1. 视图

接受请求，进行处理，与M（模型类）和T（模板）进行交互，返回应答。返回html内容HttpResponse,也可能重定向redirect

* 1. 视图函数的使用

url配置项是一个定义在一个名叫urlpatterns的列表中，其中每一个元素就是一个配置项，每一个配置项都调用url函数。

* + 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。参数名称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

* + 1. url配置的过程

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。

在项目的urls中：

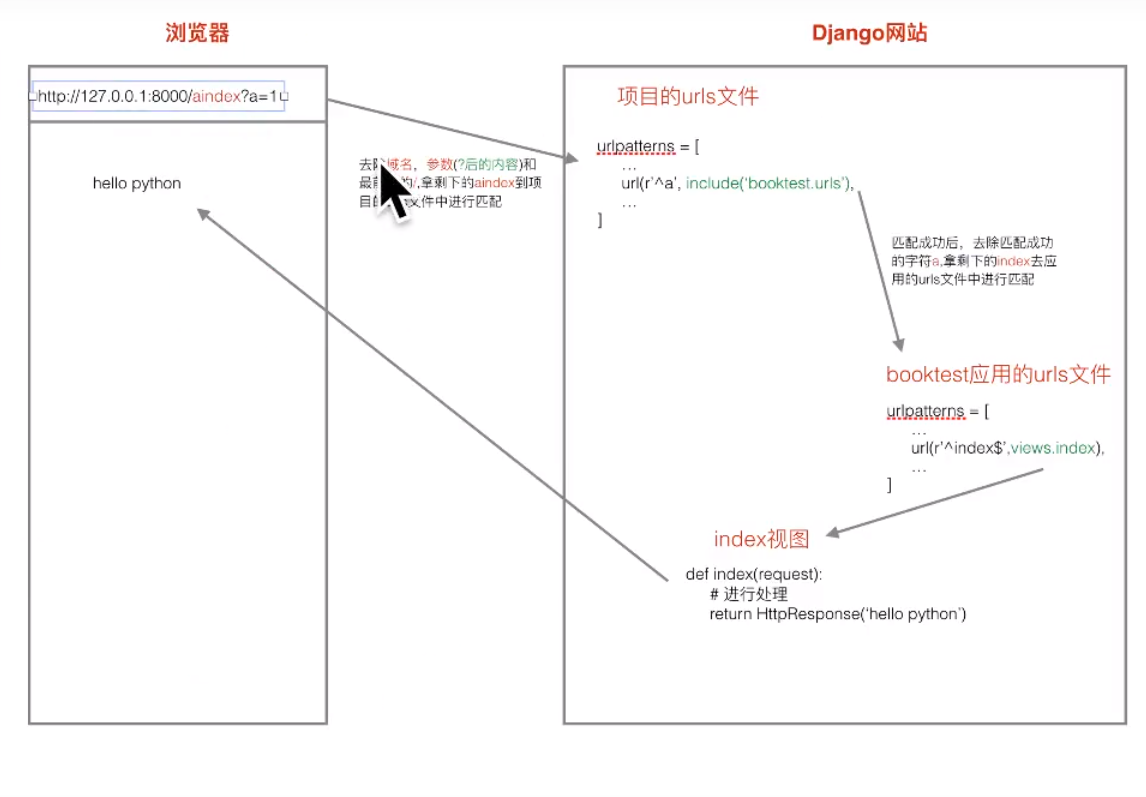
【导入包[from django.conf.urls import url]然后写url(r'^',include('booktest.urls'))】

在应用的urls中：

【导入包url[from django.conf.urls import url] 然后写入url(r'^index/$',views.index),】

* + 1. url匹配过程

注意：？后面不是url地址是给服务器传递的参数，所以加入并不影响页面的访问。



* + - 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ainde想到项目的url中从上到下进行匹配，匹配成功后，去掉已经匹配到的相同的部分，拿剩下的部分index，然后执行后面对应的动作，即跳转到应用的urls文件中，把index再进行从上到下的匹配。
      3. 如果匹配成功则调用对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。
  1. 错误视图

