分页

一、示例

1、分页示例:

```
from django.core.paginator import Paginator

objects = ['john', 'paul', 'george', 'ringo']
p = Paginator(objects, 2)
```

2、注意:
你可以向 Paginator 提供一个列表、元组或Django的QuerySet,或者任何带有count()或**len**() 方法的对象。**当计算传入的对象所含对象的数量时, Paginator会首先尝试调用 count(),接着如果传入的对象没有 count() 方法则回退调用 len() 。**这样会使类似于Django的QuerySet的对象使用更加高效的count()方法,如果存在的话。

二、使用Paginator

分页视图示例:

```
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

def listing(request):
    contact_list = Contacts.objects.all()
    paginator = Paginator(contact_list, 25)

    page = request.GET.get('page')
    try:
        contacts = paginator.page(page)
    except PageNotAnInteger:
        # If page is not an integer, deliver first page.
        contacts = paginator.page(1)
    except EmptyPage:
        # If page is out of range (e.g. 9999), deliver last page of results.
        contacts = paginator.page(paginator.num_pages)

    return render_to_response('list.html', {"contacts": contacts})
```

三、Paginator objects

1、Paginator 类拥有以下构造器:

```
class Paginator (object_list, per_page, orphans=0,
                         allow_empty_first_page=True)[source]
```

2、所需参数

object_list
使用count()或**len**()方法的列表、元组、Django QuerySet或其他可分割的对象。

per_page
在一个页面中包含的条目的最大数量，不包括孤儿(参见下面的可选参数)。

3、可选参数

orphans
在最后一页上允许的最小项数，默认为零。当你不想要最后一页的时候，就用这个。如果最后一页通常有一些小于或等于orpahans的条目，那么这些条目将被添加到上一页(这将成为最后一页)，而不是将这些条目单独留在页面上。例如，有23项，per_page=10，orphans=3，将有两页;第一页有10个条目，第二个(和最后一页)有13个条目。

allow_empty_first_page
是否允许第一个页面为空。如果False和object_list是空的，那么将会增加一个EmptyPage错误。

4、方法

Paginator.page (number)[source]
返回在提供的下标处的 Page 对象,下标以1开始。如果提供的页码不存在,抛出 InvalidPage 异常。

5、属性

Paginator.count
所有页面的对象总数。
注意:
当计算 object_list 所含对象的数量时, **Paginator 会首先尝试调用 object_list.count() 。如果 object_list 没有 count() 方法, Paginator 接着会回退使用 len(object_list)。**这样会使类似于Django's QuerySet的对象使用更加便捷的count()方法,如果存在的话。

Paginator.num_pages
页面总数。

Paginator.page_range
页码的范围,从1开始,例如 [1, 2, 3, 4]。

四、Invalidpage exception

exception InvalidPage [source]
异常的基类,当paginator传入一个无效的页码时抛出。

通常,捕获 InvalidPage 异常就够了,但是如果你想更加精细一些,可以捕获以下两个异常之一:

exception PageNotAnInteger [source]
当向 page() 提供一个不是整数的值时抛出。 InvalidPage 的子类,所以你可以通过简单 的 except InvalidPage 来处理它们。

exception EmptyPage [source]
当向 page() 提供一个有效值,但是那个页面上没有任何对象时抛出。 InvalidPage 的子类,所以你可以通过简单 的 except InvalidPage 来处理它们。

五、Page objects

class Page (object_list, number, paginator)[source]
通常不需要手动构建 Page 对象 -- 你可以从 Paginator.page()来获得它们。

1、方法

Page.has_next ()[source]
如果有下一页，返回True。
Page.has_previous ()[source]
如果有上一页,返回 True 。

Page.has_other_pages ()[source]
如果有上一页或下一页,返回 True 。

Page.next_page_number ()[source]
返回下一页的页码。如果下一页不存在,抛出 InvalidPage 异常。
Page.previous_page_number ()[source]
返回上一页的页码。如果上一页不存在,抛出 InvalidPage 异常。

Page.start_index ()[source]
返回当前页上的第一个对象,相对于分页列表的所有对象的序号,从1开始。比如,将五个对象的列表分为每页两个对象,第二页的 start_index() 会返回3。
Page.end_index ()[source]
返回当前页上的最后一个对象,相对于分页列表的所有对象的序号,从1开始。 比如,将五个对象的列表分为每页两个对象,第二页的 end_index() 会返回4 。

2、属性

Page.object_list
当前页上所有对象的列表。

Page.number
当前页的序号,从1开始。

Page.paginator
相关的 Paginator 对象。