Django绝对简明手册(写作中)

张沈鹏　　电子科技大学　生物医学工程/计算机科学与技术
你看到的此文档,可能不是最新的,
欢迎访问我的 Blog 了解最新的变化.
也欢迎加入我的Google讨论群,
讨论一切关于
C++,STL,Boost,
XML,CSS,Javascript,XUL
Python,Django
的问题

我的Google讨论群 我的Blog

-- Any Question,Please Email To zsp007@gmail.com

更新:2007.6 beta

- Django版本:9.6
- Python版本:2.5

目录

1. 序言

现在学的东西很容易忘记,写这篇文章的目的是能让我在需要时快速找回当时的感觉. Find Fun !

内容大部分摘自hideto翻译的Django Book,在此感谢

2. 辅助工具

文本替换专家2.5 : 修改站名,APP模块名时用得到

3. View函数

向浏览器输出html等的函数.

3.1. 直接输出

```
from django.http import HttpResponse
import datetime

#View函数的第一个参数总是HttpRequest对象
#offset是一个string，值由url正则表达式匹配而得。
def hours_ahead(request, offset):
   offset = int(offset)
```

```
    dt = datetime.datetime.now() + datetime.timedelta(hours=offset)
    html = "In %s hour(s), it will be %s." % (offset, dt)
    return HttpResponse(html)
```

3.2. 泛型视图

3.2.1. 渲染模板

django.views.generic.simple.direct_to_template

渲染一个给定的模板，并把在URL里捕获的参数组成字典作为

模板变量传递给它

```
(r'^foo/(?P<id>\d+)/$', 'direct_to_template', {'template':
'foo_detail.html'}),
```

3.2.2. 重定向到另一URL

```
#如果url对应的是None则返回410 HTTP(不可用)错误
('^foo/(?p<id>\d+)/$', 'redirect_to', {'url': '/bar/%(id)s/'}),
```

3.2.3. 复杂视图概览

- List/detail视图，它提供对象列表和单独对象细节的页面(例如地点列表和单独的一个地点的信息页面)
- Date-based视图，它提供year/month/day样式的以日期为中心的信息页面
- Create/update/delete视图，它允许你快速创建增，删，改对象的视图

通用的可选参数

- allow_empty

一个布尔值，指定没有对象时是否显示页面，如果它是False并且没有对象，视图将触发404错误而不是显示空页面，默认是False

- context_processors

一个视图模板的template-context processors列表，参考第10章得到更多template context processors的信息

- extra_context

一个添加到模板context的字典值，它默认为空字典，如果字典中的一个值可以调用，generic view将在渲染模板前调用它

- mimetype

用来生成结果文档的MIME类型，默认为DEFAULT_MIME_TYPE设置的值

- template_loader

当载入模板时使用的模板载入器，默认为django.template.loader

- template_name

渲染页面时使用的完整的模板名，它可以让你覆盖来自于QuerySet的默认模板名

- template_object_name

指定在模板context中的模板变量名，默认为'object'，视图列表列出不止一个对象时将在此变量值后添加'_list'

3.2.4. 显示+分页object_list

django.views.generic.list_detail.object_list视图用来创建一个显示对象列表的页面

```
from django.conf.urls.defaults import *
from django.views.generic import list_detail, date_based,
create_update
from bookstore.models import Publisher, Author, Book
author_list_info = {
'queryset' :   Author.objects.all(),
'allow_empty': True,
}
urlpatterns = patterns('',
  (r'authors/$', list_detail.object_list, author_list_info)  ,
)
```

我们只需为generic view制作一个模板来渲染即可 可选的参数:

paginate_by

你个指出每一页多少对象应该被显示的整数，如果这项被给定，视图将使用paginate_by分页视图将希望得到一个包含了从零开始索引页数的page查询字符串参数(通过GET)，或者一个在URL配置里指定的page变量.

template_name: 模板名没有被指定，视图将默认使用(app_label)/(model_name)_list.html.

app标签和模型名字两者都来自于queryset参数，app标签是模型定义所在的app的名字，模型名字则是模型类的名字的小写版本

所以当我们把Author.objects.all()作为queryset传递时，app标签将是bookstore，模型名则是author

这意味着默认的模板将是bookstore/author_list.html

模板context

除了extra_context，模板的context将包括:

object_list

对象的列表，这个变量的名字取决于template_object_name参数，而后者默认是'object' 如果template_object_name是'foo'，则这个变量的名字就是foo_list

is_paginated

一个boolean值，它表示结果是否分页

特别的，当可得到的对象的数量小于或等于paginate_by时，它的值被设为False

如果结果是分页的，context将包含以下额外变量

results_per_page 每页对象的数量(和paginate_by参数一样)

has_next 一个boolean值，它表示是否有下一页

has_previous 一个boolean值，它表示是否有上一页

page 当前页的页数，它是一个整数，从1开始

next 下一页的页数，它是一个整数，如果没有下一页，它还是一个整数并表示理论上的下一页页数，从1开始

previous 上一页的页数，它是一个整数，从1开始

pages 总页数，它是一个整数

hits 所有页面的对象的总数，不仅仅是当前页

3.2.5. 细节视图object_detail

像下面这样提供一个info字典:

```
author_detail_info = {
"queryset" : Author.objects.all(),
"template_object_name" : "author",
}
```