隐藏错误条件，关闭调试模式，设置DEBUG=False（在settings中，开发时默认等于Trus）,然后设置ALLOWED\_HOSTS=[‘\*’]允许所有ip地址访问网站。在templates中创建404.html，可以实现自定义，django会自动调用该页面。该页面中通过request\_path传递错误地址，可以被使用。

* + 1. 404错误原因
       - 1. url没有配置
         2. url配置错误
    2. 500错误：服务器端的错误。

原因：视图函数的代码出错。也有可能是url配置错误

同404错误一样，也可以在templates中自己建立一个500.html页面，实现自定义错误页面，在产生500错误时，django自动调用该页面。

* 1. 捕获url参数
     1. 位置参数，参数名可以随意指定

url(r'^delete(\d+)/$',views.delete),#删除一本图书，（\d+）获得图书的编号，作为参数传递给视图函数，同时视图函数需要定义含有该参数，不然会报错。

* + 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>

如：

rul(r’^showarg(？P<num>\d+)$’,view.show\_arg)

注意：后面视图函数的名字必须和组的名字一样（num）不一样则会报错：获得意外参数。

* 1. 视图参数request的作用

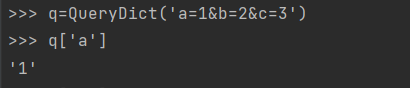
1. request就是HttpRuest的对象，包含浏览器请求/提交的信息。
2. request对象的属性
3. request.POST 保存的是post提交的参数。类型是：QueryDict(查询集是QuertSet)



1. request.GET 保存的是通过get提交的参数。类型同上为QueryDict。

QueryDict字典相似，可以根据键取的对应的值。

1. QueryDic使用：定义一个对象然后根据键盘取对应的额值。如下：





区别：如果用q[‘a’]取值，如果该键不存在则会抛出Keyerror错误，但是如果是用q.get(‘a’)取值，如果键不存在，则不会抛出错误，只是没有显示结果而已。[使用q.get(‘d’,default)，给出默认值，如果d对应有值的话显示出该值，没有的话显示default]。

注意：字典中一个键可以对应多个值，但是QueryDict一个键可以对应多个值。

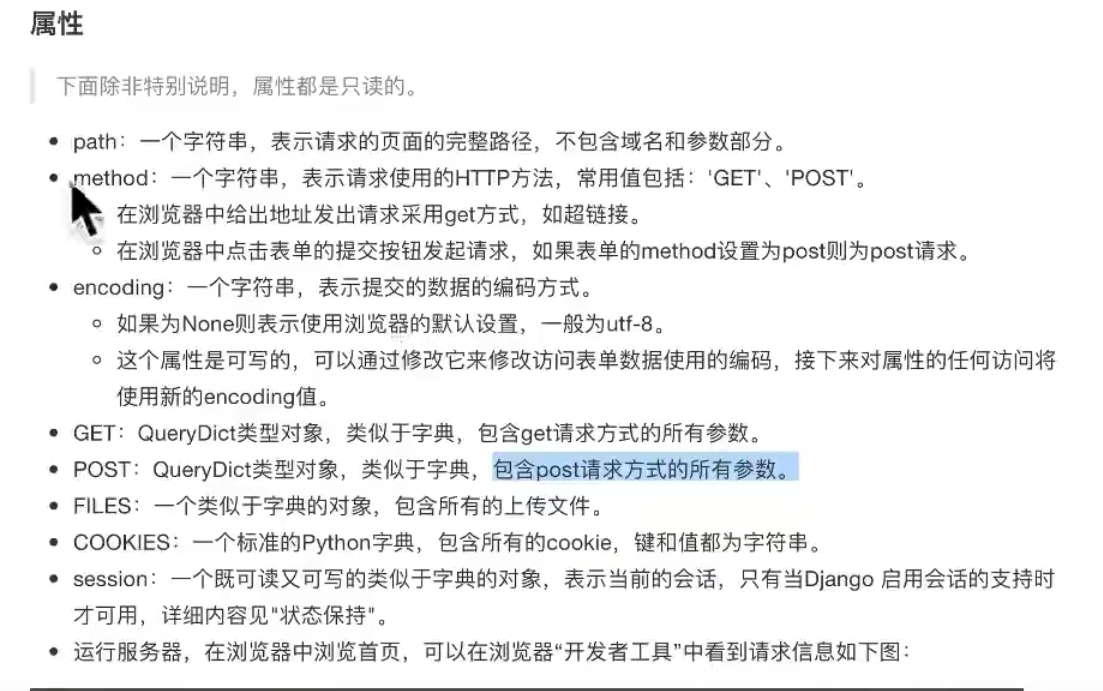
如：



但是，使用以上两种取值方法，只能取出多个值里面最后的那一个，如果要全部取错，则需要使用

【对象.getlist(‘键’)】



1. 

