

장고 차근차근 시작하기

Second Edition

당신의 파이썬/장고 페이스메이커가 되겠습니다. ;)

EP28. URL Reverse를 통해 유연하게 URL 생성하기

URL Dispatcher

urls.py 변경만으로 "각 뷰에 대한 URL"이 변경되는 유연한 URL 시스템

```
# "/blog/", "/blog/1/" 주소로 서비스하다가
```

```
urlpatterns = [  
    path('blog/', blog_views.post_list, name='post_list'),  
    path('blog/<int:pk>', blog_views.post_detail, name='post_detail'),  
]
```

```
# 다음과 같이 변경을 하면,
```

```
# 이제 "/weblog/", "/weblog/1/" 주소로 서비스하게 됩니다.
```

```
urlpatterns = [  
    path('weblog/', blog_views.post_list, name='post_list'),  
    path('weblog/<int:pk>', blog_views.post_detail, name='post_detail'),  
]
```

URL Reverse의 혜택

- 개발자가 일일이 URL을 계산하지 않아도 됩니다. 만세 ~~~
- URL이 변경되더라도, URL Reverse가 변경된 URL을 추적
 - 누락될 일이 없어요.

직접 URL을 계산한다면 ~

1. blog앱 Post목록을 볼려면, post_list 뷰를 호출해야하니깐,
2. urls.py 를 뒤적뒤적거리며, URL계산계산
3. 계산완료 ! → /blog/ 주소를 쓰면 되겠네.

```
<!-- blog/templates/blog/layout.html 내에서의 링크 -->  
<a href="/blog/">블로그 글 목록</a>
```

```
<!-- blog/templates/blog/post_form.html 내에서의 링크 -->  
<a href="/blog/">블로그 글 목록</a>
```

```
<!-- blog/templates/blog/comment_form.html 내에서의 링크 -->  
<a href="/blog/">블로그 글 목록</a>
```

그런데, 이 blog앱을 다른 프로젝트에도 쓰려고 옮겼는 데, URL Prefix를 weblog로 쓰고 싶어요.
그냥 안된다고 할 것인가? 아악 ~~~ ☹

URL 계산은 장고에게 양보하세요.

1. blog앱 Post목록을 볼려면, post_list 뷰를 호출해야하니깐,
2. ~~urls.py 를 뒤적뒤적거리며, URL계산계산~~
3. ~~계산완료! → /blog/ 주소를 쓰면 되겠네.~~

```
<!-- blog/templates/blog/layout.html 내에서의 링크 -->  
<a href="{% url 'blog:post_list' %}">블로그 글 목록</a>
```

```
<!-- blog/templates/blog/post_form.html 내에서의 링크 -->  
<a href="{% url 'blog:post_list' %}">블로그 글 목록</a>
```

```
<!-- blog/templates/blog/comment_form.html 내에서의 링크 -->  
<a href="{% url 'blog:post_list' %}">블로그 글 목록</a>
```

그런데, 이 blog앱을 다른 프로젝트에도 쓰려고 옮겼는 데, URL Prefix를 weblog로 쓰고 싶어요.
그냥 안된다고 할 것인가? 아악 ~~~ ☹️ ➔ 코드 변경 거의 없이 가능합니다.

URL Reverse를 수행하는 4가지 함수 (1)

- url 템플릿태그
 - 내부적으로 reverse 함수를 사용
- reverse 함수
 - 매칭 URL이 없으면 NoReverseMatch 예외 발생
- resolve_url 함수
 - 매핑 URL이 없으면 "인자 문자열"을 그대로 리턴
 - 내부적으로 reverse 함수를 사용
- redirect 함수
 - 매칭 URL이 없으면 "인자 문자열"을 그대로 URL로 사용
 - 내부적으로 resolve_url 함수를 사용

URL Reverse를 수행하는 4가지 함수 (2)

```
{% url "blog:post_detail" 100 %}  
{% url "blog:post_detail" pk=100 %}
```

→ 문자열 URL

```
reverse('blog:post_detail', args=[100])  
reverse('blog:post_detail', kwargs={'pk': 100})
```

→ 문자열 URL

```
resolve_url('blog:post_detail', 100)  
resolve_url('blog:post_detail', pk=100)  
resolve_url('/blog/100/')
```

→ 문자열 URL

```
redirect('blog:post_detail', 100)  
redirect('blog:post_detail', pk=100)  
redirect('/blog/100/')
```

→ HttpResponse 응답 (301 or 302)

모델 객체에 대한 detail 주소 계산

- 매번 다음과 같은 코드로 하실 수도 있겠지만,

```
resolve_url('blog:post_detail', pk=post.pk)
```

```
redirect('blog:post_detail', pk=post.pk)
```

```
{% url 'blog:post_detail' post.pk %}
```

- 다음과 같이 사용하실 수도 있습니다. 어떻게?

```
resolve_url(post)
```

```
redirect(post)
```

```
{{ post.get_absolute_url }}
```


