Questions for Django Trainee at Accuknox

Abhay Jagtap
abhay.jagtap000@gmail.com

Topic: **Django Signals**

Question 1: By default are django signals executed synchronously or asynchronously? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

Answer: Django signals are executed synchronously by default. It will run in the same thread. As a proof, I have written following code:

Q1.py | signals.py × | apps.py | settings.py

myproject > myapp > signals.py > ...

```python
import time
import logging
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver

# Set up logging
logger = logging.getLogger(__name__)

# Signal Receiver
@receiver(post_save, sender=User)
def user_saved_signal(sender, instance, **kwargs):
    logger.info("Signal execution started...")
    time.sleep(5)  # Simulating a delay
    logger.info("Signal execution finished...")

```

```
(assignmentEnv) abhay@Abhays-MacBook-Air myproject % python manage.py shell

Python 3.9.6 (default, Nov 11 2024, 03:15:38)
[Clang 16.0.0 (clang-1600.0.26.6)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import User
>>> import time
>>>
>>> # Measure execution time
>>> start_time = time.time()
>>> User.objects.create(username="testuser")
end_time = time.time()

print(f"User created in {end_time - start_time:.2f} seconds")
<User: testuser>
>>> end_time = time.time()
>>>
>>> print(f"User created in {end_time - start_time:.2f} seconds")
User created in 5.01 seconds
>>> zsh: quit       python manage.py shell
(assignmentEnv) abhay@Abhays-MacBook-Air myproject %
```

Here we can see User.objects.create(username="testuser") took 5.01 seconds. Which means the execution was stopped for 5 seconds which proves that django signals are executed synchronously by default. If it is asynchronously then it would not have waited till 5 seconds.

Whole code is added in Github Repository.

Question 2: Do django signals run in the same thread as the caller? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

Ans: Yes, django signals run in the same thread as the caller. To prove this, I will call thread IDs in both the processes.

```python
# Set up logging
logger = logging.getLogger(__name__)

# Signal Receiver
@receiver(post_save, sender=User)
def user_saved_signal(sender, instance, **kwargs):
    # Q1
    # logger.info("Signal execution started...")
    # time.sleep(5)  # Simulating a delay
    # logger.info("Signal execution finished...")

    #Q2
    logger.info(f"Signal Handler: Running in thread {threading.get_ident()}")
```

Q1.py | signals.py M | tests.py M | apps.py | settings.py

myproject > myapp > tests.py > ...

```python
from django.contrib.auth.models import User

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Function to create a user and trigger the signal
def create_user():
    logger.info(f"Main Function: Running in thread {threading.get_ident()}")
    User.objects.create(username="testuser")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
(assignmentEnv) abhay@Abhays-MacBook-Air myproject % python manage.py shell

Python 3.9.6 (default, Nov 11 2024, 03:15:38)
[Clang 16.0.0 (clang-1600.0.26.6)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from myapp.tests import create_user
>>> create_user()
INFO:myapp.tests:Main Function: Running in thread 8328677440
INFO:myapp.signals:Signal Handler: Running in thread 8328677440
>>> 
```

As we can see the thread ID is the same. So it proves that django signals run in the same thread as the caller.

Question 3: By default do django signals run in the same database transaction as the caller? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

Ans: Django signals, by default, do not run in the same database transaction as the caller. This is because Django signals like **post_save** are fired after the transaction has been committed. The signal handler's exceptions do not affect the transaction because the database operation has already been committed before the signal is triggered.

Q1.py | signals.py M | tests.py | apps.py | settings.py

myproject > myapp > signals.py > user_saved_signal

```python
def user_saved_signal(sender, instance, **kwargs):
    # logger.info("Signal execution started...")
    # time.sleep(5)  # Simulating a delay
    # logger.info("Signal execution finished...")

    #Q2
    # logger.info(f"Signal Handler: Running in thread {threading.get_ident()}")

    #Q3
    logger.info("Signal Handler: Start")
    # Raise an exception to test rollback
    raise Exception("Simulating an error in signal handler")
    logger.info("Signal Handler: End")  # This line will not execute
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
(assignmentEnv) abhay@Abhays-MacBook-Air myproject % python manage.py shell

Python 3.9.6 (default, Nov 11 2024, 03:15:38)
[Clang 16.0.0 (clang-1600.0.26.6)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.db import IntegrityError
>>> from django.contrib.auth.models import User
>>>
>>> # Try to create a user, but the signal will raise an exception
>>> try:
...     User.objects.create(username="testuser")
... except Exception as e:
...     print(f"Caught an error: {e}")
...
# Check if the user was created
user_exists = User.objects.filter(username="testuser").exists()
Caught an error: Simulating an error in signal handler
>>> # Check if the user was created
>>> user_exists = User.objects.filter(username="testuser").exists()
>>> print(f"User exists: {user_exists}")
User exists: True
>>> 
```

The signal handler raised an exception after the user was saved, but the user was already created because the **post_save** signal is triggered after the transaction is committed.
If Django signals ran within the same transaction, the exception would have caused the entire transaction to roll back, but that’s not the case here.

Topic: Custom Classes in Python

Description: You are tasked with creating a Rectangle class with the following requirements:

1. An instance of the `Rectangle` class requires `length:int` and `width:int` to be initialized.
2. We can iterate over an instance of the `Rectangle` class
3. When an instance of the `Rectangle` class is iterated over, we first get its length in the format: **`{'length': <VALUE_OF_LENGTH>}`** followed by the width **`{width: <VALUE_OF_WIDTH>}`**

Rectangle.py U × signals.py tests.py apps.py settings.py

Rectangle.py > ...

```python
class Rectangle:
    def __init__(self, length: int, width: int):
        # Initialize the length and width attributes
        self.length = length
        self.width = width

    def __iter__(self):
        # Define the iteration behavior
        yield {'length': self.length}
        yield {'width': self.width}

# Example:
rect = Rectangle(5, 10)

# Iterating over the instance
for item in rect:
    print(item)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
(assignmentEnv) Abhays-MacBook-Air:Python Assignment abhay$ python3 Rectangle.py
{'length': 5}
{'width': 10}
(assignmentEnv) Abhays-MacBook-Air:Python Assignment abhay$
```