通过浏览器输入的地址是属于get请求方式。

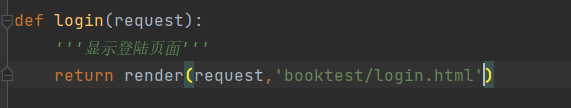
request.path获得访问的地址。

* 1. 登陆案例
     1. 显示登陆页面

1. 设计url通过浏览器访问<http://127.0.0.1:800/login>时显示登陆页面



1. 设计url对应的视图函数login



1. 编写模板文件login.html

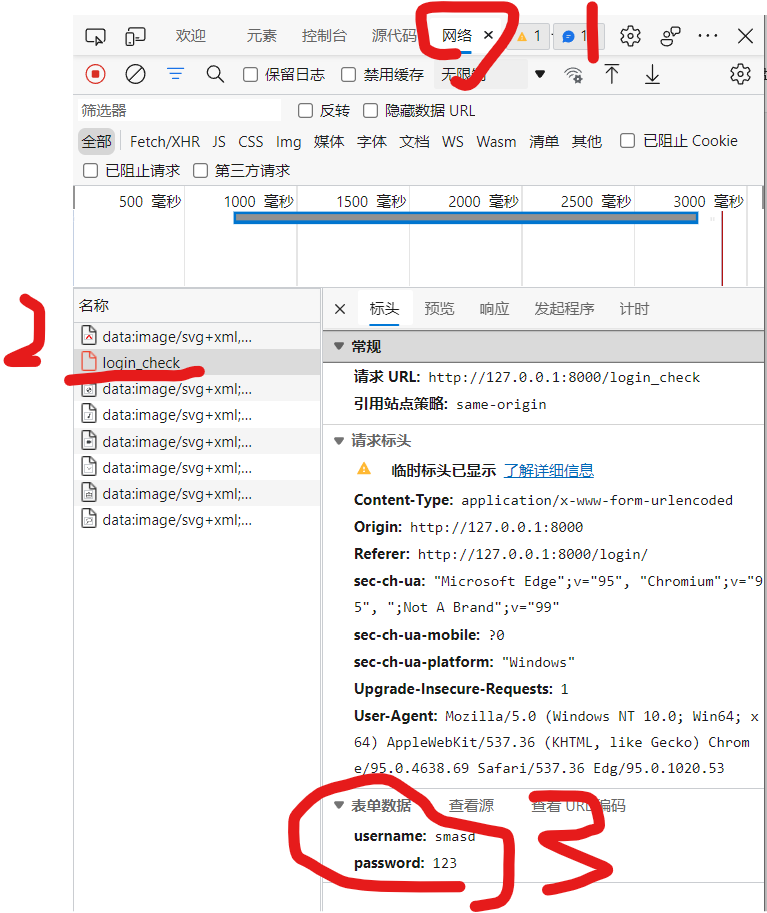


注意：该登陆页面需要设置一个表单，即

<form method =”post/get” action= “/login\_check”>

需要定义表单的提交方式，有两种post和get

post：提交的参数在请求头里面，适合安全性要求较高



get：提交的参数在url中



输入框：标签<input type =”text” name=”username”>[定义了输入框的类型和输入内容的名字（即是QueryDict对象的键，输入的值为该键对应的值，在获取用户名时候，可利用该键取出如数的数据），密码页面同]

登录按钮：<input type=”submit” value=”登录”>[按钮类型为submit]

效果：

url：/login

视图函数：def login

模板文件：login.html

* + 1. 登录校验功能

1. 设计url，实现功能：点击登陆页面的登录按钮发起请求，进入<http://127.0.0.1:8000/login_check>时，进行登录名和密码的校验。
2. 设计url对应的视图函数login——check

