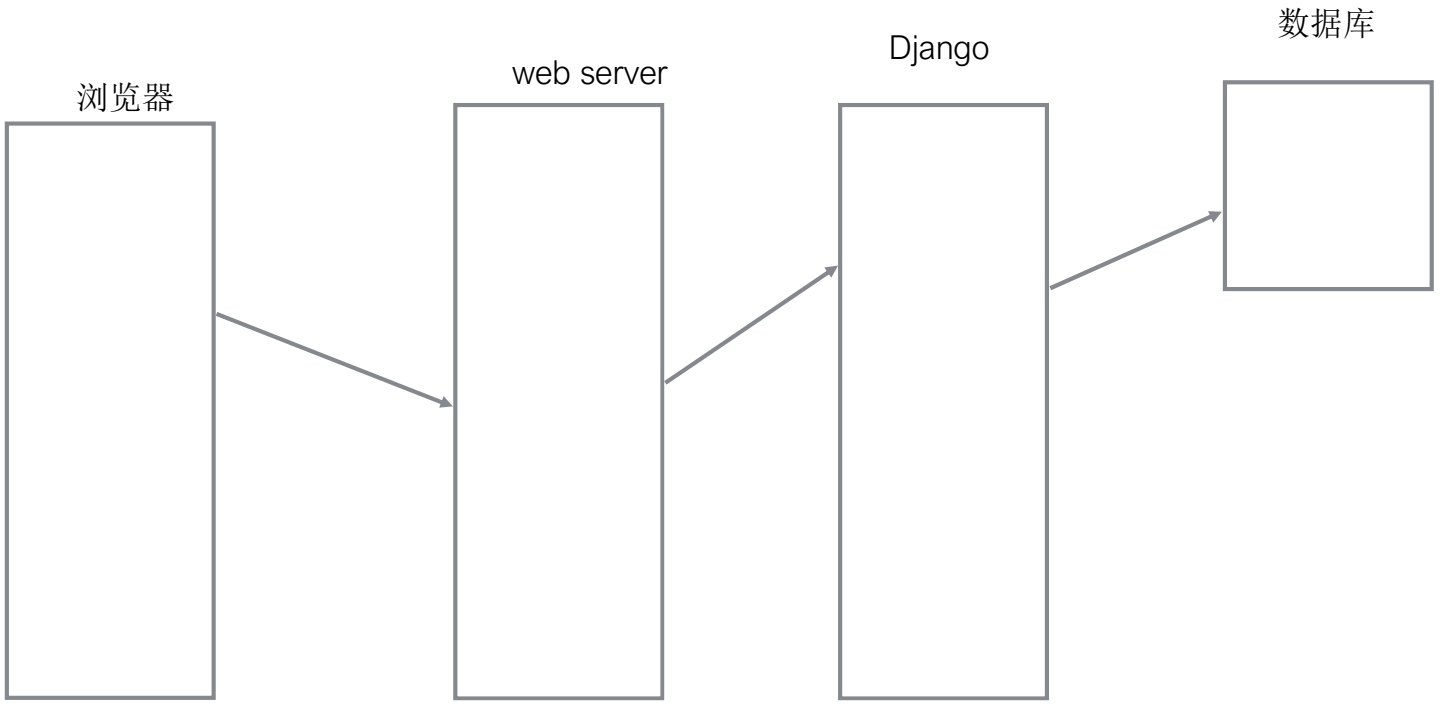


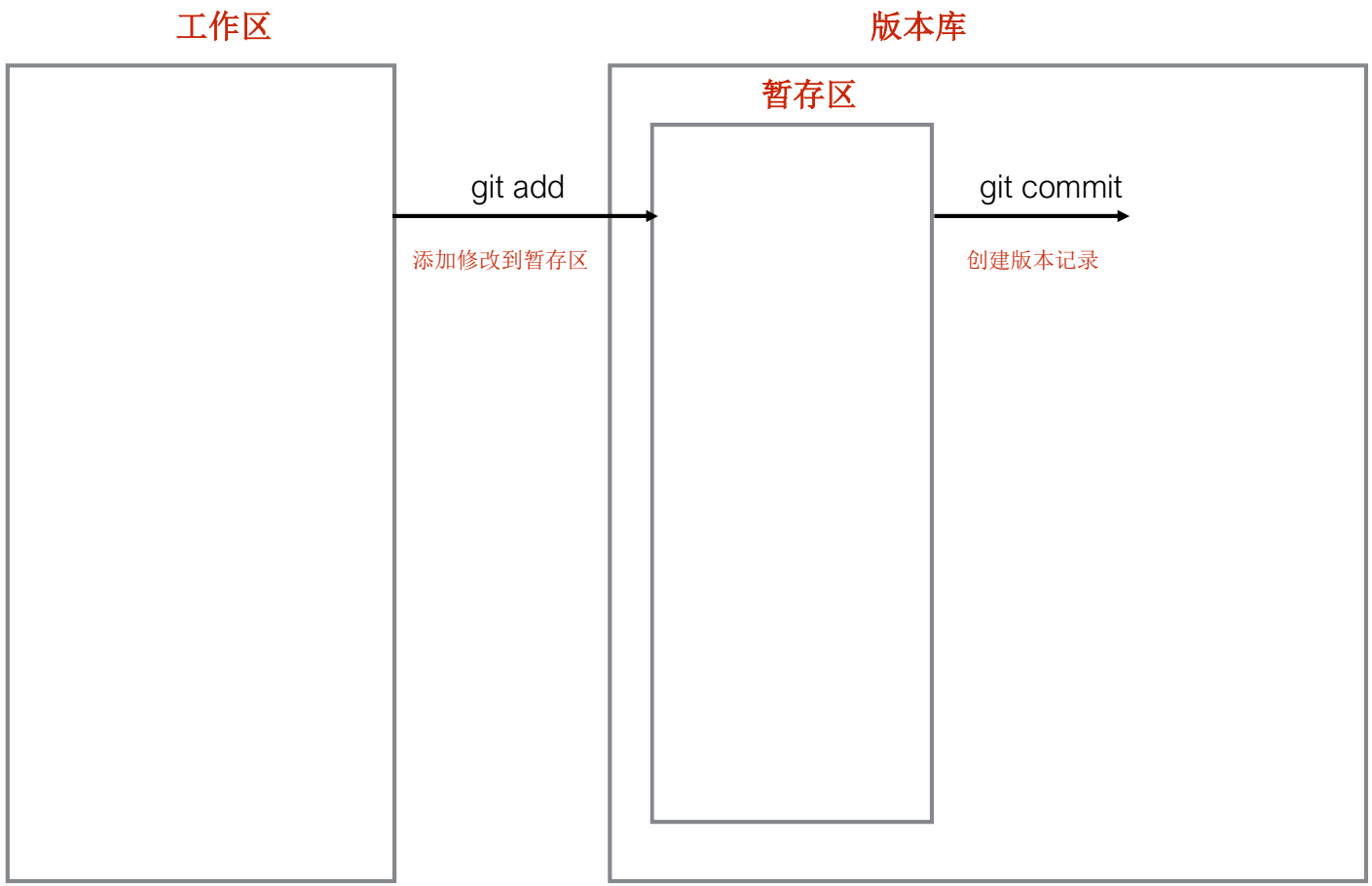
分布式中央服务器挂掉之后仍然可以进行工作。
集中式中央服务器挂掉之后不能进行工作。

git merge —no-ff -m ‘禁用fast-forward’ dev

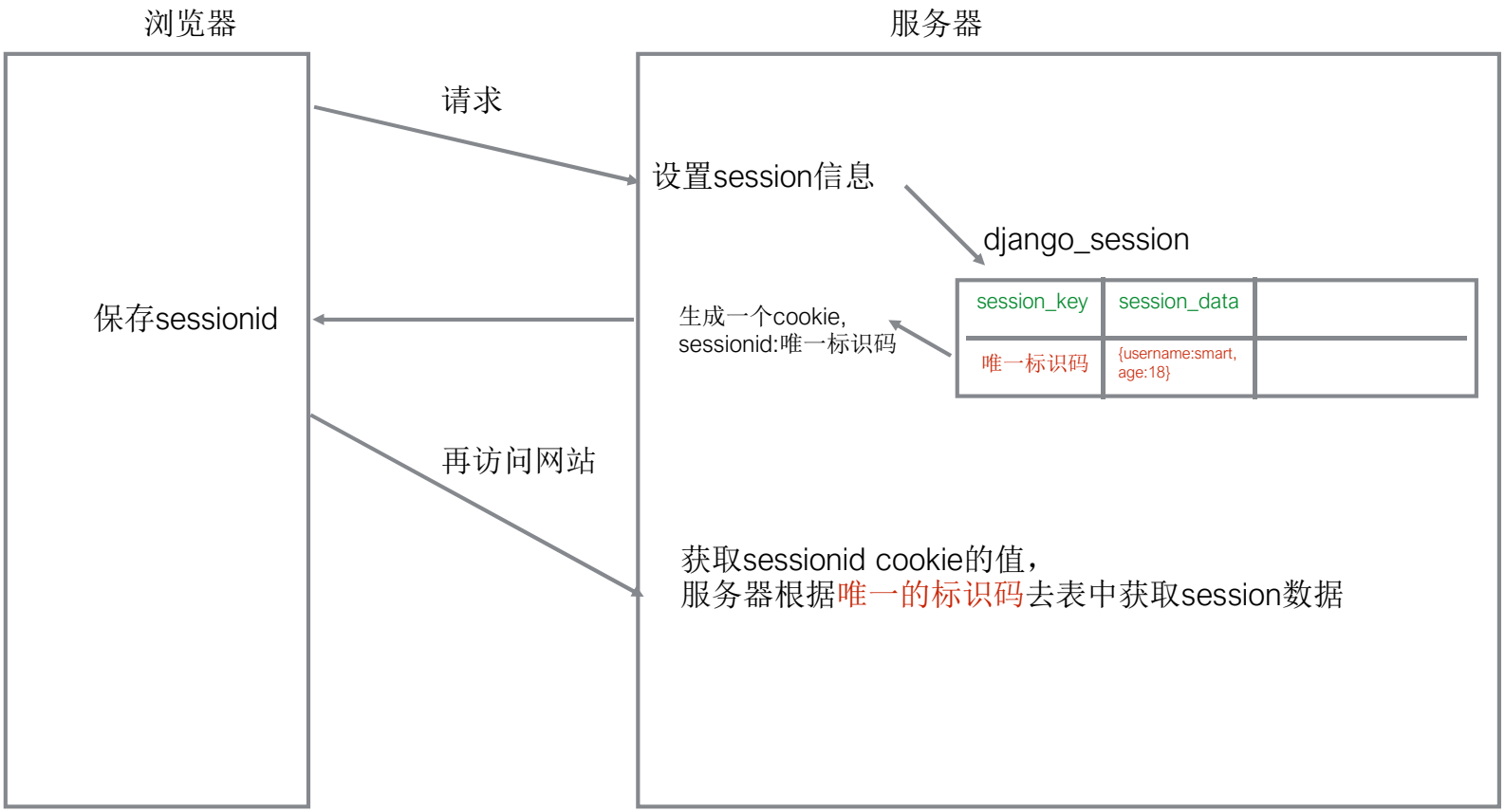
HEAD^:HEAD版本的前一个版本。
HEAD~1：HEAD版本的前一个版本。
HEAD~100：HEAD版本的前100个版本。