URL模式如下

```
(r'^authors/(?P<object_id>\d+)/$', list_detail.object_detail,
```

```
author_detail_info),
```

必需的参数

queryset

object_id

或者

slug

slug_field 对象中包含slug的域的名字，如果你使用slug参数，则它是必需的，如果你使用object_id参数则不能要

template_name_field 如果你的对象有一个'the_template'域(属性)并且该域包含一个字符串'foo.html'，你把template_name_field设置为the_template，则这个对象的generic view将使用模板'foo.html'

除了extra_context，模板的context是

object

对象，这个变量的名字取决于template_object_name参数，默认是'object'，如果template_object_name是'foo'，这个变量的名字就是foo

3.2.6. 日期显示最近

django.views.generic.date_based_archive_index视图提供了一个通过日期显示最近的对象的顶级首页 一个典型的publisher可能想把最近发表的books设为语法高亮，我们可以使用archive_index视图来做这件事，下面是info dict

book_info = { "queryset" : Book.objects.all(), "date_field" : "publication_date" }

相应的URL配置: (r'^books/$', date_based.archive_index, book_info)

必需的参数:

date_field 在QuerySet的模型中的DateField或者DateTimeField的名字，基于日期的存档使用它来决定页面上的对象

queryset 存档处理的QuerySet的对象

可选参数:

allow_future 一个布尔值，它指定是否在这个页面引入"未来"的对象

num_latest 传递到模板context的最近的对象数目，默认为15 模板的context为如下列表:

date_list datetime.date对象的列表，表示对应queryset的所有具有对象的years，他们排反序

例如，如果你有一个从2003到2006的blog列表，这个列表将包含4个datetime.date对象:每年一个latest

系统中的num_latest个对象，通过date_field排倒序

例如如果num_latest为10，latest将为queryset中的最近的10个对象

4. Url

4.1. 匹配参数

```
(r'^now/plus\d+hours/$', hours_ahead)
```

可以匹配

```
/now/plus2hours/
/now/plus125hours/
/now/plus100000000000hours/
```

限制最多99小时，即只允许1个或2个数字

```
(r'^now/plus(\d{1,2})hours/$', hours_ahead)
```

4.2. 常用正则式

```
.     任意字符
\d          任意数字
[A-Z]       从A到Z的任意字符(大写)
[a-z]       从a到z的任意字符(小写)
[A-Za-z]    从a到z的任意字符(大小写不敏感)
[^/]+       任意字符直到一个前斜线(不包含斜线本身)
+           一个或多个前面的字符
?           零个或多个前面的字符
{1,3}       1个到3个之间前面的字符(包括1和3)
```

4.3. 命名变量

传给函数以year命名的变量

```
(r'^articles/(?P<year>\d{4})/$', views.year_archive),
```

4.4. 指定参数

传递参数,动态选择模板,有时配合默认参数会很方便 urls.py

```
from django.conf.urls.defaults import *
from mysite import views
urlpatterns = patterns('',r'^foo/$', views.foobar_view,
{'template_name': 'template1.html'}),
(r'^bar/$', views.foobar_view, {'template_name': 'template2.html'}),
)
```

views.py

```
from django.shortcuts import render_to_response
from mysite.models import MyModel
def foobar_view(request, template_name):
        m_list = MyModel.objects.filter(is_new=True)
        return render_to_response(template_name, {'m_list': m_list})
```

4.5. include URL配置

注意:指向inclue()的正则表达式不包含$(字符串结尾匹配符)，但包含一个末尾斜线

```
 (r'^photos/', include('mysite.photos.urls')),
```

include时可以附加额外的参数

```
#额外选项将一直传递给include的URL配置的每一行,仅当你确认在include的URL配置里每个视图接受你传递的选项时才有意义
(r'^blog/', include('inner'), {'blogid': 3}),
```

5. 模版

5.1. 传入参数

```
from django.template import Context, Template
t = Template("My name is {{name}}.")
c = Context({"name": "Stephane"})
t.render(c)
```

输出

```
'My name is Stephane.'
```

渲染的对象还可以是,字典

```
 person = {'name': 'Sally', 'age': '43'}
```

对应的模板为

```
{{ person.name }} is {{ person.age }} years old.'
```

类

```
class Person(object):
        def __init__(self,first_name,last_name):
                self.first_name, self.last_name = first_name,
last_name
```

对于的模板和上面类似

对于类似 Contexst({'items': ['apples', 'bananas', 'carrots']}) 的数组 items.2就表示 carrots

默认情况下如果变量不存在，模板系统会把它渲染成空string

5.2. 调用对象的方法

如对于Context({'var': '123ss'})可以在模板中用var.upper来调用upper()方法,被调用的方法不能有参数

如果一个方法触发了异常,会致渲染失败.

