Let's Explore Some Built-in

Decorators In Django Framework

Django provides several built-in view decorators that you can use to add functionality or modify the behavior of your views.

1. login_required

If you added this decorator to your view function, it means only authenticate users can access that view. Unauthenticated user will redirect to login page, if you have added the login page path to your LOGIN_REDIRECT_URL in settings.

```
from django.contrib.auth.decorators import login_required
from django.shortcut import render

@login_required
def user_profile(request):
    return render(request, "profile.html")
```

This profile page is only access by the logged in users

2. require_http_methods

It allows you to specify a list of allowed HTTP methods for the view.

```python
from django.views.decorators.http import require_http_methods
from django.shortcut import render

@require_http_methods(["GET", "POST"])
def user_profile(request):
    return render(request, "profile.html")
```

Here you can pass the list of allowed http method. It means you can only send GET & POST request for this view function.

3. cache_page

This decorator cache the entire view output for a specified amount of time. This can improve performance by serving cached content for repeated requests.

```python
from django.views.decorators.cache import cache_page
from django.shortcut import render

@cache_page(60 * 15)  # Cache for 15 minutes
def user_profile(request):
    return render(request, "profile.html")
```

Here you can pass the cache_page decorator and specify the time. and this entire view going to be cache for 15 minutes.

4. gzip_page

gzip_page decorator is used to compress the content of a view using the gzip compression algorithm before sending it to the client, if your browser allows.

```
from django.views.decorators.gzip import gzip_page
from django.shortcut import render

@gzip_page
def profile(request):
    return render(request,
                  'profile.html',
                    {'data': some_data}
                 )
```

You can pass the gzip_page decorator, that will compress the view content.

5. permission_required

Restricts access to the view based on the user's permissions. If the user does not have the required permissions, they are redirected to a login page.

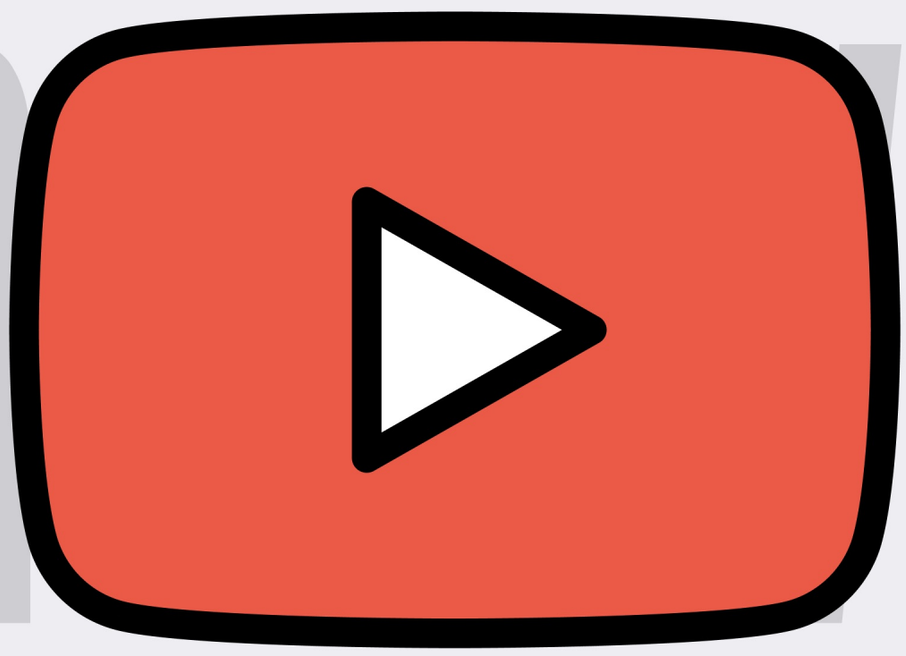
```python
from django.contrib.auth.decorators import
permission_required

@permission_required('myapp.can_view_content')
def dashboard(request):
    return render(request, "dashboard.html")
```

You can use the permission decorator to add the permission follow by the <app_label>.<permission_code> on the per view. If user hasn't permission to access the view, it redirect to login page.

These are just a few examples of the built-in view decorators in Django. You can combine decorators as needed for your views, and Django also allows you to create custom decorators for more specific functionality.

For more django related content

Subscribe To My Youtube Channel



Link in bio

DID YOU FIND THIS HELPFUL

Follow for more