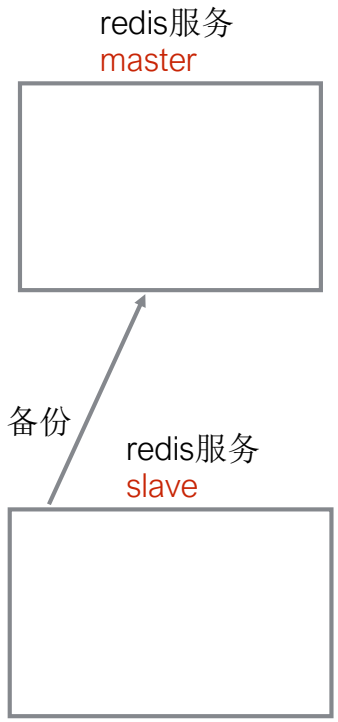

浏览器
web server
Django
数据库


工作区
版本库
暂存区
git add
添加修改到暂存区
git commit
创建版本记录

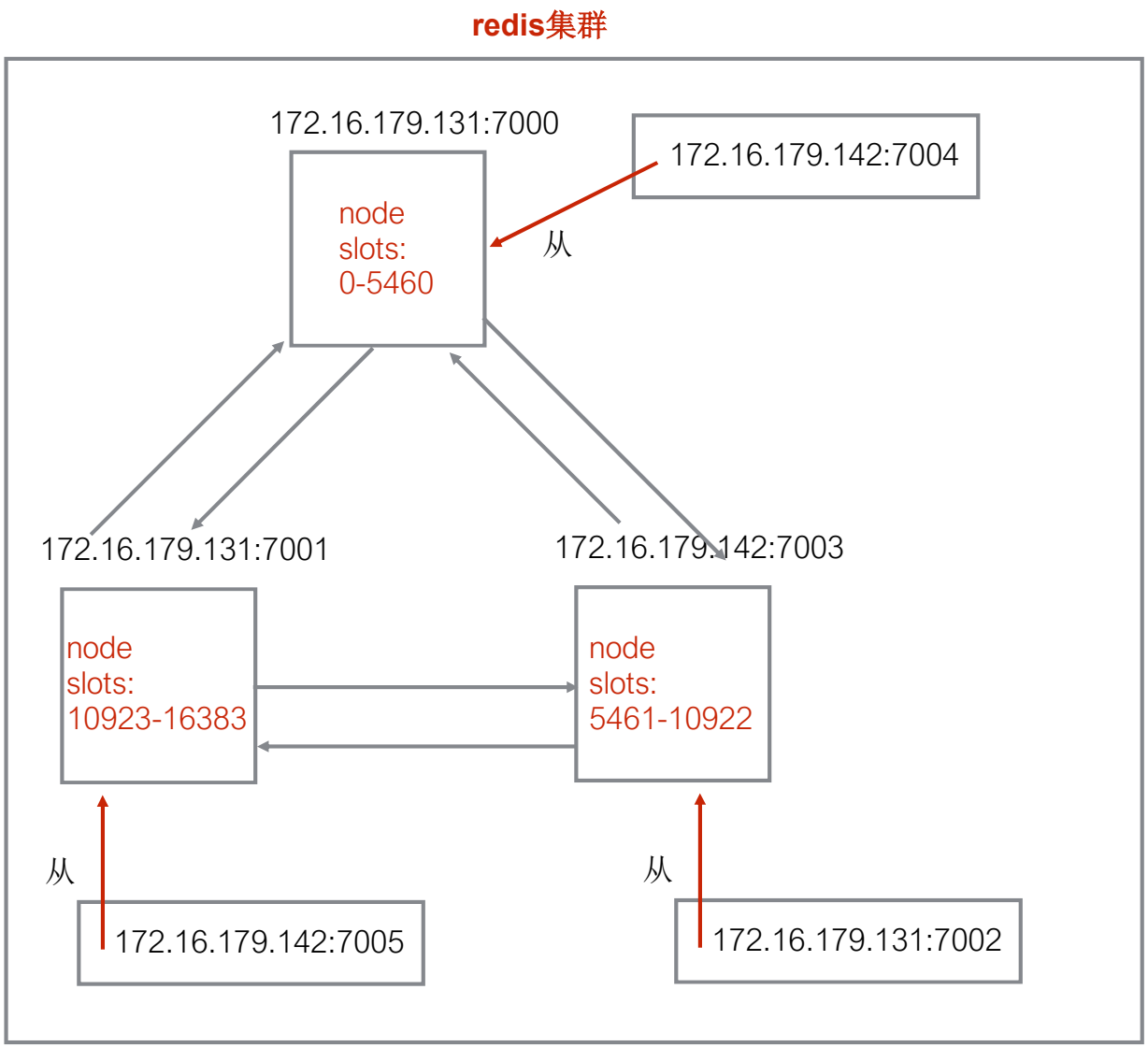
1.什么是cookie?
2.什么是session?
你去办健身卡，你的信息存在健身俱乐部的电脑中，给你一卡号。



读取session:username = request.session['username']
设置session:request.session['username']='smart'