实现功能：接受表单提交过来的数据，进行登录校验，若用户名和密码正确则跳转到登陆成功页面。若失败则跳转到登陆页面。redirect

* 1. ajax请求：异步的javascript：

异步解释：发起请求之后，不管回调函数有没有执行，会先执行接下来的内容。默认为异步。【同步ajax请求，添加参数‘async’=false,作用：先执行完服务器的回调函数之后，代码才会继续接下来的内容】



1. 试用情况：在页面刷新之后能够保留用户输入的数据，优化用户的使用体验。
2. 作用:在不重新加载页面的情况下对页面进行局部的刷新。【如在当前页面不动的情况下，浏览器后台请求其他视图函数进行处理然后把获得的数据显示在该页面面上】
3. 发起ajax请求：

$.ajax({

‘url’:请求地址，

‘type’:请求方式(post/get)

‘dataType’:预期返回的数据格式，一般为json

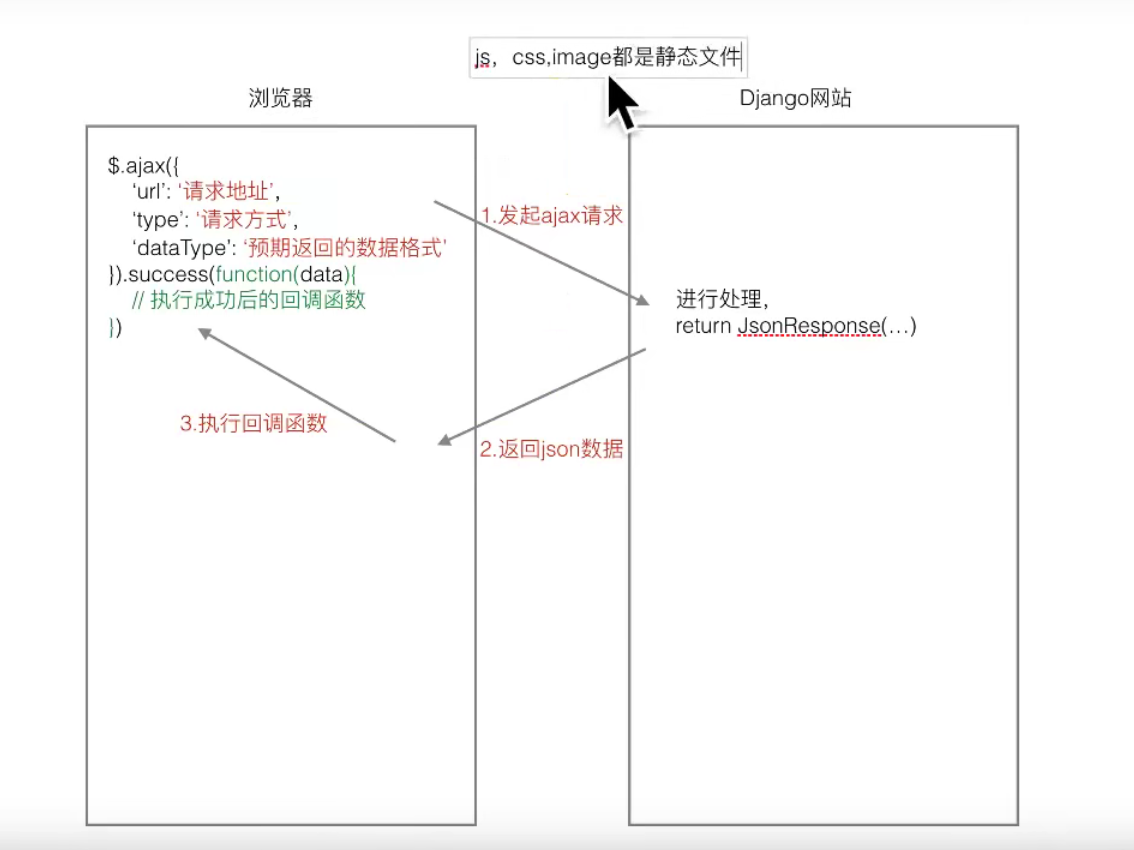
‘data’:参数

}).success(function(data){

#该函数称为执行成功后的回调函数。

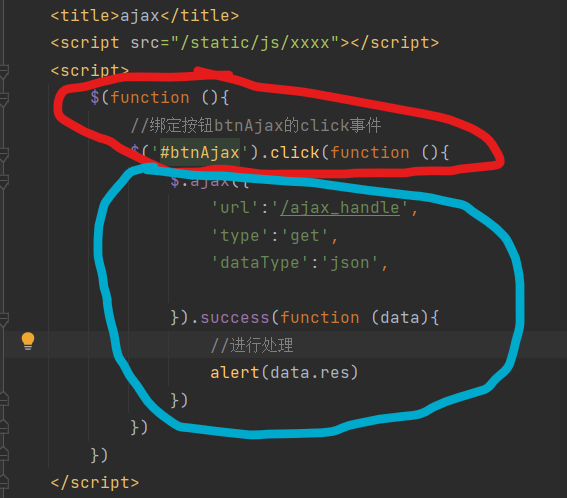
})

1. 具体执行流程
2. 前端发起ajax请求