如果异常有一个值为True的silent_variable_failure属性，这个变量会渲染成空string

```
class SilentAssetionError(AssertionError):
        silent_variable_failure = True

class PersonClass4:
        def first_name(self):
                raise SilentAssertionError

p = PersonClass4()

#将被渲染为空字串
t.render(Context({"person": p}))
```

设置方法的alters_data属性为true可以禁止在模板中调用该方法

```
def delete(self):
        pass
delete.alters_data = True
```

5.3. 插入变量

```
<p>Sincerely,<br />{{ 变量名 }}</p>
```

5.4. 块语句

5.4.1. if

{% if %}标签不允许同一标签里同时出现and和or

可以使用嵌套的{% if %}来表示and和or

```
{%if xxx%}
        <p>内容1</p>
{%else%}
        <p>内容2</p>
{%endif%}
```

5.4.2. ifequal/ifnotequal

```
{% ifequal user currentuser %}
<h1>Welcome!</h1>
{% endifequal %}
```

参数可以是硬编码的string,数字等,如{% ifequal section 'sitenews' %},{% ifequal section 1.3 %}

5.4.3. for

{% for %}标签允许你按顺序遍历一个序列中的各个元素

```
<ul>
{% for athlete in athlete_list %}
        <li>{{ athlete.name }}</li>
{% endfor %}
</ul>
```

在标签里添加reversed来反序循环列表,如{% for athlete in athlete_list reversed %}

{% for %}标签内置了一个forloop变量

- forloop.counter表示循环的次数，从1开始计数
- forloop.counter0类似于forloop.counter，但它是从0开始计数
- forloop.revcounter表示循环中剩下的items数量，第一次循环时设为items总数，最后一次设为1
- forloop.revcounter0类似于forloop.revcounter，但它是表示的数量减一，即最后一次循环时设为0
- forloop.first当第一次循环时值为True，在特别情况下很有用

```
{% for object in objects %}
        {% if forloop.first %}<li class="first">{% else %}<li>{%
endif %}
                {{ object }}
        </li>
{% endfor %}
```

- forloop.last当最后一次循环时值为True
- forloop.parentloop在嵌套循环中表示父循环的forloop

5.4.4. include

{% include 'includes/nav.html' %}

如果给定的模板名不存在，Django将做下面两件事情中的一件：

1，如果DEBUG设置为True，你将看到一个TemplateDoesNotExist异常的错误页面

2，如果DEBUG设置为False，标签将什么也不显示

5.4.5. 注释

{# This is a comment #} 模板渲染时注释不会输出，一个注释不能分成多行

5.4.6. 使用模板文件

```
from django.template.loader import get_template
from django.template import Context
from django.http import HttpResponse
import datetime
def current_datetime(request):
        now = datetime.datetime.now()
        t = get_template('current_datetime.html')
        html = t.render(Context({'current_date': now}))
        return HttpResponse(html)
```

或者更直接

```
from django.shortcuts import render_to_response
import datetime
def current_datetime(request):
        now = datetime.datetime.now()
        return render_to_response('current_datetime.html',
{'current_date': now})
```

如果你很懒或者你想保持代码整洁，使用Python内建的locals()方法返回一个包含当前作用域里面的所有变量和它们的值的字典

locals()导致了一点点开销，因为Python不得不动态创建字典,如果你手动指定context字典则可以避免这项开销

```
def current_datetime(request):
        current_date = datetime.datetime.now()
        return render_to_response('current_datetime.html', locals())
```

5.5. 过滤器

5.5.1. date

把ship_date变量传递给过滤器,并给date过滤器传递了一个参数“F j, Y”，date过滤器以给定参数的形式格式化日期

```
{{ship_date|date:"F j, Y"}}
```

5.5.2. escape / linebreaks

escape转义给定的string里出现的&符，引号，尖括号

常用于处理用户提交的数据和确认合法的XML和XHTML数据

escape文本内容然后把换行转换成p标签

```
{{ my_text|escape|linebreaks }}
```

显示bio变量的前30个字，过滤器参数一直使用双引号

```
{{ bio|truncatewords:"30" }}
```

5.5.3. addslashed

在任何后斜线，单引号，双引号前添加一个后斜线,常用于输出文本到JavaScript字符串

5.5.4. length

返回值的长度，你可以在一个list或string上做此操作

6. Model数据库

```
from django.db import models

class Author(models.Model):
        name = models.CharField(maxlength=30)
    headshot = models.ImageField(upload_to='/tmp')
    email = models.EmailField()

    def __str__(self):
                return self.name

class Publisher(models.Model):
        website = models.URLField()

class Book(models.Model):
        authors = models.ManyToManyField(Author)  #多对多
        publisher = models.ForeignKey(Publisher)  #1对1
        publication_date = models.DateField()
```

属性名对应列名

属性的类型(如CharField)对应数据库列类型

除非你自己定义一个主键，Django会自动为每个模型生成一个integer主键域id

每个Django模型都必须有一个单列的主键

```
p = Publisher(website='http://www.apress.com/')
p.save()
publisher_list = Publisher.objects.all()
```

DjangoZipManual (2009-12-25 07:10:12由localhost编辑)