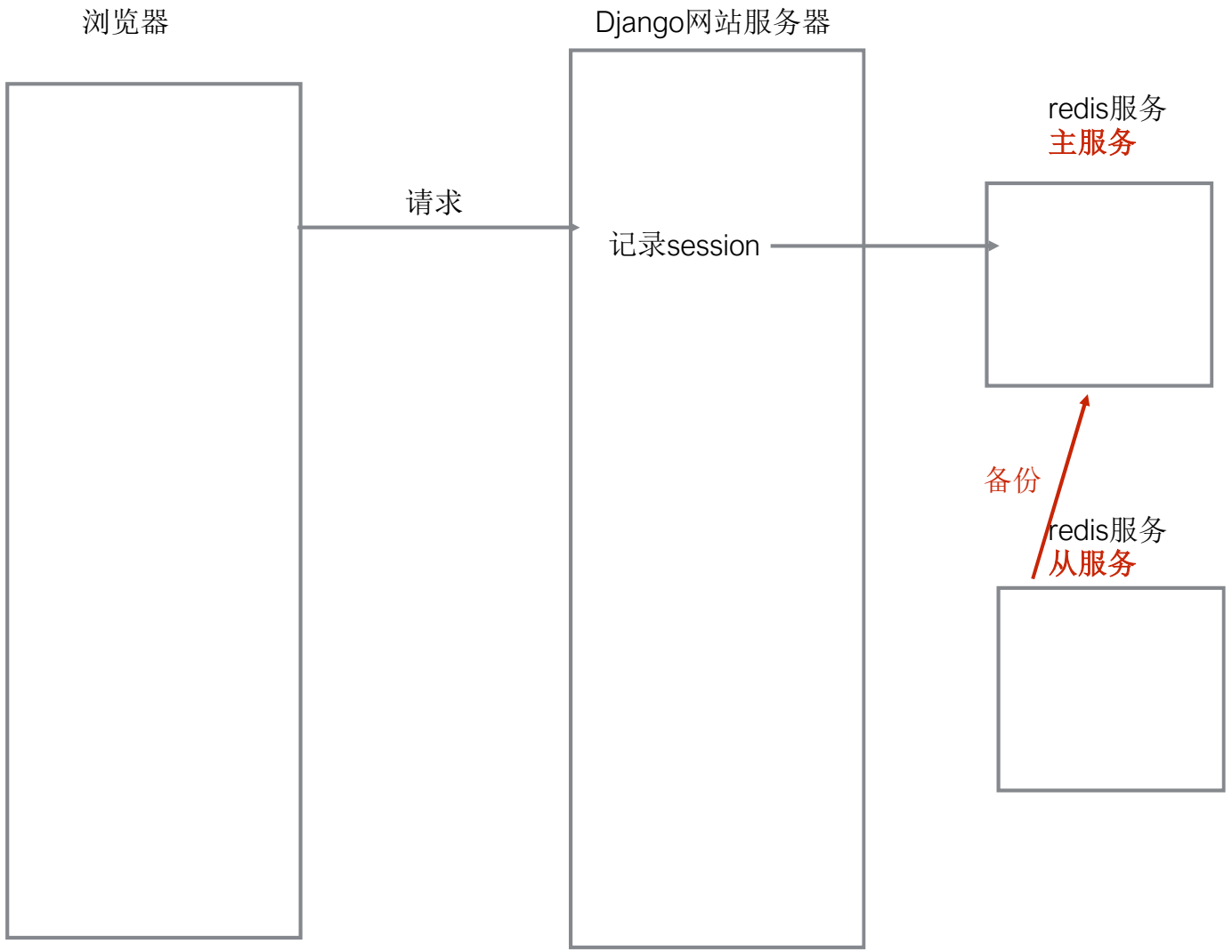
一个主可以有多个从，
从服务还可以有自己的从服务。


redis集群
172.16.179.131:7000
172.16.179.142:7004
node
slots:
0-5460
从
172.16.179.131:7001
172.16.179.142:7003
node
slots:
10923-16383
node
slots:
5461-10922
从
从
172.16.179.142:7005
172.16.179.131:7002

