HERE'S A SUBSCRIPTION YOU CAN'T REFUSE

GraphQL Wroclaw #6 Online - July 6, 2021

VALENTINOGAGLIARDI

WHOAMI

Js developer who

loves **Django**, **tests**,

and **tech writing**.

Building stuff **@20tab**

THE PLAN

- Adding **GraphQL subscriptions** to a **Django** project
- Working **asynchronously** with the Django ORM

CONFIGURATION

```
# django_project/settings.py

INSTALLED_APPS = [
    ...
    “orders.apps.OrdersConfig", # sample app
    "ariadne.contrib.django",
    "channels",
]

…


ASGI_APPLICATION = "django_project.asgi.application"
```

ROUTING

```python
# django_project/asgi.py

import os

from django.core.asgi import get_asgi_application

from ariadne.asgi import GraphQL
from channels.routing import URLRouter
from django.urls import path

from orders.schema import schema

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "django_project.settings")


application = URLRouter(
    [
        path("graphql/", GraphQL(schema)),
        path("", get_asgi_application())
    ]
)
```

SCHEMA

```
scalar Date

enum OrderState {
    PAID
    UNPAID
}

type User {
    username: String
    email: String
}

type Order {
    date: Date
    state: OrderState
    user: User
}
```

```
type Query {
    getOrders: [Order]
}

type Subscription {
    getOrder: Order
}
```

SUBSCRIPTION IN ARIADNE

```python
# orders/schema.py
# Your imports here …

type_defs = gql("""Your schema here (or in a separate file)""")

subscription = SubscriptionType()


@subscription.source("getOrder")
async def generate_order(obj, info):
    """Disclaimer: for demo purpose only."""
    while True:
        await asyncio.sleep(1)
        """Wrong!"""
        yield Order.objects.last()


@subscription.field("getOrder")
def resolve_order(order, info):
    return order


schema = make_executable_schema(type_defs, [date_scalar], subscription)
```

ORM ACCESS

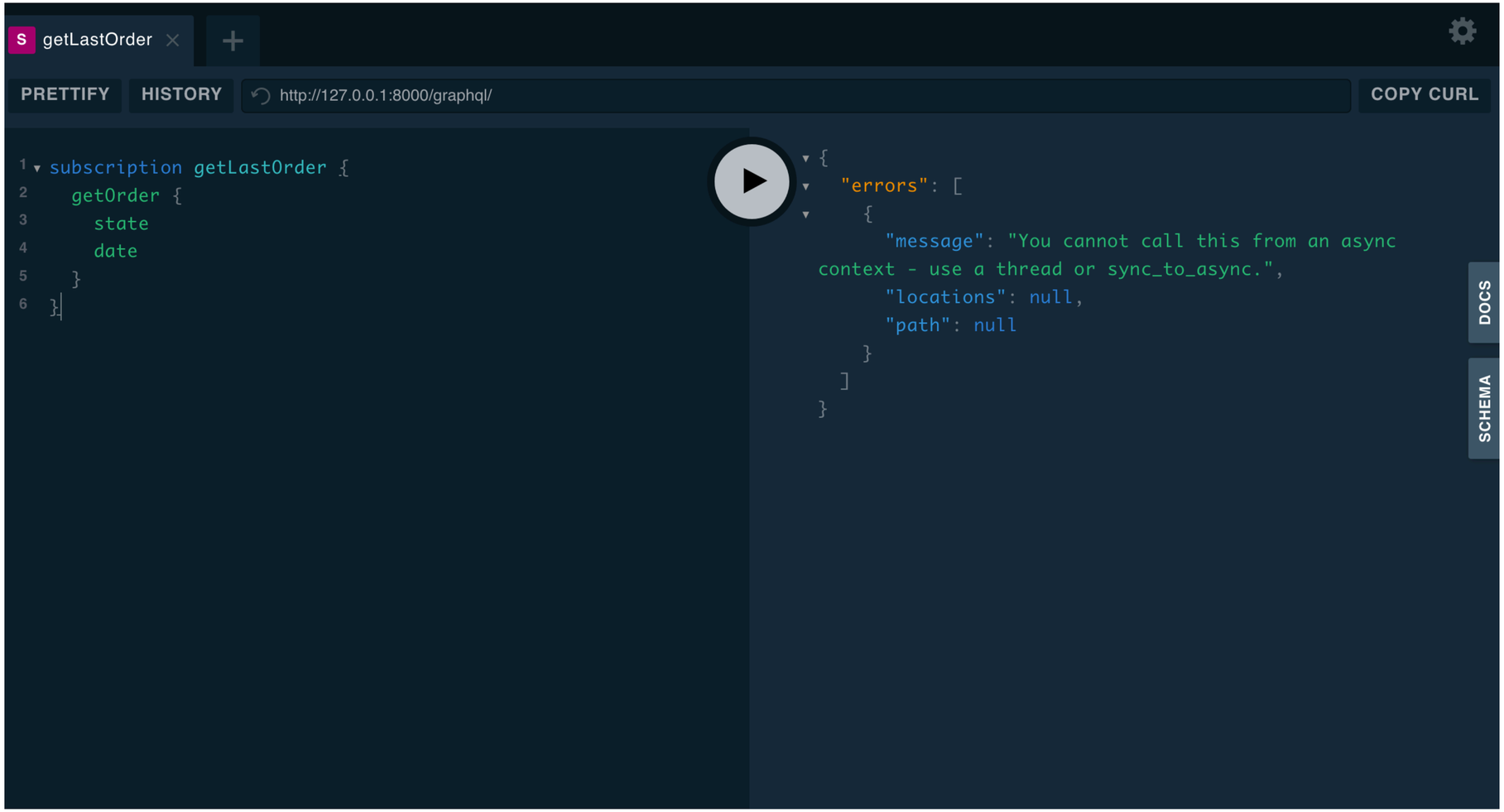
```
...

@subscription.source("getOrder")
async def generate_order(obj, info):
     """Disclaimer: for demo purpose only."""
    while True:
        await asyncio.sleep(1)
        """Wrong!"""
        yield Order.objects.last()

...
```

```
{
    "errors": [
        {
            "message": "You cannot call this from an async context - use
a thread or sync_to_async.",
        }
    ]
}
```