3. 后台视图进行处理，处理完之后返回json数据
4. 浏览器接受json数据，放入function(data)里面然后执行回调函数。



注意：django里面使用静态文件时，需要创建一个静态文件夹。

1. 在项目的的根目录(应用的根目录)下面新建一个静态文件夹[static]放置静态文件。
2. 在项目的settings.py中设置STATICFILES\_DIRS=[OS.path.join(BASE\_DIR,'static')] #设置静态文件的保存目录。
3. 新建的文件夹中新建三个子目录分别放置：js、css、images文件
4. 在ajax模板文件中，定义发起ajax请求。



红色：为捕获按钮的动作，一点击按钮则执行ajax

蓝色：为ajax主体，执行时，从上到下，向进入其中定义的url地址，在视图函数中进行数据处理，返回一个jason对象，作为回调函数函数的参数。

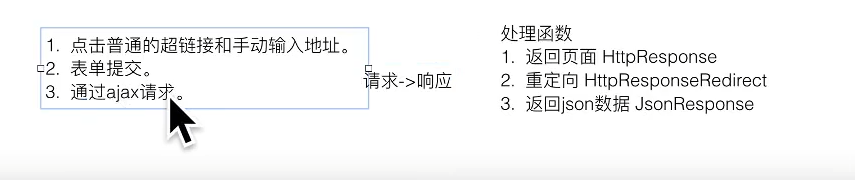
定义一个信息标签：





* 1. ajax登陆案例：利用ajax实现页面部分刷新，

1. 过程



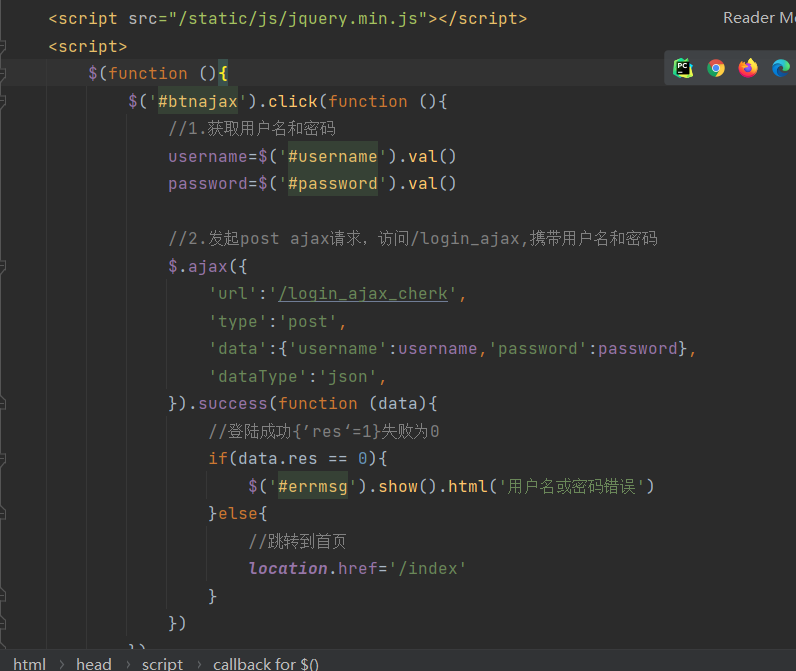
1. 步骤

首先分析出请求地址时需要携带的参数。

(如果是登陆页面则需要携带用户名和密码，利用post)