mysql
oracle
sql server
关系型数据库

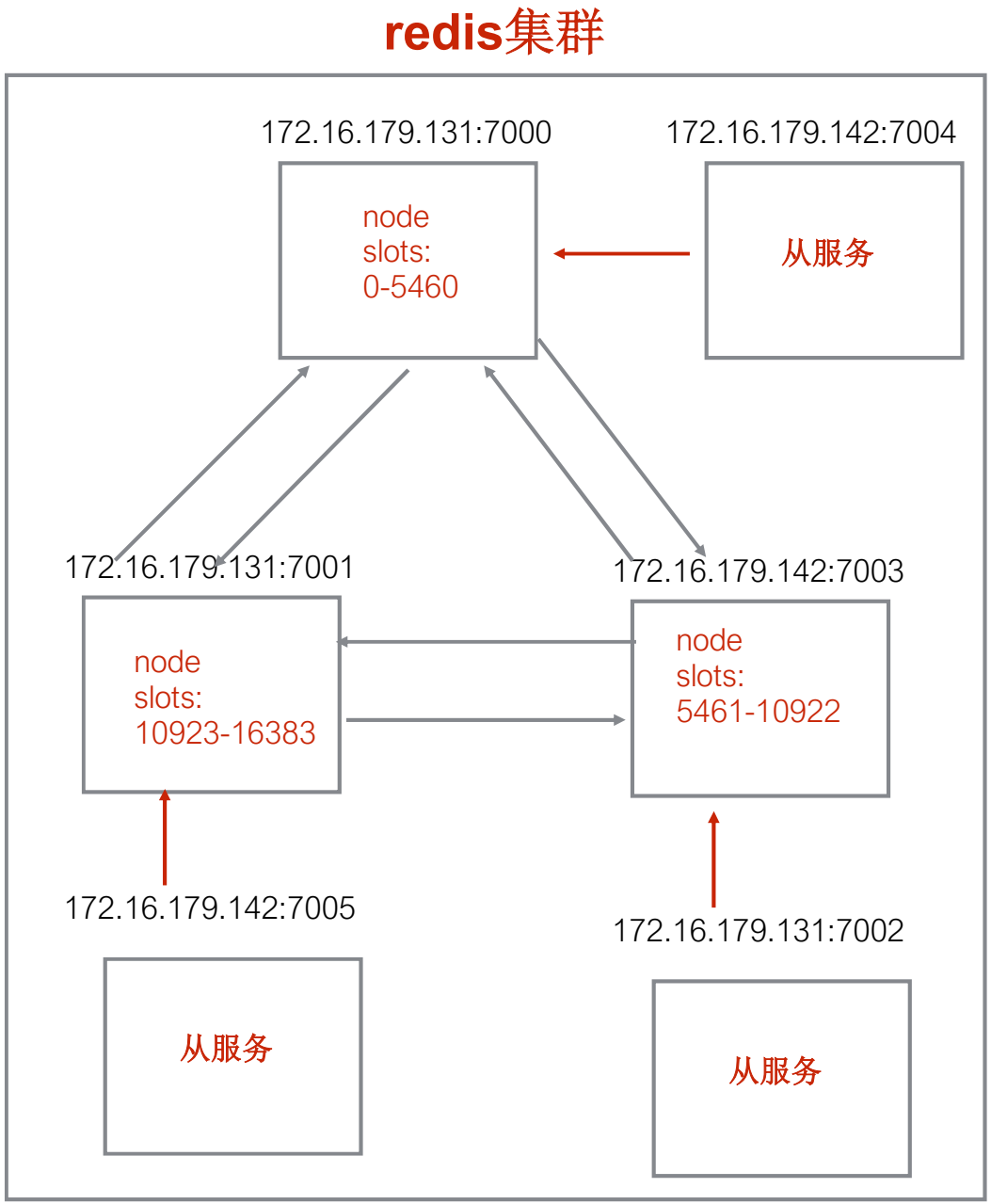
通用的操作语言:SQL语句

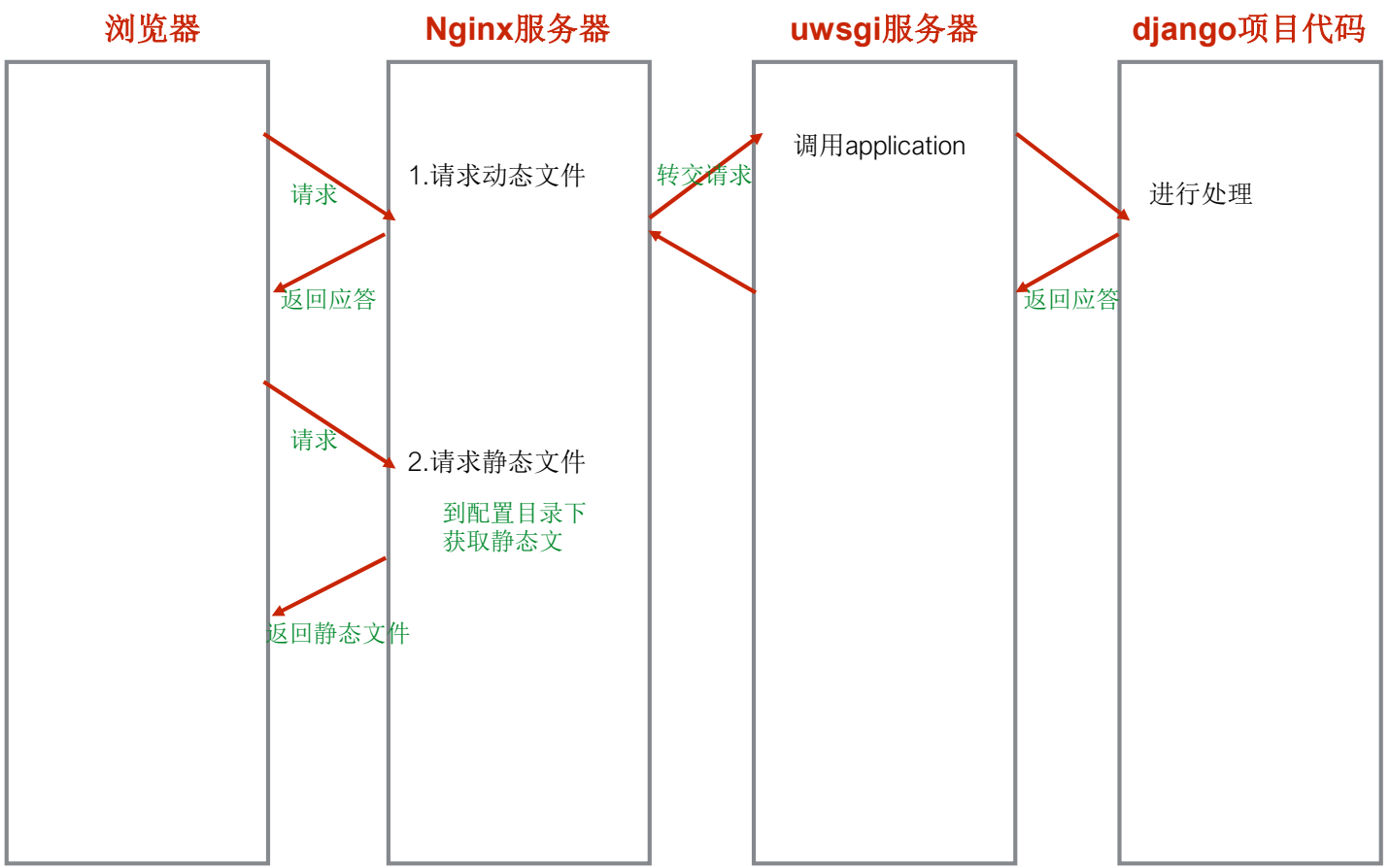
事务:一组sql操作，要么都成功，要么都失败。


浏览器
Django网站服务器
redis服务
主服务
请求
记录session
备份
redis服务
从服务

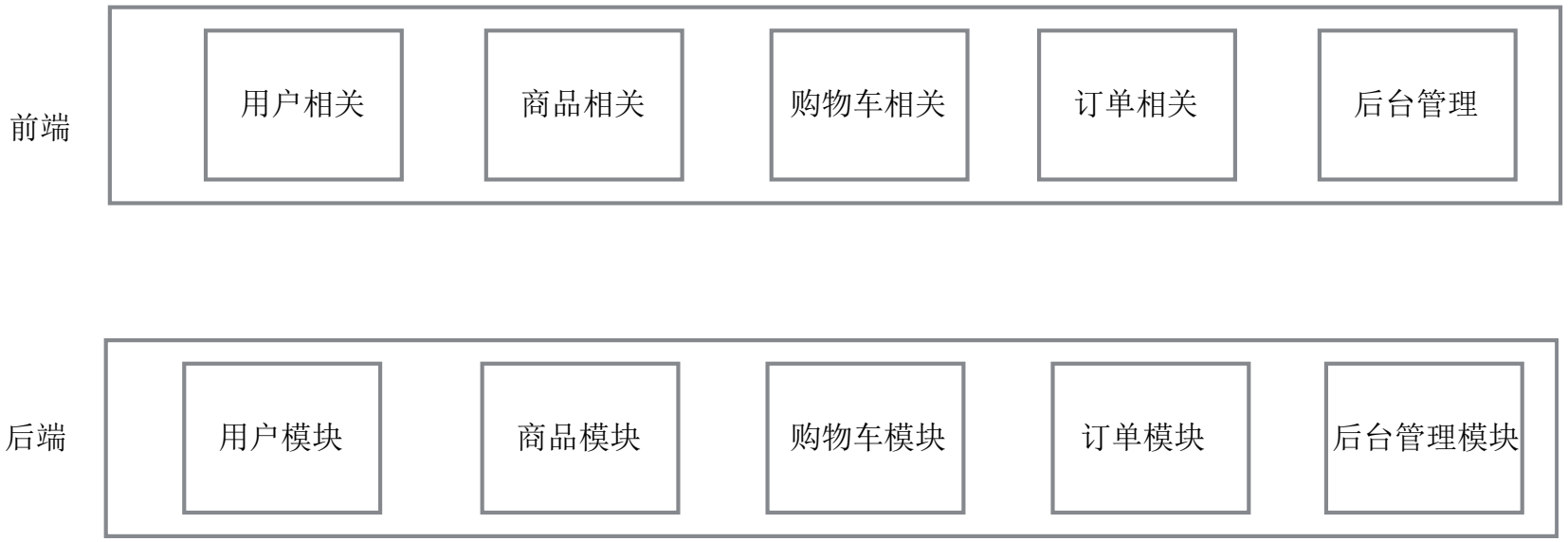
集群中设置数据:
set key value

集群:一组通过网络连接的计算机，共同对外提供服务，像一个独立的服务器。




浏览器
Nginx服务器
uwsgi服务器
django项目代码
请求
1.请求动态文件
转交请求
调用application
进行处理
返回应答
返回应答
请求
2.请求静态文件
到配置目录下
获取静态文
返回静态文件

项目架构


前端
用户相关
商品相关
购物车相关
订单相关
后台管理
后端
用户模块
商品模块
购物车模块
订单模块
后台管理模块
mysql
session 缓存服务器
redis
异步任务处理
celery
分布式文件存储系统
fastdfs

