DataLoader

> DataLoader allows you to decouple unrelated parts of your application without sacrificing the performance of batch data-loading. While the loader presents an API that loads individual values, all concurrent requests will be coalesced and presented to your batch loading function. This allows your application to safely distribute data fetching requirements throughout your application and maintain minimal outgoing data requests

```
import strawberry
import asyncio
from strawberry.dataloader import DataLoader

async def fetch(product_id):
    url = f"https://[URL]/product/v1/products/{product_id}"
    resp = requests.get(url)
    data = json.loads(resp.text)
    product = Product.from_json(data)
    return product


async def load_product(product_id_list: List[int]) -> Product:
    futures = [
        asyncio.ensure_future(fetch(url)) for url in product_id_list
  ] # 태스크(퓨처) 객체를 리스트로 만듦
    result = await asyncio.gather(*futures)

    return result


loader = DataLoader(load_fn=load_product)


async def product_resolver(root, info) -> Product:
    product_id = root.id
    return await loader.load(product_id)


@strawberry.type
class ProductInfo(FromJson):
    id: int
    name: str
    product: Product = strawberry.field(resolver=product_resolver)
```

관련 문서

Plugin Backlinks: 아무 것도 없습니다.