Django signals run in the same thread as the caller. This means that when a signal is sent, the receiver functions are executed in the same thread that triggered the signal.

```
# models.py
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver
import threading

class MyModel(models.Model):
    name = models.CharField(max_length=100)

@receiver(post_save, sender=MyModel)
def my_model_saved(sender, instance, created, **kwargs):
    # Print the current thread name
    print(f'MyModel instance saved: {instance.name}, created: {created}')
    print(f'Current thread: {threading.current_thread().name}')

# In Django shell or a view
from myapp.models import MyModel

# This will trigger the post_save signal
new_instance = MyModel(name='Test Instance')
new_instance.save()
```

In this code, we have a model **MyModel** and a signal receiver **my_model_saved**. Inside the receiver, we print the name of the current thread using **threading.current_thread().name**.