视图函数处理完成之后，所返回的jason格式，然后再回调函数里面进行判断显示详细内容。

1. 设计登陆页面（/login\_ajax），即视图函数，同时配置对应的url
2. 编写模板文件login\_ajax.html,在里面写jquery发起ajax请求，ajax请求中包含跳转页面的的url

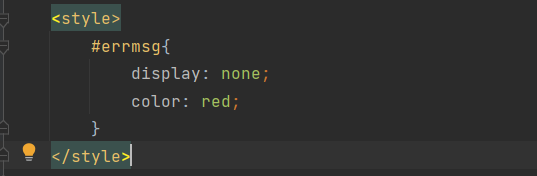


1. 定义跳转页面对应的视图函数login\_ajax\_check，该视图函数要通过request.POST.get拿到账号和密码然后进行校验，最后返回一个json对象，

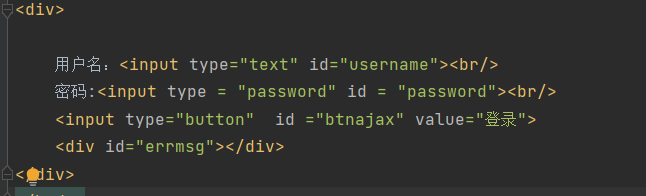
注意：不能直接返回重定向的网页，因为：ajax访问该函数是在后台访问的，所以该番薯返回的对象也在后台，如果返回的是网页，那么这个网页只是在后台运行，没有显示。



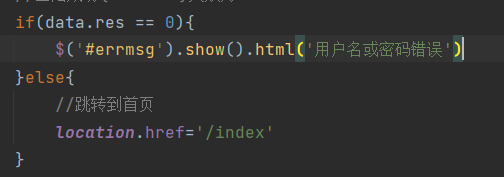
1. ajax拿到校验函数的json对象之后在回调函数中，进行判断，
   1. 如果res==0，则提示登陆密码错误，此时需要在<style>标签里面设置一个errmsg提示信息。



同时需要在<div>中定义



运用代码为：



* 1. res==1则跳转到登录成功页面，跳转方法为：

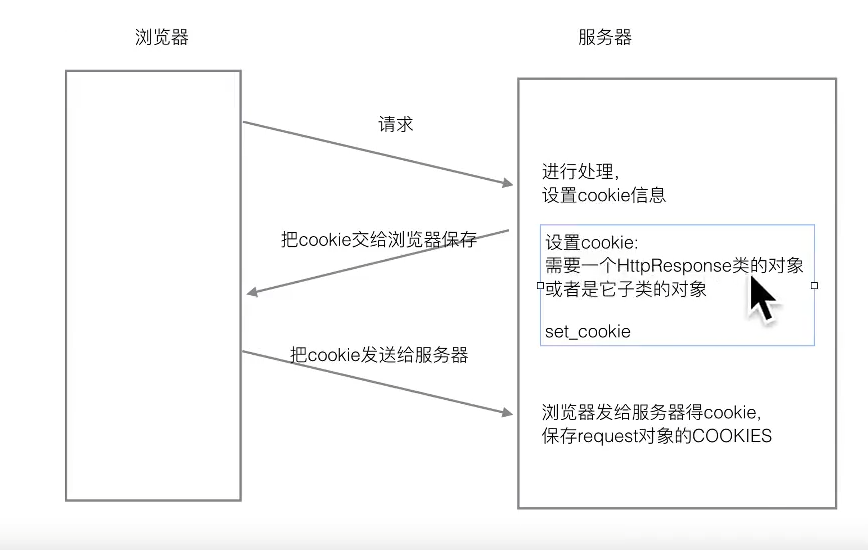


* 1. 状态保持

http协议是无状态的。下一次访问一个页面的时候浏览器并不知道上一次对这个页面做了什么。

1. Cookie

比如：点完餐的时候，服务员会给你一个取餐的单子，只要你拿出这个单子来，服务员就会把你之前点了的东西那给你。所以这个“单子”就相当于Cookie，“我”相当于浏览器，“服务员”相当于服务器。