数据库设计

以空间换取时间

用户表

ID
用户名
密码
邮箱
激活标识
权限标识

地址表

ID
收件人
收件地址
邮编
联系方式
是否默认
用户ID

redis实现购物车功能
redis保存用户历史浏览记录

订单信息表

订单ID
地址ID
用户ID
支付方式
*总数目
*总金额
运费
支付状态
创建时间

订单商品表

ID
订单ID
sku ID
商品数量
商品价格
评论

商品SKU表

ID
名称
简介
价格
单位
库存
销量
*图片
状态
种类ID
spu ID

商品图片表

ID
图片
sku ID

商品种类表

ID
种类名称
logo
图片

商品SPU表

ID
名称
详情

首页分类商品展示表

ID
sku ID
种类ID
展示标识
index

首页轮播商品表

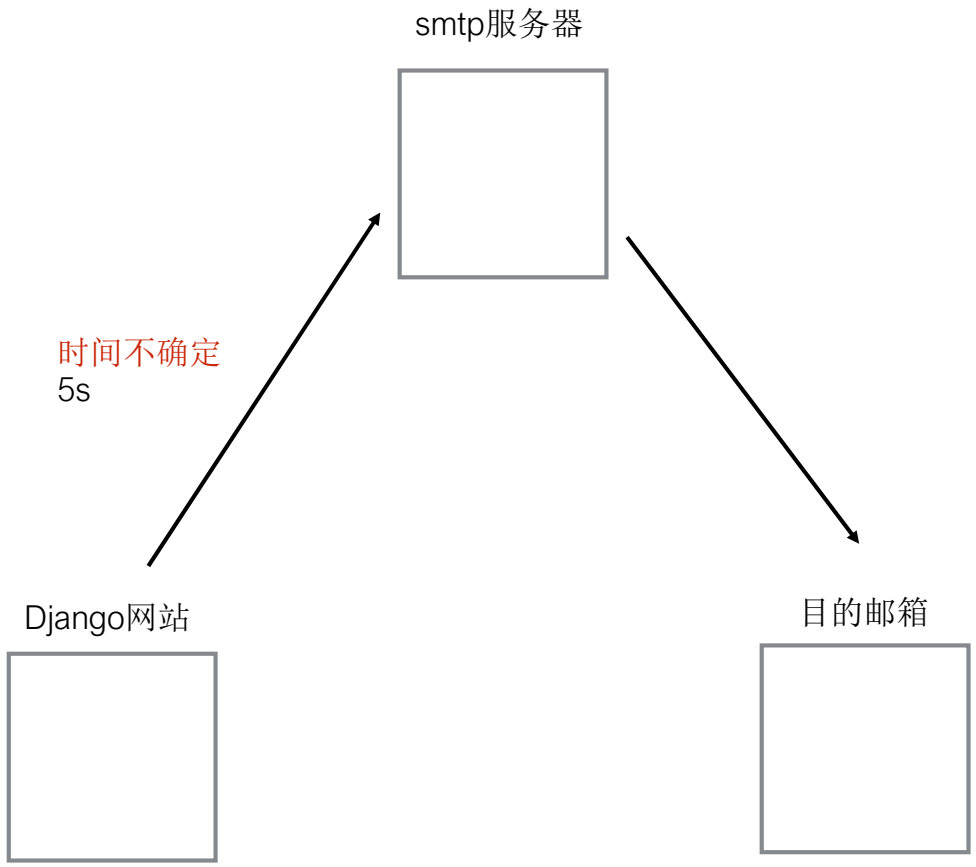
ID
sku id
图片
index

首页促销活动表

ID
图片
活动url
index

盒装草莓
500g草莓
2kg草莓

发送邮件



celery

任务的发出者，中间人，任务的处理者可以在同一台电脑上启动，
也可以不在同一台电脑上。

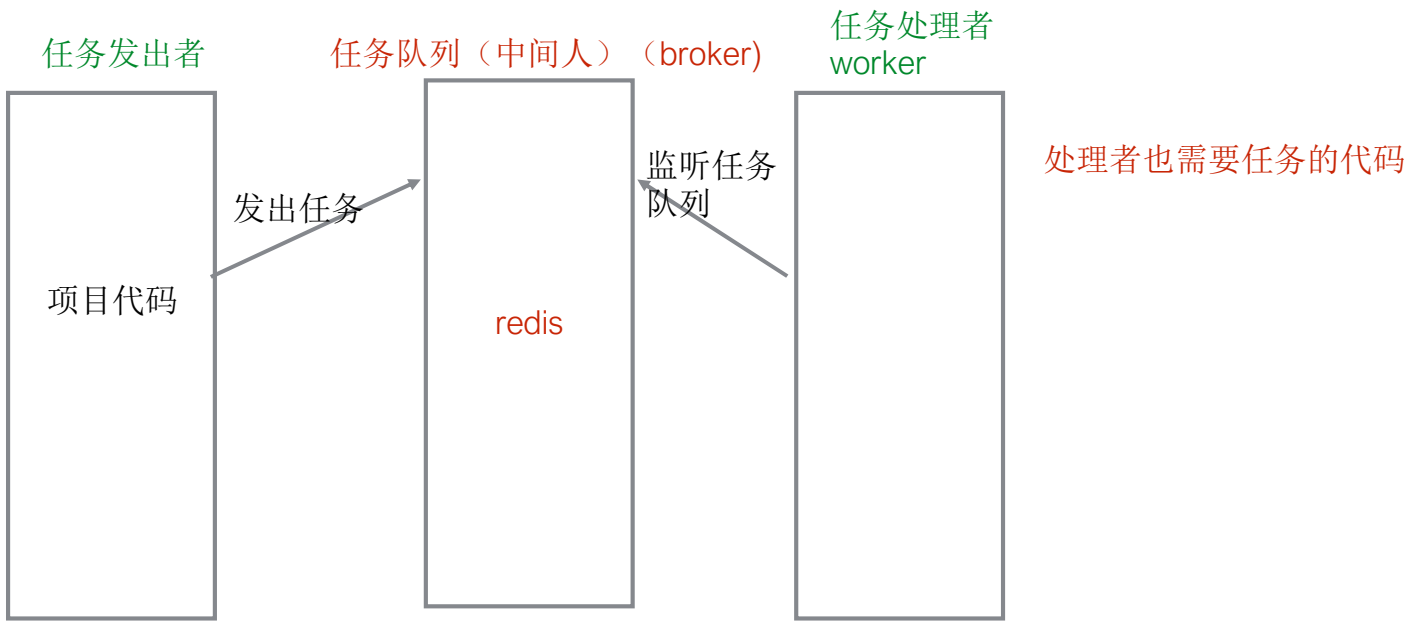
1.打游戏
2.点外卖

任务发出者：
你

点外卖:
发出任务

美团:
中间人

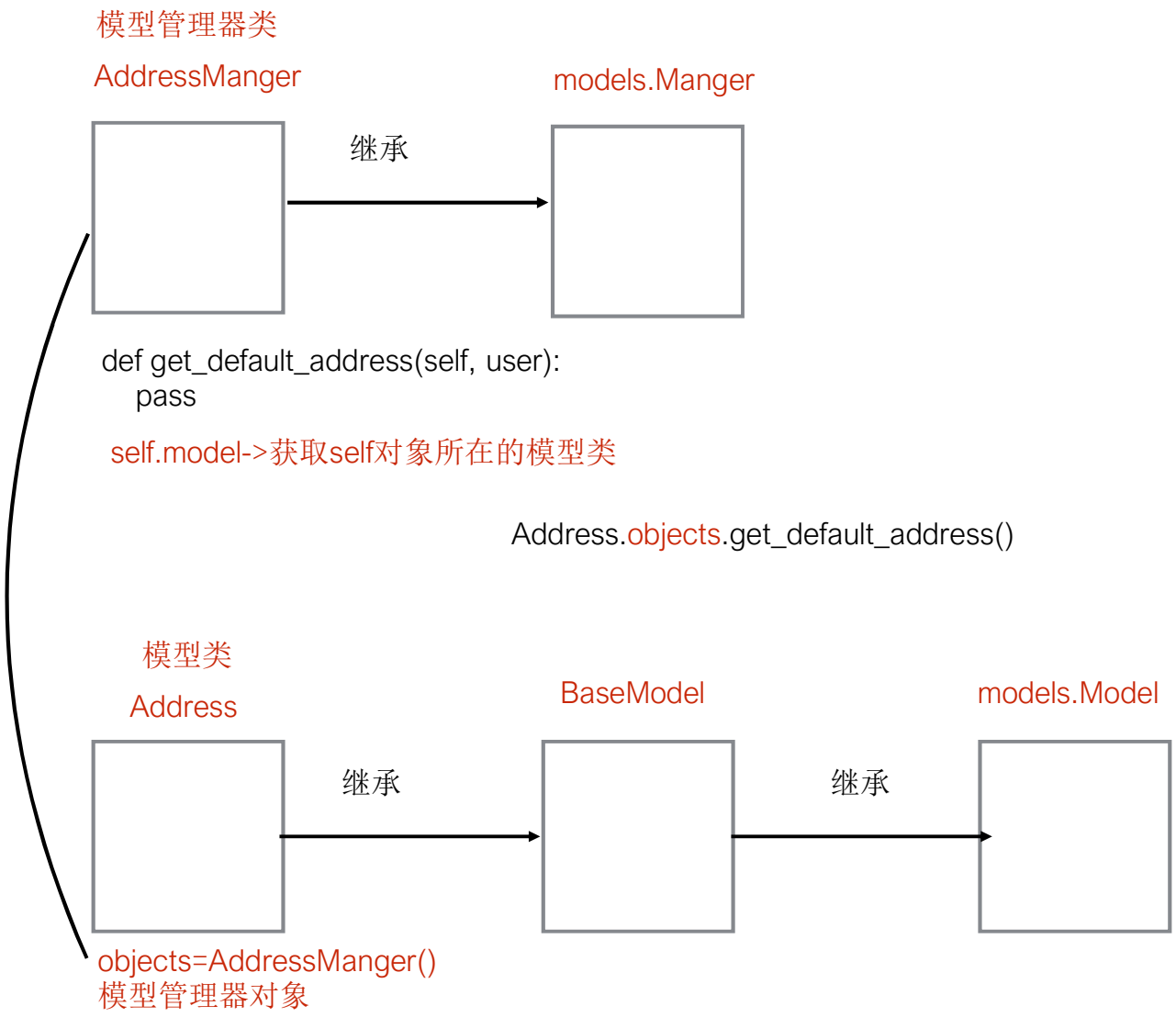
商家：
任务处理者



处理者也需要任务的代码

任务处理者所在电脑必须有网。

安装: pip install celery


模型管理器类
AddressManger
models.Manger
继承
def get_default_address(self, user):
pass
self.model->获取self对象所在的模型类
Address.objects.get_default_address()
模型类
Address
BaseModel
models.Model
继承
继承
objects=AddressManger()
模型管理器对象

redis存储历史浏览记录

1.什么时候需要添加历史浏览记录？
访问商品的详情页面的时候（在商品详情对应的视图中），需要添加历史浏览记录。

2.什么时候需要获取历史浏览记录？
访问用户中心个人信息页的时候获取历史浏览记录。

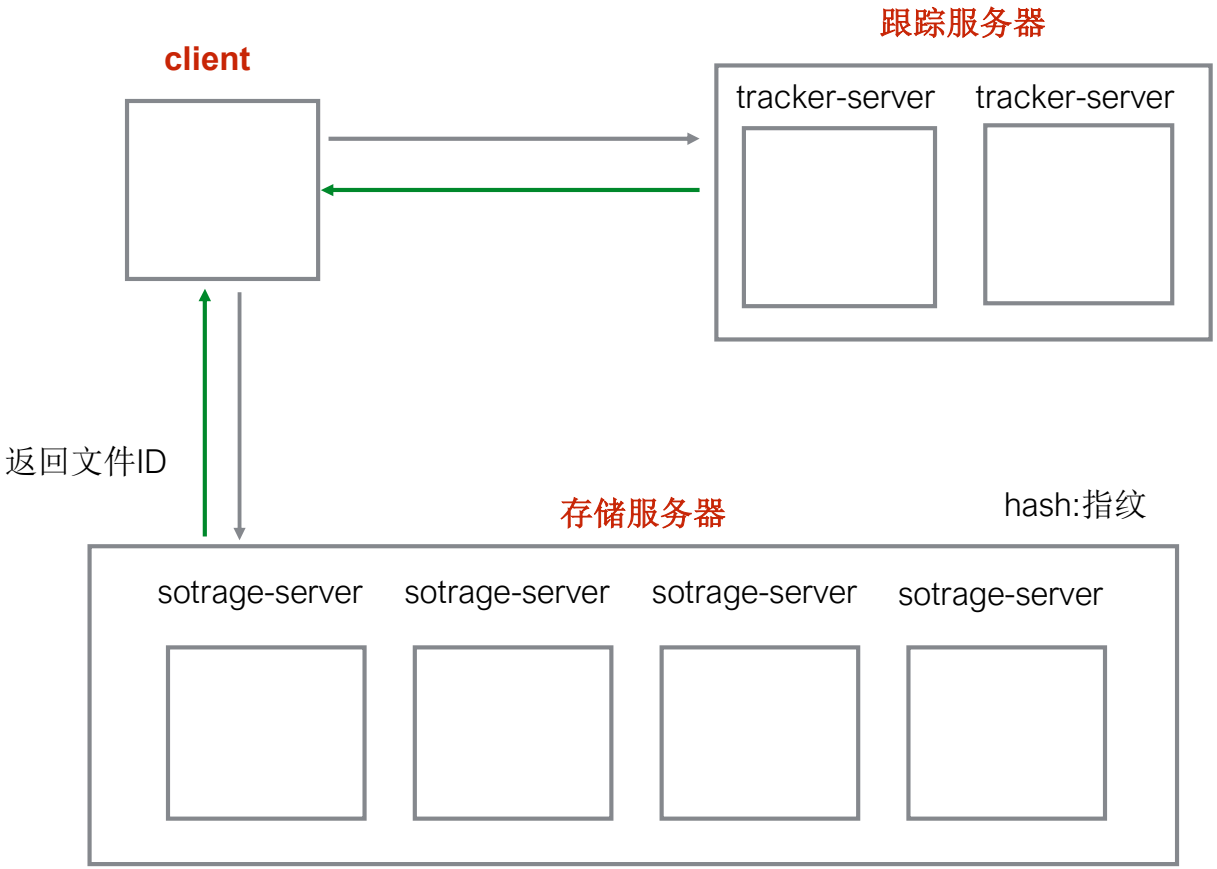
3.历史浏览记录需要存储在哪里？
redis数据库->内存型的数据库
使用redis数据库存储历史浏览记录。

4.redis中存储历史浏览记录的格式？
string
hash
list
set
zset

存储用户的历史浏览记录时，所有用户的历史浏览记录用一条数据保存
hash:
history : user_用户id:'1,2,3'

