Agiliq

When and how to use Django DetailView

When to use DetailView?

Django provides several class based generic views to accomplish common tasks. One among them is DetailView.

DetailView **should be used** when you want to present detail of a single model instance.

DetailView **shouldn't be used** when your page has forms and does creation or update of objects. FormView, CreateView and UpdateView are more suitable for working with forms, creation or updation of objects.

Vanilla view can achieve everything which DetailView can, but DetailView has an advantage of avoiding a lot of boilerplate code which would be needed with View.

Let's write a view by subclassing **View** and then modify the view to subclass **DetailView**. DetailView would help us avoid several lines of code and would also provide better separation of concern.

Vanilla View

Assume there is a model called Book in app `books` which looks like:

```
# books/models.py
class Book(models.Model):
    title = models.CharField(max_length=100)
    isbn = models.CharField(max_length=100)
```

Agiliq

You want to have a page which shows detail of a particular book. The url looks like:

```
# books/urls.py
from django.urls import path

from . import views

app_name = 'books'
urlpatterns = [
    path('<int:pk>/', views.BookDetailView.as_view(), name='detail'),
]
```

Vanilla view looks like:

```
# books/views.py
class BookDetailView(View):
    def get(self, request, *args, **kwargs):
        book = get_object_or_404(Book, pk=kwargs['pk'])
        context = {'book': book}
        return render(request, 'books/book_detail.html', context)
```

books/book_detail.html looks like the following:

```
<h3>Book detail</h3>
<p>{{book.title}}</p>
<p></p>
```

You should be seeing book detail at `http://localhost:8000/books/1/` .

By subclassing DetailView

Modify books/views.py code to look like:

Agiliq

Reload the page and you would still see book detail.

DetailView helped us avoid the following boilerplate code

— Avoid providing `get()` implementation

— Avoid creation of context

— Avoid passing context to the template

— Avoid returning HttpResponse() objects created by render().

Filter queryset before showing detail page

You might only detail pages for `published` Books to be accessible and unpublished books should give 404. This scenario assumes that there is a BooleanField called `is_published` on Book.

```
class Book(models.Model):
    title = models.CharField(max_length=100)
    isbn = models.CharField(max_length=100)
    is_published = models.Booleanfield(default=True)

    def __unicode__(self):
        return self.title
```

Create a book with is_published=False.

```
In [1]: from books.models import Book

In [2]: Book.objects.create(title='My Month Book', isbn='978-92-95055-02-6', i
Out[2]: <Book: Book object (2)>
```

You can modify BookDetailView as such:

```
class BookDetailView(DetailView):
      queryset = Book.objects.filter(is_published=True)
```

Notice that we removed the `model` attribute on the view and instead provided a `queryset` attribute.

Restrict users to only see books created by them

Let’s think of a hypothetical strange requirement. A user should only be allowed to view a book which has been written by them. This assumes that there is a `user` Foreign Key on Book.

```
In [3]: u = User.objects.latest('pk')

In [5]: Book.objects.create(title='My Month Book', isbn='978-92-95055-02-6', i
Out[5]: <Book: Book object (3)>
```

Modify BookDetailView to the following:

```
class BookDetailView(DetailView):

        def get_queryset(self):
                if self.request.user.is_authenticated:
                        return Book.objects.filter(is_published=True, user=sel
                else:
                        return Book.objects.none()
```

A user would only be able to see a detail view if the book is published and if the user is the writer of this book.

Notice that we removed the `queryset` attribute on the view and instead provided a `get_queryset()` implementation.

Use slug as url parameter instead of pk

You might want to use isbn as the url parameter instead of pk. Eg: You might want detail view of book 1 to show up at `/books/<isbn>/` .

Let’s modify BookDetailView to look like:

```
    slug_field = 'isbn'
    slug_url_kwarg = 'isbn'

        def get_queryset(self):
                if self.request.user.is_authenticated:
                        return Book.objects.filter(is_published=True, user=sel
                else:
                        return Book.objects.none()
```

Assuming `isbn` of a Book is `978-92-95055-02-6` and the logged in user is the creator of Book, then the detail page would be accessible at `http://localhost:8000/books/978-92-95055-02-6/` .

Our other posts on generic class views

— TemplateView

— ListView

— FormView

— CreateView

Thank you for reading the Agiliq blog. This article was written by Akshar on Jan 1, 2019 in django .

You can subscribe ⚛ to our blog.

We love building amazing apps for web and mobile for our clients. If you are looking for development help, contact us today ✉.

DJANGO ADMIN COOKBOOK
JOURNEYMAN PYTHON
BUILDING MULTI TENANT APPLICATIONS WITH DJANGO
DJANGO ORM COOKBOOK
BUILDING APIS WITH DJANGO AND DJANGO REST FRAMEWORK
DJANGO PROJECTS COOKBOOK

Agiliq

Upvote 2 | Funny 0 | Love 0 | Surprised 0 | Angry 0 | Sad 0

Comments and reactions for this thread are now closed. ✕

0 Comments The Agiliq Blog 1 Login

Recommend 4 Tweet Share Sort by Oldest

This discussion has been closed.

ALSO ON THE AGILIQ BLOG

Tastypie with ForeignKey - Agiliq Blog | Django web app development

5 years ago

ihimobileapps — This is great coding App Developers Long Island, NY and NYC . http://www.ihimobileapps.com/

AngularJS injectors internals - Agiliq Blog | Django web app development

2 years ago

Palani Suresh — Thanks for the valuable information to be share with us.This is very helpful to me,understanding the concepts of

Building Chrome Extensions - Agiliq Blog | Django web app development

5 years ago

Avatar Techy Ashutosh — I like the way of coding, but i am confused from where i put this code? Can you help me Karambir?

Getting started with Django tastypie - Agiliq Blog | Django web app …

5 years ago

Avatar Larry Wachira — Great tutorial. Very clear explanation.

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

Agiliq

About Us

Agiliq

Email us : **hello@agiliq.com**

Phone us: **+919949997612**