PLAYGROUND


getLastOrder
PRETTIFY
HISTORY
http://127.0.0.1:8000/graphql/
COPY CURL
subscription getLastOrder {
getOrder {
state
date
}
}
{
"errors": [
{
"message": "You cannot call this from an async context - use a thread or sync_to_async.",
"locations": null,
"path": null
}
]
}
DOCS
SCHEMA

ASYNC ORM ACCESS

```
# orders/schema.py
# Your imports here ...

...

from channels.db import database_sync_to_async
...


@subscription.source("getOrder")
async def generate_order(obj, info):
    """Disclaimer: for demo purpose only."""
    while True:
        await asyncio.sleep(1)
        yield await database_sync_to_async(lambda: Order.objects.last())()

...
```

SUBSCRIPTION

```
subscription getLastOrder {
  getOrder {
    state
    date
  }
}
```

FOREIGN KEY

```
subscription getLastOrder {
  getOrder {
    state
    date
    user {
       username
       email
    }
  }
}
```

```
{
    "errors": [
        {
            "message": "You cannot
call this from an async context - use
a thread or sync_to_async.",
        }
    ]
}
```

ASYNC ORM ACCESS (FK)

```
# orders/schema.py
# Your imports here ...

...

@subscription.source("getOrder")
async def generate_order(obj, info):
    """Disclaimer: for demo purpose only."""
    while True:
        await asyncio.sleep(1)
        yield await database_sync_to_async(
            lambda: Order.objects.select_related("user").last()
        )()

…
```

SUBSCRIPTION

```
subscription getLastOrder {
  getOrder {
    state
    date
    user {
       username
       email
    }
  }
}
```

```
{
  "data": {
    "getOrder": {
      "state": "PAID",
      "date": "2026-04-12",
      "user": {
        "username": "caty88",
        "email": "caty@dev.io"
      }
    }
  }
}
```

CORS

```
# django_project/asgi.py

...
from starlette.middleware.cors import CORSMiddleware
...

application = URLRouter(
    [
        path(
            "graphql/",
            CORSMiddleware(
                GraphQL(schema),
                """Origin * for dev only!"""
                allow_origins=["*"],
                allow_methods=["*"]
            ),
        ),
        path("", get_asgi_application()),
    ]
)
```

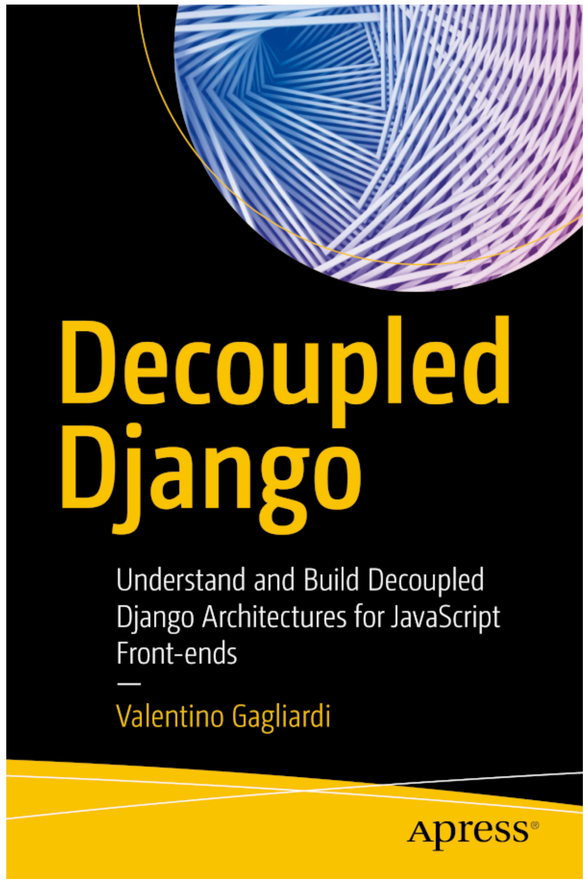
FRONTEND

- `HttpLink` and `WebSocketLink` (Apollo client)
- `reconnect: true`
- `useSubscription()`

https://www.apollographql.com/docs/react/data/subscriptions/

THANK YOU!

https://valentinog.com/blog/channels-ariadne/



VALENTINOGAGLIARDI