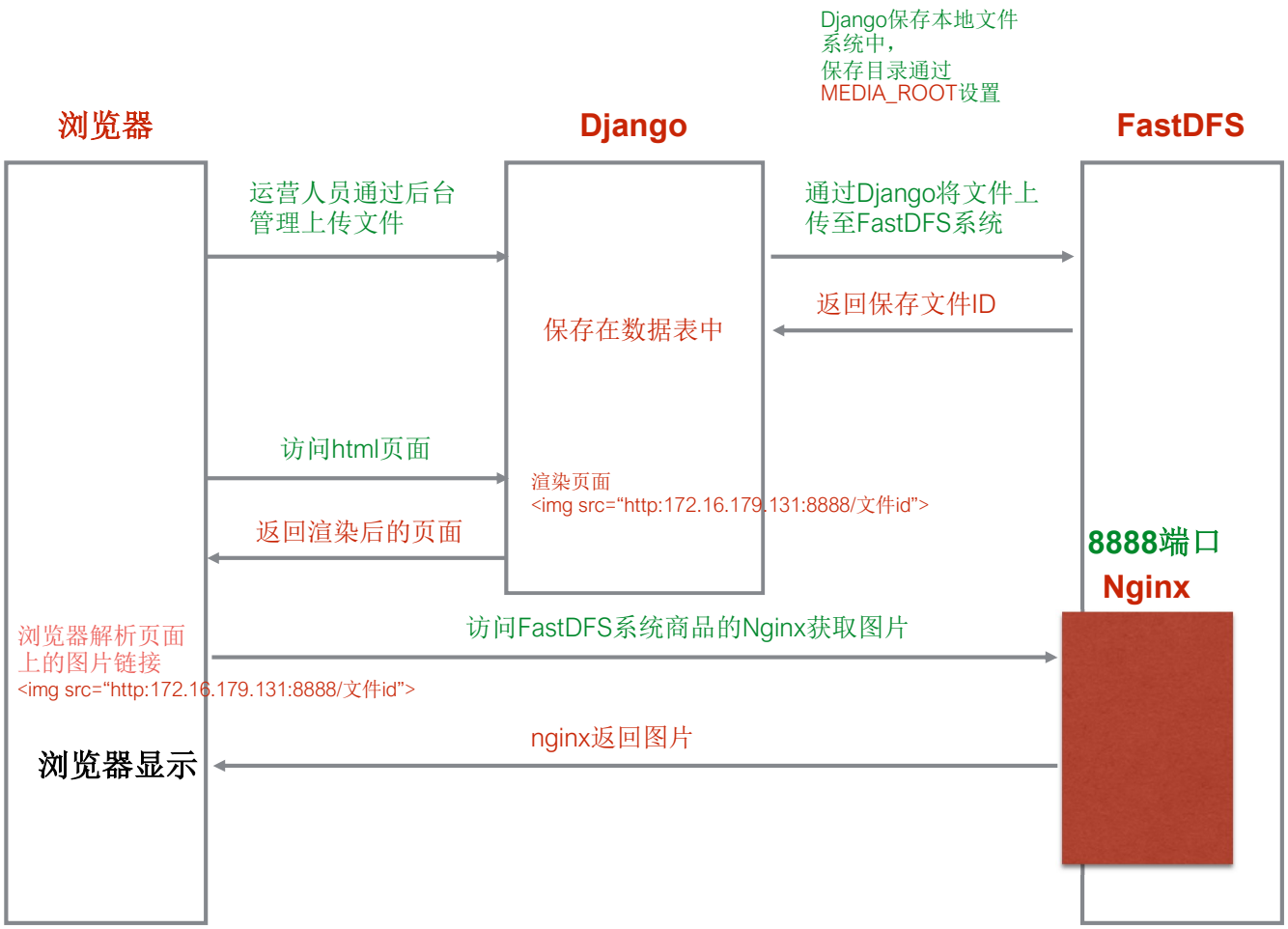
每个用户的历史浏览记录用一条数据保存：
list:
history_用户id:[2，3, 1]
添加历史浏览记录时，用户最新浏览的商品的id从列表左侧插入。
首先先去列表元素中查看用户是否浏览过商品，如果浏览过，先移除元素，再添加到列表的左侧。
否则直接添加到列表左侧。

FastDFS



海量存储，存储容量扩展方便。
文件内容重复。
结合**nginx**提高网站访问图片的效率。

项目处理图片流程


Django保存本地文件
系统中，
保存目录通过
MEDIA_ROOT设置
浏览器
Django
FastDFS
运营人员通过后台
管理上传文件
通过Django将文件上
传至FastDFS系统
返回保存文件ID
保存在数据表中
访问html页面
渲染页面
<img src="http:172.16.179.131:8888/文件id">
返回渲染后的页面
8888端口
Nginx
浏览器解析页面
上的图片链接
<img src="http:172.16.179.131:8888/文件id">
访问FastDFS系统商品的Nginx获取图片
nginx返回图片
浏览器显示

redis存储购物车记录分析

1.什么时候添加购物车记录？
当用户点击加入购物车时需要添加购物车记录。

2. 什么时候需要获取购物车记录?
使用到购物车中数据和访问购物车页面的时候需要获取购物车记录。

3. 使用什么存储购物车记录？
redis存储购物车记录。

4. 分析存储购物车记录的格式?
一个用户的购物车记录用户一条数据保存。
hash:
'cart_用户id': {'sku_id1':商品数目, 'sku_id2':商品数目}
例子：
'cart_1'：｛' 1'：3，' 2'：5｝
获取用户购物车中商品的条目数：
使用hlen

添加购物车记录: sku_id=1, count=2

hgetall

网站性能的优化

1. 页面静态化
2. 数据缓存

生成一个首页的静态页面。
当用户访问首页的时候，直接返回静态页面。

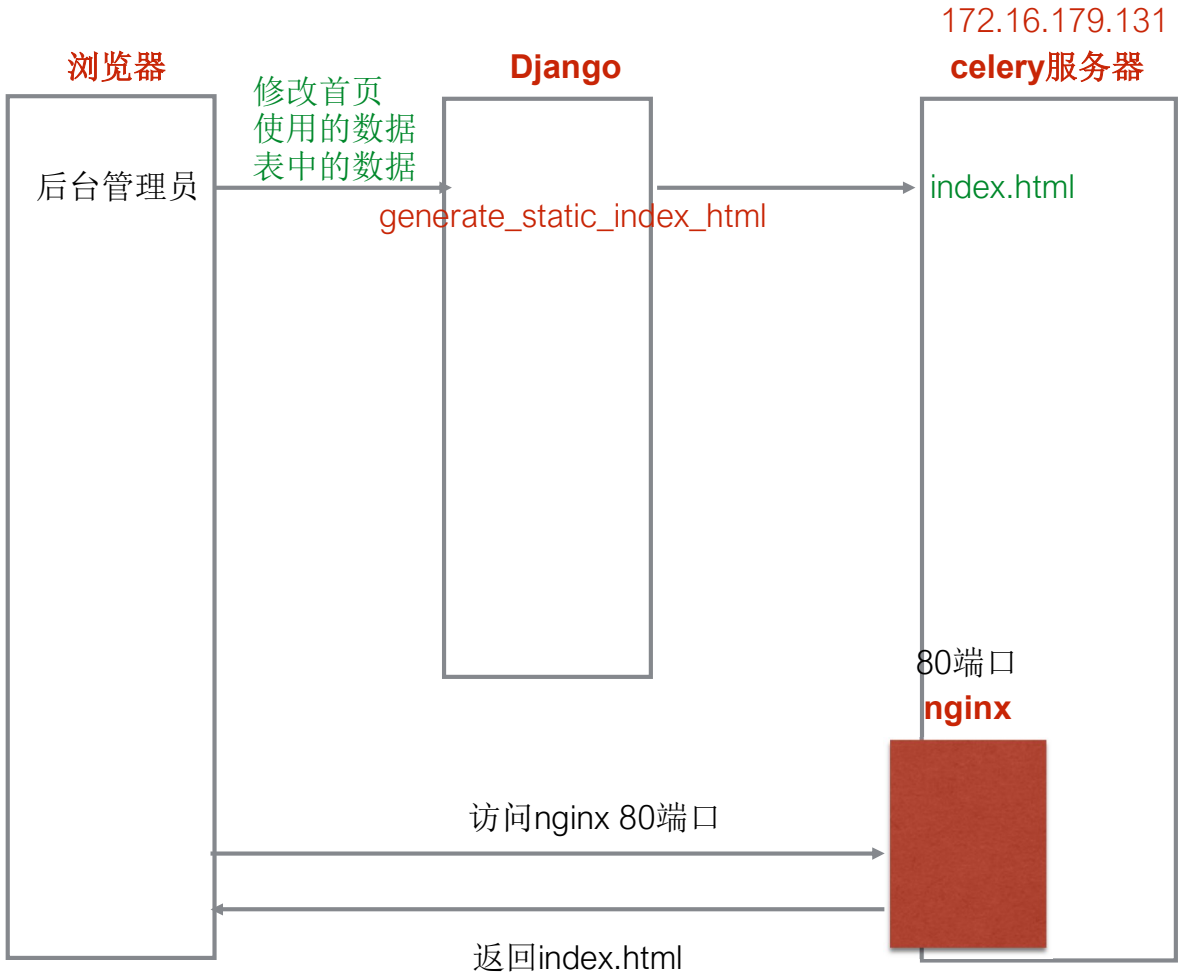
什么时候需要重新生成静态页面？
管理员在后台修改了首页上数据表里的信息的时候，需要重新生成静态页面。
celery