모델 클래스에 get_absolute_url() 구현

- resolve_url 함수는 가장 먼저 get_absolute_url() 함수의 존재 여부를 체크하고, 존재할 경우 reverse를 수행하지 않고 그 리턴값을 즉시 리턴 ([관련코드](#))

```
# django/shortcuts.py
```

```
def resolve_url(to, *args, **kwargs):  
    if hasattr(to, 'get_absolute_url'):  
        return to.get_absolute_url()  
    # 중략  
    try:  
        return reverse(to, args=args, kwargs=kwargs)  
    except NoReverseMatch:  
        # 나머지 코드 생략
```

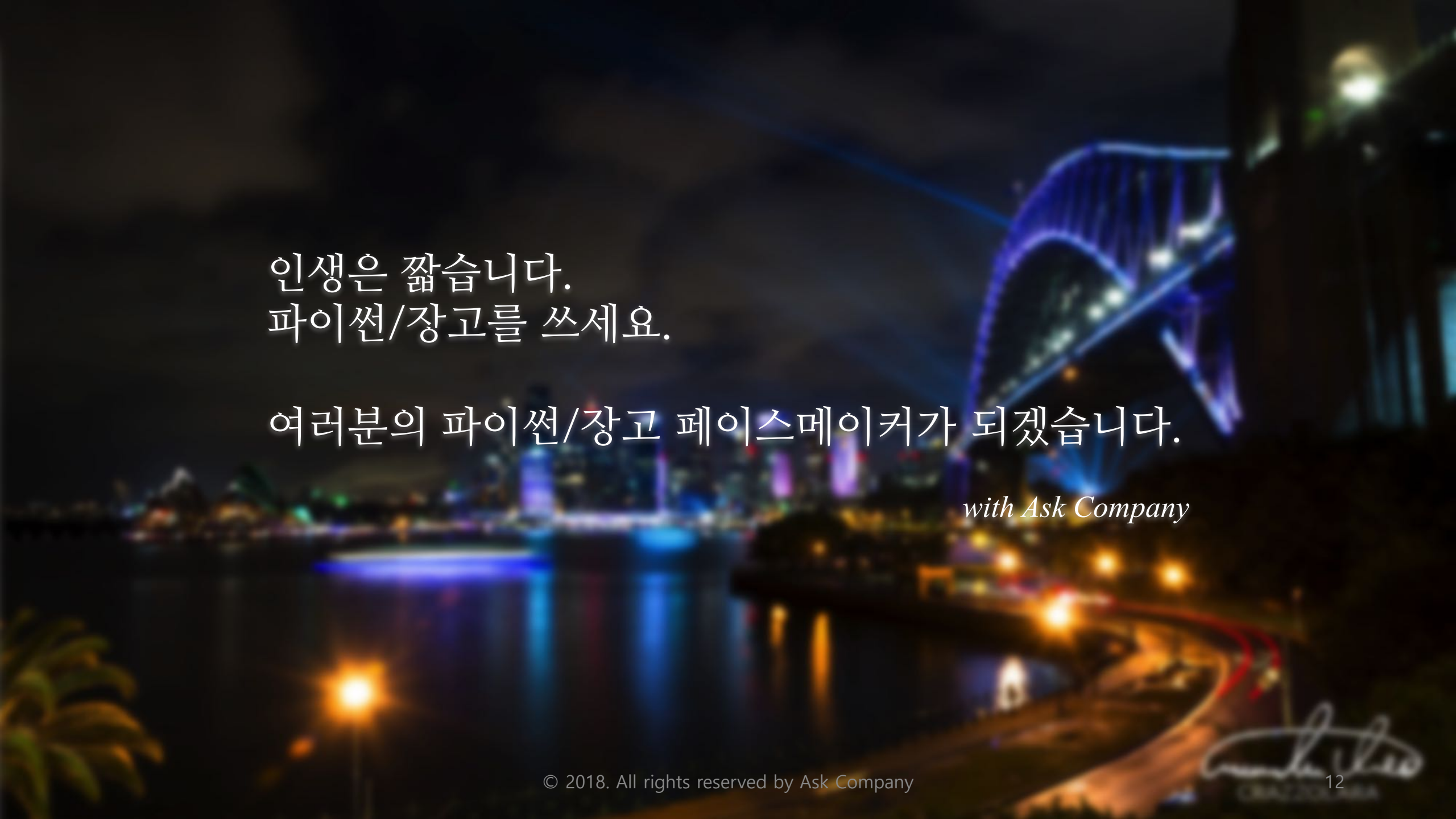
resolve_url/redirect를 위한 모델 클래스 추가 구현

```
from django.urls import reverse

class Post(models.Model):
    # 중략
    def get_absolute_url(self):
        return reverse('blog:post_detail', args=[self.pk])
```

그 외 활용

- CreateView / UpdateView
 - success_url을 제공하지 않을 경우, 해당 model instance 의 get_absolute_url 주소로 이동이 가능한지 체크하고, 이동이 가능할 경우 이동
 - 생성/수정하고나서 Detail화면으로 이동하는 것은 자연스러운 시나리오
- 특정 모델에 대한 Detail뷰를 작성할 경우
 - Detail뷰에 대한 URLConf설정을 하자마자, 필히 get_absolute_url설정을 해주세요. 코드가 보다 간결해집니다.

A nighttime photograph of a cityscape featuring a bridge with blue and white lights. The lights are reflected in the water below. The sky is dark, and the city lights create a vibrant scene.

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

with Ask Company