Django2.0 path

作者：alaikis
原文链接：https://ld246.com/article/1518632846392
来源网站：链滴

2017 年 12 月 2 号，Django2.0 发布！！！ 所以之前 1.8 版本的已经有所改变。
其中 url 变成 path。
9 月 23 日 Django 发布了 2.0a1 版本，这是一个 feature freeze 版本，如果没有什么意外的话 2.0 正式版不会再增加新的功能了。按照以往的规律，预计正式版将在 12 月发布。
2.0 无疑是一个里程碑版本，因为这是第一个只支持 Python3.X 的版本，和 1.x 是不兼容的。
What’ s new in Django2.0 文档中一共列出了三个新的特性：

更简单的 URL 路由语法 (Simplified URL routing syntax)
admin 应用的针对移动设备的优化改进(Mobile-friendly <code>contrib.admin</code>)
支持 SQL 开窗表达式(Window expressions)

第一个特性，主要用于动态路由定义上。在 Django2.0 代码实现中，主要的变化是新增了 <code>django.urls.path</code> 函数，它允许使用一种更加简洁、可读的路由语法。比如之前的版本的码：
[python] view plain copy

url(r'^articles/(?P[0-9]{4})//span>, views.year_archive),

新语法支持类型转化，在上述的例子中， year_archive 函数接收到的 year 参数就变成整数而不字符串。在新版本中也可以写为：
如果你有接触过 Flask 框架，就会发现和 Variable-Rules 的语法形式和功能都是相类似的。
<h2 id="问题引入">问题引入</h2>
下面是 Django1.X 的一段代码：
[python] view plain copy

from django.conf.urls import url
def year_archive(request, year):

<pre><code class="highlight-chroma">   year = int(year) # convert str to int
</code></pre>

<pre><code class="highlight-chroma">   # Get articles from database
</code></pre>

def detail_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

def edit_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

def delete_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

urlpatterns = [

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles/(?P[0-9]{4})/', year_archive),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('article/(?P[a-zA-Z0-9]+)/detail/', detail_view),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles/(?P[a-zA-Z0-9]+)/edit/', edit_view),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles/(?P[a-zA-Z0-9]+)/delete/', delete_view),
</code></pre>

]

考虑下这样的两个问题：
第一个问题，函数 <code>year_archive</code> 中 year 参数是字符串类型的，因此需要先转为整数类型的变量值，当然 <code>year=int(year)</code> 不会有诸如如 TypeError 或者 ValueError 的异常。那么有没有一种方法，在 url 中，使得这一转化步骤可以由 Django 自动完成？
第二个问题，三个路由中 article_id 都是同样的正则表达式，但是你需要写三遍，当之后 article_id 规则改变后，需要同时修改三处代码，那么有没有一种方法，只需修改一处即可？
在 Django2.0 中，可以使用 <code>path</code> 解决以上的两个问题。
<h2 id="基本示例">基本示例</h2>
这是一个简单的例子：
[python] <a href="https://ld246.com/forward?goto=http%3A%2F%2

blog.csdn.net%2Fqq_40272386%2Farticle%2Fdetails%2F78800507%23" title="view plain" target="_blank" rel="nofollow ugc">view plain copy

from django.urls import path
from . import views
urlpatterns = [

<pre><code class="highlight-chroma">   path('articles/2003/', views.special_case_2003),
</code></pre>

<pre><code class="highlight-chroma">   path('articles//', views.year_archive),
</code></pre>

<pre><code class="highlight-chroma">   path('articles///', views.month_archive),
</code></pre>

<pre><code class="highlight-chroma">   path('articles////', views.article_detail),
</code></pre>

]

基本规则：

使用尖括号(<code><></code>)从 url 中捕获值。
捕获值中可以包含一个转化器类型（converter type），比如使用 `` 捕获一个整数变量。若果没转化器，将匹配任何字符串，当然也包括了 <code>/</code> 字符。
无需添加前导斜杠。

以下是根据 2.0 官方文档 而整理的示例分析表：
<table>
<thead>
<tr>
<th>请求 URL</th>
<th>匹配项</th>
<th>视图函数调用形式</th>
</tr>
</thead>
<tbody>
<tr>
<td>/articles/2005/03/</td>
<td>第 3 个</td>
<td>views.month_archive(request, year=2005, month=3)</td>
</tr>

<tr>
<td>/articles/2003/</td>
<td>第 1 个</td>
<td>views.special_case_2003(request)</td>
</tr>
<tr>
<td>/articles/2003</td>
<td>无</td>
<td>-</td>
</tr>
<tr>
<td>/articles/2003/03/building-a-django-site/</td>
<td>第 4 个</td>
<td>views.article_detail(request, year=2003, month=3, slug=”building-a-django-site”)</td>

</tr>
</tbody>
</table>
<h2 id="path转化器"><a href="https://ld246.com/forward?goto=https%3A%2F%2Fkinegratii.github.io%2F2017%2F09%2F25%2Fdjango2-url-path%2F%23path%25E8%25BD%25AC%25E%258C%2596%25E5%2599%25A8" title="path转化器" target="_blank" rel="nofollow ugc"></>path 转化器</h2>
<blockquote>
文档原文是 Path converters，暂且翻译为转化器。
</blockquote>
Django 默认支持以下 5 个转化器：

str,匹配除了路径分隔符（<code>/</code>）之外的非空字符串，这是默认的形式
int,匹配正整数，包含 0。
slug,匹配字母、数字以及横杠、下划线组成的字符串。
uuid,匹配格式化的 uuid，如 075194d3-6885-417e-a8a8-6c931e272f00。
path,匹配任何非空字符串，包含了路径分隔符