把页面用到的数据缓存起来，如果使用这些数据的时候，先从缓存中获取，
如果获取不到，再去查询数据库。

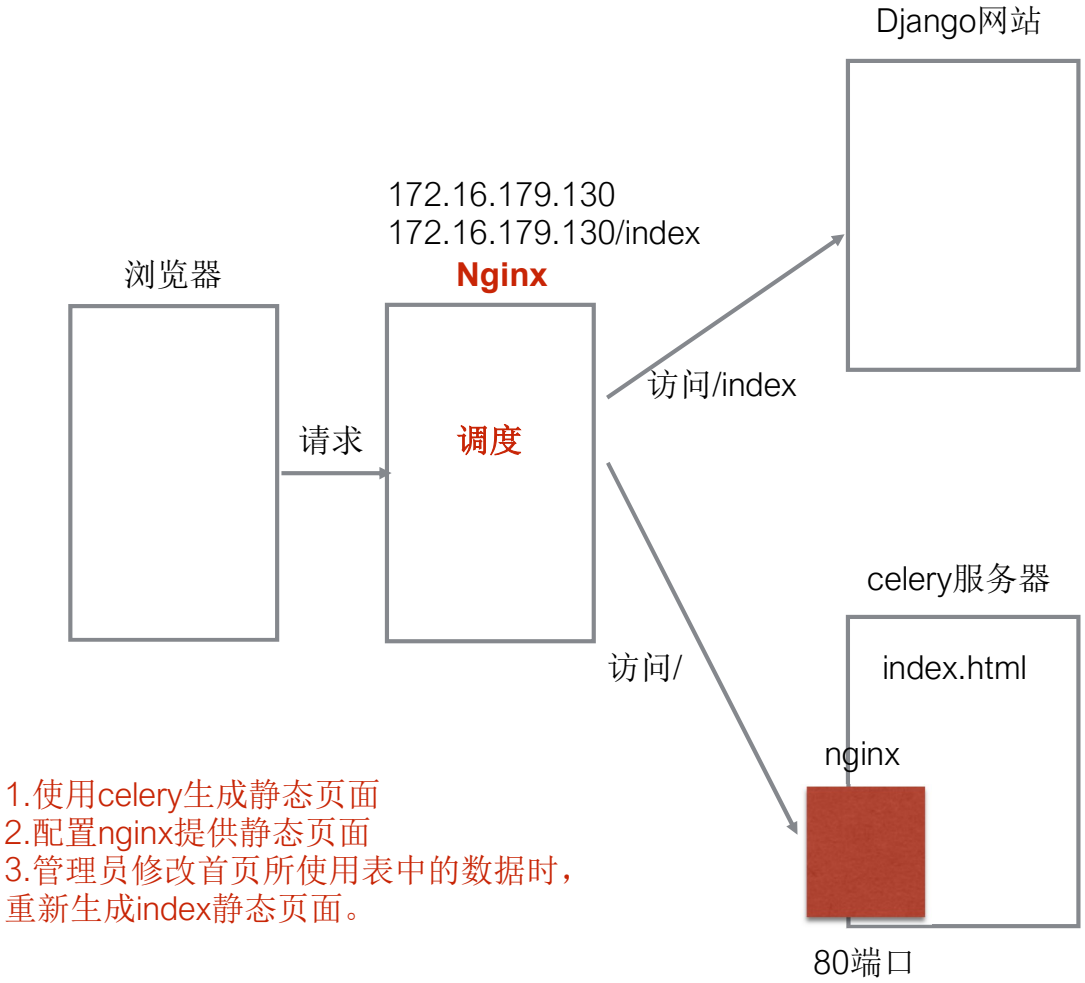
什么时候缓存数据需要更新？
管理员在后台修改了首页上数据表里的信息的时候，需要更新缓存数据。

网站本身性能的优化，减少数据库的查询的次数。
防止恶意的攻击。DDOS攻击

静态页面的生成及Nginx提供静态页面



调度Nginx服务器



1.使用celery生成静态页面
2.配置nginx提供静态页面
3.管理员修改首页所使用表中的数据时，
重新生成index静态页面。

草莓

select * from df_goods_sku where name like ‘%草莓%’ or desc like ‘%草莓%’

搜索引擎:

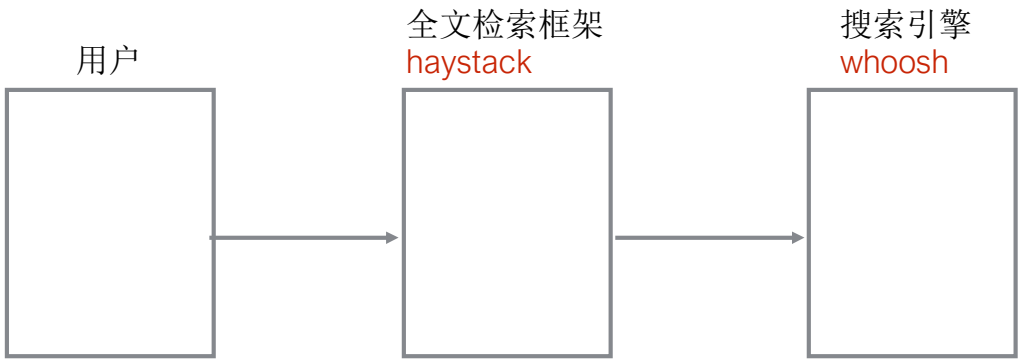
1)可以对表中的某些字段进行关键词分析，建立关键词对应的索引数据。

很
好吃
的
草莓：sku_id1 sku_id2 sku_id5

索引:字典目录

全文检索框架:

可以帮助用户使用搜索引擎。



对搜索结果进行分页。

get传参数:/cart/add?sku_id=1&count=3
post传参数: {‘sku_id’:1, ‘count’:3}
url传参数:url配置时捕获参数

添加商品到购物车:
1）请求方式，采用ajax post
　如果涉及到数据的修改(新增，更新，删除), 采用post
　如果只涉及到数据的获取，采用get
2) 传递参数: 商品id 商品数量

表单中的checkbox,只有被选中时值才会被提交。
request.POST->QueryDict
request.POST.getlist(‘sku_ids’)

订单创建

用户点击提交订单时，需要传递的参数:
收货地址 支付方式 商品id

订单信息表:df_order_info
订单商品表:df_order_goods

用户每下一个订单，就需要向df_order_info表中加入一条记录。
用户的订单中有几个商品，就需要向df_order_goods表中加入几条记录。

订单信息表

订单ID
地址ID
用户ID
支付方式
*总数目
*总金额
运费
支付状态
创建时间

订单商品表

ID
订单ID
sku ID
商品数量
商品价格
评论

创建订单信息记录缺少的参数:
order_id
total_count
total_price
transit_price

{1：5，2：3， 10：2}

鸡腿：stock 1

加锁 select * from df_goods_sku where id=17 **for update**;

事务结束，锁释放。

1.悲观锁
2.乐观锁

在查询数据的时候不加锁，
在更新时进行判断。

判断更新时的库存和之前
查出的库存是否一致。

update df_goods_sku
set stock=0, sales=1
where id=17 and stock=1;