1. cookie以键值对进行保存，设置cookie需要一个HttpResponse或者是它的子类【HttpResponseRedirect,JsonResponse】的对象。



1. 通过浏览器访问一个网站时，服务器会将浏览器储存的跟网站有关的所有cookie信息发送给该网站服务器。即；

request.COOKIE



1. cookie是基于域名安全的
2. 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。

如果需要关闭浏览器后cookie还保存，则需要设置过期时间：



或者

+

1. 记住用户名案例：

在登陆页面的视图函数中设置cookie



1. Session
2. session的特点：
3. session是以键值对进行储存的。
4. session依赖于cookie。唯一的标识码保存于cookie中
5. session也是有过期时间，如果不指定，默认两周会过期。
6. session对象和方法
7. 以键值对的格式写session

request.session[‘键’]=值

1. 根据键读取值,可以设置默认值，原有值便取值，原无值取默认。

request.session.get(‘键’，默认值)

1. 清除所有的session，在储存表中删除值部分，sessionid仍然保留

request.session.clear()

1. 清除session数据，在存储中删除session的整条数据，sessionid也被删除。

request.session.flush()

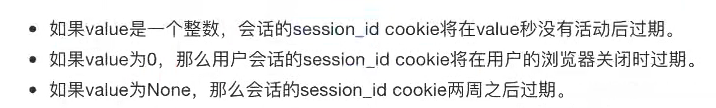
1. 删除session中的指定键及值，在存储中只删除某个键及对应的值

del request.session[‘键’]

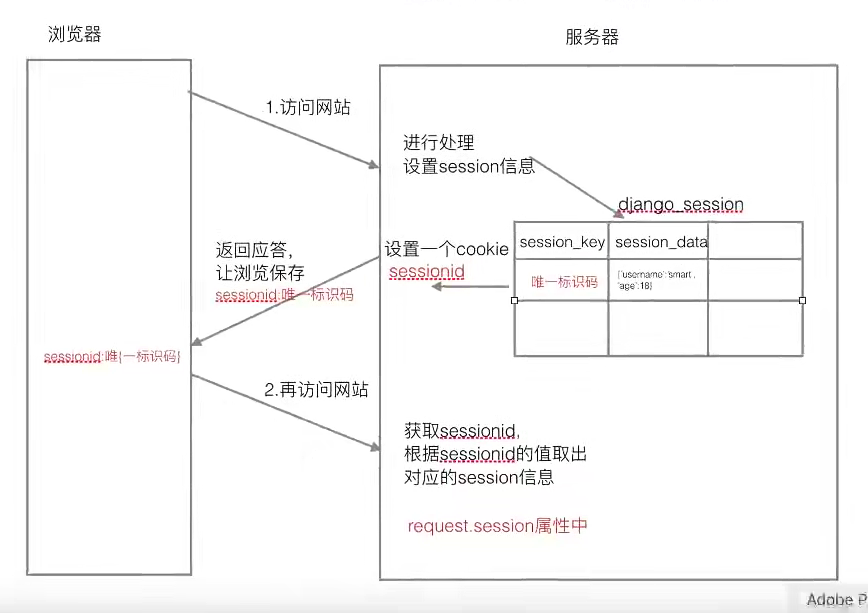
1. 设置会话的超时时间，如果没有指定过期时间则两个星期后过期。

request.session.set\_expiry(value)【cookie是从set值的时候就可以设置过期时间[

response.set\_cookie('num',1,max\_age=14\*24\*3600)]】



* 1. cookie与session的区别。

1. session保存的值的类型是什么取出来的就是什么，而cookie保存的值只能是字符串，取出来也是字符串
2. session是在服务器端调用数据库生成的一个表用来存储用户的数据，返回一个特定的sessionid值，这个值给用户（浏览器）保存在cookie中，下次浏览器要取数据的时候利用cookie调用这个sessionid在session中就可以取出原先保存的数据。而cookie是直接把用户储存的数据保存在cookie中，只要用户一调用cookie就能拿到原先保存的值。
3. 
4. 应用场景
5. cookie：记住用户名，安全性要求不高
6. session：涉及到安全性要求比较高的数据，银行卡账户，密码。