🡪**Django处理一个Http请求的过程**

第一步Django确定使用哪一个根URL Configuration配置。它在settings.py中的ROOT\_URLCONF配置。

第二步是读取该根URL Configuration配置，并且根据扩展includes等机制对配置中的条目逐条对比。

第三步是一旦有条目匹配上，则执行该view函数。并且传递HttpRequest实例， 匹配过程中的命名参数或者位置参数(positional parameter)。

如果没有匹配上则根据Django的错误处理机制处理。

URLCONF中使用位置参数 (positional parameter) 例子:

urlpatterns = [

url(r'^articles/([0-9]{4})/([0-9]{2})/$', views.month\_archive, name=”archive”), # 非命名参数根据位置传入view函数

url(r'^articles/([0-9]{4})/([0-9]{2})/([0-9]+)/$', views.article\_detail, name=”detail”),

]

URLCONF中使用命名参数 (keyword parameter) 例子：

urlpatterns = [

# 命名参数， year, month传入view

url(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$', views.month\_archive, name=”archieve”),

url(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<day>[0-9]{2})/$', views.article\_detail, name=”detail”),

]

🡪**使用includes扩展Django的urls.py**

from django.conf.urls import include, url

urlpatterns = [

# ... snip ...

url(r'^community/', include('django\_website.aggregator.urls')),

url(r'^contact/', include('django\_website.contact.urls')),

# ... snip ...

]

首先我们可以注意到，扩展的前缀r'^community/'上没有$的终止符，并且最后带有/。通过include关键字，Django会把以community开头的访问都重导到django.webiste.aggregator.urls中进行匹配，重导而传入的url地址为去除community以后余下的字符串。所以在'django\_website.aggregator.urls不需要再写community/的前缀。

🡪**url反向匹配。**

不推荐直接在模板或者view中写入hard-coded的url地址。因为维护成本太高，而且容易形成过期的url.Django提供url反向匹配机制。为了实现这种反向匹配机制，必须为在url configuration中为url匹配命名（设置含有name=’xx’）

在views中使用django.urls.reverse函数。reverse(“view函数名称“， args=(函数列表))

例如：

from django.urls import reverse

from django.http import HttpResponseRedirect

def redirect\_to\_year(request):

# ...

year = 2006

# ...

return HttpResponseRedirect(reverse('news-year-archive', args=(year,)))

在模板中使用url来反向匹配 {% url ‘view函数名称’ 参数 %}

例如：

<a href="{% url 'news-year-archive' 2012 %}">2012 Archive</a>

🡪**URLConf的命名空间**

它的作用就是即使不同的APP使用相同的url名称，在URL的命名空间模式下也可以让你得到想要的url地址。

例如：

根URLConf中：

from django.conf.urls import url,include

urlpatterns = [

url(r'^app01/$',include('app01.urls'),namespace='app01'),

url(r'^app02/$',include('app02.urls'),namespace='app02'),

]

app01中的url.py:

from app01 import views

app\_name = 'app01'

urlpatterns = [

url(r'^index01/(?P<para>\d+)/$', views.index\_list, name='index')

]

app02中的url.py:

from app02 import views

app\_name = 'app02'

urlpatterns = [

url(r'^index02/(?P<para>\d+)/$', views.index\_list, name='index')

]

在Views.py以及模板中进行url反向匹配：

{% url 'app01:index' para=666 %}

以及

from django.shortcuts import render,HttpResponse,redirect,reverse

return redirect(reverse('app01:detail',kwargs={'para':666}))

🡪