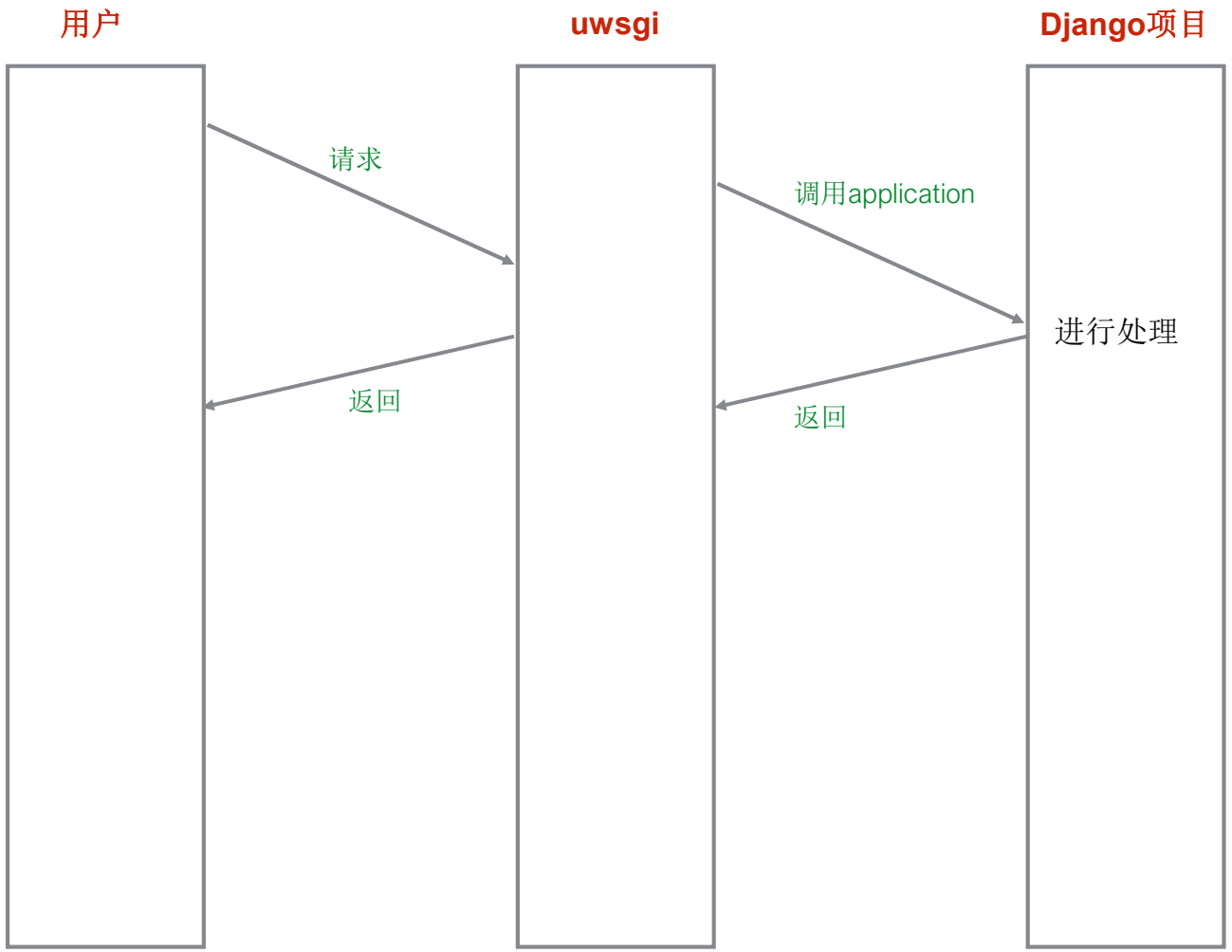
在冲突比较少的时候，
使用乐观锁。

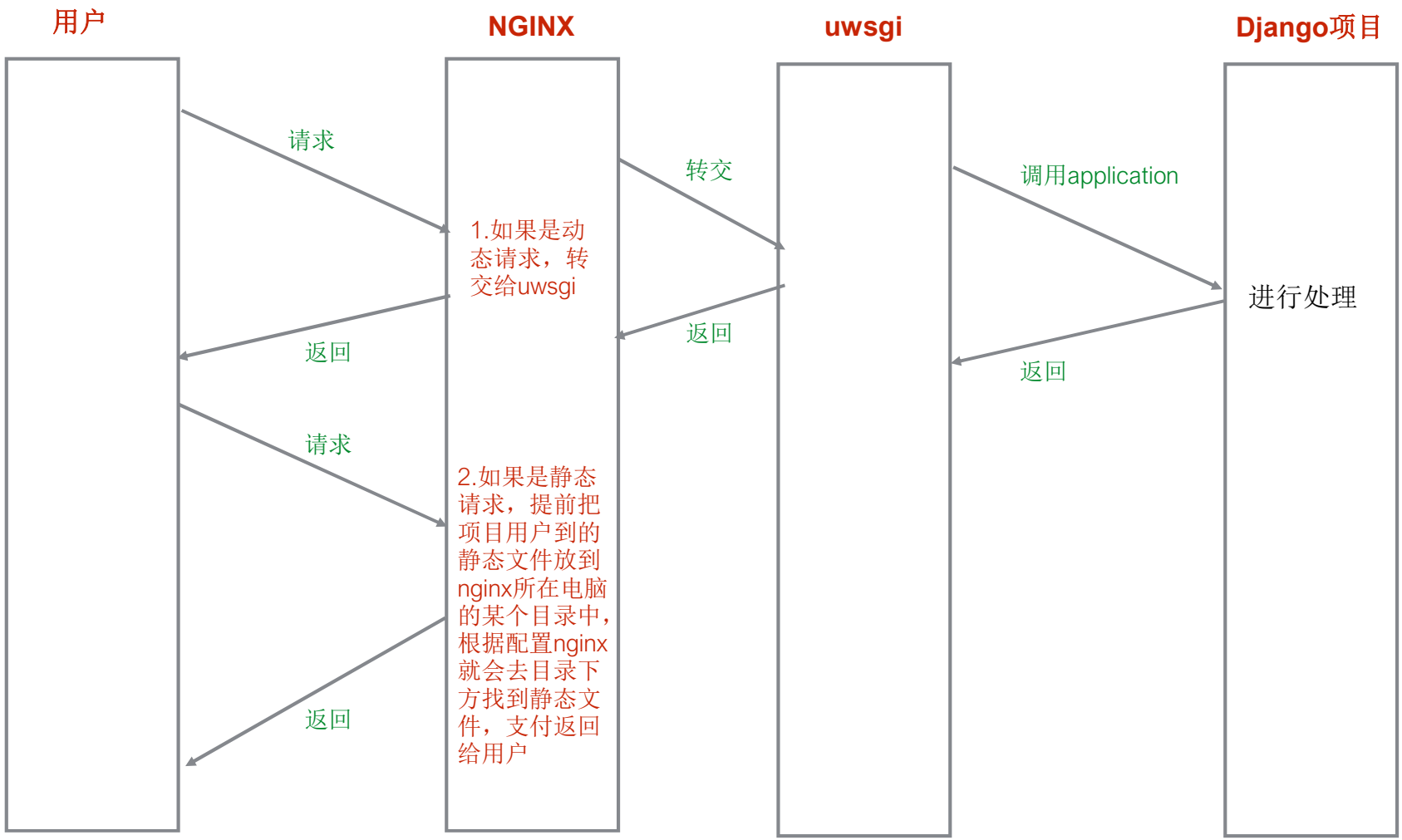
乐观锁重复操作的代价比较大。

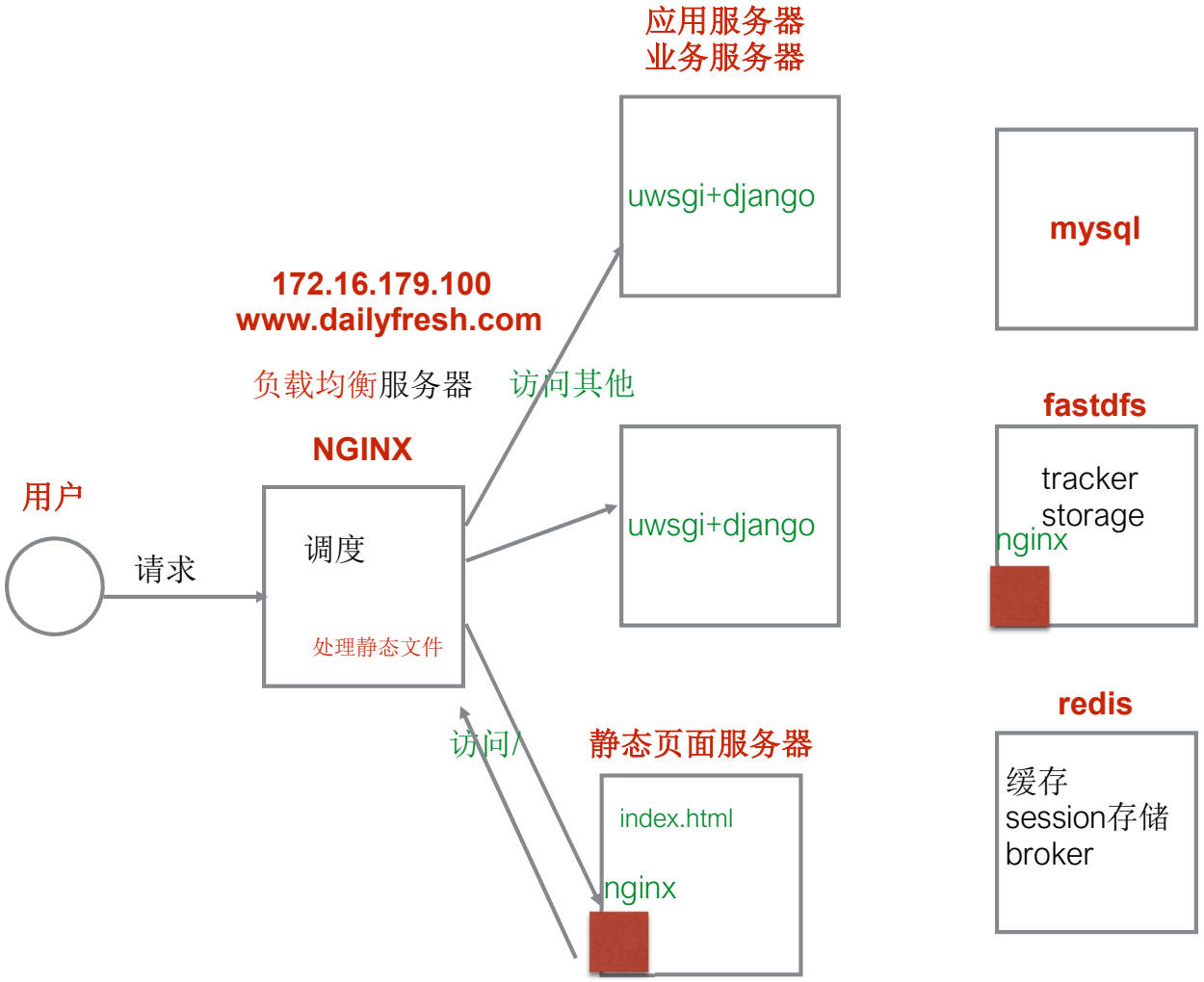
进程1 **用户1**	进程2 **用户2**
向df_order_info 添加一条记录	向df_order_info 添加一条记录
查询sku_id=17的商品信息	查询sku_id=17的商品信息
库存判断	库存判断
向df_order_goods中添加记录	向df_order_goods中添加记录
商品库存更新：0	商品库存更新:0

采用网络请求方式，去请求支付宝平台的地址

网站如果想让支付宝平台访问，需要有公网的ip.


用户
uwsgi
Django项目
请求
调用application
进行处理
返回
返回


用户
NGINX
uwsgi
Django项目
请求
1.如果是动态请求，转交给uwsgi
转交
调用application
进行处理
返回
返回
返回
请求
2.如果是静态请求，提前把项目用户到的静态文件放到nginx所在电脑的某个目录中，根据配置nginx就会去目录下方找到静态文件，支付返回给用户
返回


应用服务器
业务服务器
uwsgi+django
mysql
172.16.179.100
www.dailyfresh.com
负载均衡服务器
访问其他
NGINX
fastdfs
用户
tracker
storage
nginx
请求
调度
uwsgi+django
处理静态文件
redis
访问/
静态页面服务器
缓存
session存储
broker
index.html
nginx