<h2 id="注册自定义转化器">注册自定义转化器</h2>
对于一些复杂或者复用的需要，可以定义自己的转化器。转化器是一个类或接口，它的要求有三：

<code>regex</code> 类属性，字符串类型

<code>to_python(self, value)</code> 方法，value 是由类属性 <code>regex</code> 所配到的字符串，返回具体的 Python 变量值，以供 Django 传递到对应的视图函数中。

<code>to_url(self, value)</code> 方法，和 <code>to_python</code> 相反，value 是一具体的 Python 变量值，返回其字符串，通常用于 url 反向引用。

例子：

[python] view plain copy

class FourDigitYearConverter:

<pre><code class="highlight-chroma"> regex = '[0-9]{4}'
</code></pre>

<pre><code class="highlight-chroma"> def to_python(self, value):
</code></pre>

<pre><code class="highlight-chroma">   return int(value)
</code></pre>

<pre><code class="highlight-chroma"> def to_url(self, value):
</code></pre>

<pre><code class="highlight-chroma">   return '%04d' % value
</code></pre>

使用 <code>register_converter</code> 将其注册到 URL 配置中：
[python] view plain copy

from django.urls import register_converter, path
from . import converters, views
register_converter(converters.FourDigitYearConverter, 'yyyy')
urlpatterns = [

<pre><code class="highlight-chroma"> path('articles/2003/', views.special_case_2003),
</code></pre>

<pre><code class="highlight-chroma"> path('articles//', views.year_archive),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> ...
</code></pre>

]

<h2 id="使用正则表达式">使用正表达式</h2>
如果上述的 paths 和 converters 还是无法满足需求，也可以使用正则表达式，这时应当使用 <code>django.urls.re_path</code> 函数。
在 Python 正则表达式中，命名式分组语法为 <code>(?Ppattern)</code> ，其中 name 为名， pattern 为待匹配的模式。
之前的示例代码也可以写为：
[python] view plain copy

from django.urls import path, re_path
from . import views
urlpatterns = [

<pre><code class="highlight-chroma"><span class="highlight
cl"> path('articles/2003/', views.special_case_2003),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> re_path('articles/(?P[0-9]{4})/', views.year_archive),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> re_path('articles/(?P[0-9]{4})/(?P[0-9]{2})/', views.month_archive),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> re_path('articles/(?P[0-9]{4})/(?P[0-9]{2})/(?P[^/]+)/', views.article_detail),
</code></pre>

]

这段代码和之前的代码实现了基本的功能，但是还是有一些区别：

这里的代码匹配更加严格，比如 year=10000 在这里就无法匹配。
传递给视图函数的变量都是字符串类型，这点和 <code>url</code> 是一致的。

无命名分组

一般来说，不建议使用这种方式，因为有可能引入歧义，甚至错误。
<h2 id="Import变动"><a href="https://ld246.com/forward?goto=https%3A%2F%2Fkinegratii
github.io%2F2017%2F09%2F25%2Fdjango2-url-path%2F%23Import%25E5%258F%2598%25E
%258A%25A8" title="Import变动" target="_blank" rel="nofollow ugc">Import 变动</h2

<code>django.urls.path</code> 可以看成是 <code>django.conf.urls.url</code> 的增强
式。
为了方便，其引用路径也有所变化。
<table>
<thead>
<tr>
<th>1.X</th>
<th>2.0</th>
<th>备注</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>django.urls.path</td>
<td>新增，url 的增强版</td>
</tr>
<tr>
<td>django.conf.urls.include</td>
<td>django.urls.include</td>
<td>路径变更</td>
</tr>
<tr>
<td>django.conf.urls.url</td>
<td>django.urls.re_path</td>
<td>异名同功能，url 不会立即废弃</td>
</tr>
</tbody>
</table>
<h2 id="总结"><a href="https://ld246.com/forward?goto=https%3A%2F%2Fkinegratii.github.
o%2F2017%2F09%2F25%2Fdjango2-url-path%2F%23%25E6%2580%25BB%25E7%25BB%259
" title="总结" target="_blank" rel="nofollow ugc">总结</h2>
新的 path 语法可以解决一下以下几个场景：

类型自动转化
公用正则表达式

将问题引入一节的代码使用新的 path 函数可以改写如下：
[python] <a href="https://ld246.com/forward?goto=http%3A%2F%2
blog.csdn.net%2Fqq_40272386%2Farticle%2Fdetails%2F78800507%23" title="view plain" tar
et="_blank" rel="nofollow ugc">view plain <a href="https://ld246.com/forward?goto=h
tp%3A%2F%2Fblog.csdn.net%2Fqq_40272386%2Farticle%2Fdetails%2F78800507%23" title="
opy" target="_blank" rel="nofollow ugc">copy

from django.urls import path, register_converter
class ArticleIdConverter:

<pre><code class="highlight-chroma"><span class="highlight
cl"> regex = '[a-zA-Z0-9]+'

</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> def to_python(self, value):
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl">   return value
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> def to_url(self, value):
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl">   return value
</code></pre>

register_converter(ArticleIdConverter, 'article_id')
def year_archive(request, year):

<pre><code class="highlight-chroma"><span class="highlight
cl"> year = int(year) # convert str to int
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> # Get articles from database
</code></pre>

def detail_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

def edit_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

def delete_view(request, article_id):

<pre><code class="highlight-chroma"><span class="highlight
cl"> pass
</code></pre>

urlpatterns = [

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles/(?P[0-9]{4})/', year_archive),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('article//detail/', detail_view),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles//edit/', edit_view),
</code></pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"> url('articles//delete/', delete_view),
</code></pre>

]

从流程来看，包含了四个步骤：匹配 => 捕获 => 转化 => 视图调用，和之前相比多转化这一步。