**get\_object()**

get\_object is actually a method of django.views.generic.detail.SingleObjectMixin

**What is mixin?** In object-oriented programming languages, a **mixin** is a class that contains methods for use by other classes without having to be the parent class of those other classes.

get\_object() is the method that gets the object for which the DetailView is meant for.

Essentially what the method does is it search the \*\*kwargs passed into the method for a 'slug' or a 'pk'. If it is able to find either, it basically does a Model.objects.get(slug=slug) (or pk=pk), if neither is found it throws an error.

**Documentation:**

**get\_object(*queryset=None*)**

Returns the single object that this view will display. If **queryset** is provided, that queryset will be used as the source of objects; otherwise, [**get\_queryset()**](https://docs.djangoproject.com/en/2.2/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin.get_queryset) will be used. **get\_object()** looks for a [**pk\_url\_kwarg**](https://docs.djangoproject.com/en/2.2/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin.pk_url_kwarg) argument in the arguments to the view; if this argument is found, this method performs a primary-key based lookup using that value. If this argument is not found, it looks for a [**slug\_url\_kwarg**](https://docs.djangoproject.com/en/2.2/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin.slug_url_kwarg) argument, and performs a slug lookup using the [**slug\_field**](https://docs.djangoproject.com/en/2.2/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin.slug_field).

When [**query\_pk\_and\_slug**](https://docs.djangoproject.com/en/2.2/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin.query_pk_and_slug) is **True**, **get\_object()** will perform its lookup using both the primary key and the slug.

**urls.py**

path('blog/<slug:title\_slug>/', PostDetailView.as\_view(), name='blog\_detail'),

path('blog/<slug:title\_slug>/update/',

PostUpdateView.as\_view(), name='blog\_update'),

path('blog/<slug:title\_slug>/delete/',

PostDeleteView.as\_view(), name='blog\_delete'),

**views.py**

class PostDetailView(DetailView):

model = BlogPost

template\_name = 'blogs/blog-detail.html'

def get\_object(self):

slug = self.kwargs.get('title\_slug')

return get\_object\_or\_404(BlogPost, slug=slug)

class PostUpdateView(UpdateView):

model = BlogPost

template\_name = 'blogs/blog-update.html'

fields = ['title',]

def get\_object(self):

slug = self.kwargs.get('title\_slug')

return get\_object\_or\_404(BlogPost, slug=slug)

class PostDeleteView(DeleteView):

model = BlogPost

template\_name = 'blogs/blog-delete.html'

success\_url = "/"

def get\_object(self):

slug = self.kwargs.get('title\_slug')

return get\_object\_or\_404(BlogPost, slug=slug)

**Note:** if you have `pk` no need for `get\_object()` in GCBV.

**urls.py**

path('post/<int:pk>/', PostDetailView.as\_view(), name='post-detail'),

path('post/<int:pk>/update/', PostUpdateView.as\_view(),

name='post-update'),

path('post/<int:pk>/delete/', PostDeleteView.as\_view(),

name='post-delete'),

**views.py**

class PostDetailView(DetailView):

model = Post

class PostUpdateView(UpdateView):

model = Post

fields = ['title', 'content']

class PostDeleteView(DeleteView):

model = Post

success\_url = '/'