Dataclass

1. Bản chất của `@dataclass` là gì?

`@dataclass` là một decorator dùng để **tự động tạo các phương thức đặc biệt** cho một class:

- `__init__`: hàm khởi tạo
- `__repr__`: biểu diễn đối tượng dưới dạng chuỗi
- `__eq__`: so sánh hai đối tượng
- `__lt__`, `__le__`, `__gt__`, `__ge__` (nếu `order=True`)
- `__hash__` (nếu `frozen=True`)

□□ Điều này giúp bạn tiết kiệm rất nhiều công sức khi làm việc với các lớp dữ liệu thuần túy (plain data objects).

2. So sánh class bình thường vs dataclass

Ví dụ: Class bình thường

```
class Product:
    def __init__(self, name, price, in_stock=True):
        self.name = name
        self.price = price
        self.in_stock = in_stock

    def __repr__(self):
        return f"Product(name={self.name!r}, price={self.price!r}, in_stock={self.in_stock!r})"

    def __eq__(self, other):
        return (self.name, self.price, self.in_stock) == (other.name, other.price, other.in_stock)
```

Với `@dataclass`:

```
from dataclasses import dataclass

@dataclass
class Product:
    name: str
```

```
    price: float
    in_stock: bool = True
```

3. Cấu hình nâng cao

`frozen=True`: bất biến (immutable)

aaa

```
@dataclass(frozen=True)
class User:
    username: str
    email: str

u = User("alice", "alice@example.com")
u.username = "bob"  #  Sẽ raise FrozenInstanceError
```

`order=True`: thêm các toán tử so sánh

```
@dataclass(order=True)
class Point:
    x: int
    y: int

Point(1, 2) < Point(2, 1)  #  True
```

`init=False`: loại bỏ `__init__`

```
@dataclass
class Token:
    value: str
    secret: str = "secret"
    valid: bool = field(init=False, default=True)
```

`init=False` giúp loại bỏ trường đó khỏi constructor.

4. `field()` – Tùy chỉnh thuộc tính

```
from dataclasses import dataclass, field
from typing import List

@dataclass
```

```
class Order:
    items: List[str] = field(default_factory=list)
```

Lý do dùng `default_factory=list` thay vì `items: List[str] = []` là vì:

- Trong Python, **giá trị mặc định là list (hoặc dict) dùng chung** giữa các thể hiện khác nhau nếu không cẩn thận.
- `default_factory` đảm bảo mỗi instance có list riêng biệt.

5. Post-init xử lý: `__post_init__`

Được gọi **ngay sau** `__init__` do `@dataclass` tạo.

```
@dataclass
class Product:
    name: str
    price: float

    def __post_init__(self):
        if self.price < 0:
            raise ValueError("Price must be non-negative")
```

6. So sánh sâu (deep comparison)

`dataclass` hỗ trợ `__eq__` mặc định — so sánh theo từng field, không theo địa chỉ bộ nhớ.

```
a = Product("Pen", 1.5)
b = Product("Pen", 1.5)
print(a == b)  #  True
```

7. Kế thừa với `dataclass`

```
@dataclass
class Animal:
    name: str

@dataclass
class Dog(Animal):
    breed: str
```

Bạn có thể kế thừa như class bình thường.

8. Kiểm soát `repr`, `eq`, `compare`, `hash` từng field

```
@dataclass
class User:
    username: str = field(compare=True)
    password: str = field(repr=False, compare=False)
```

- `repr=False`: ẩn khỏi `__repr__`
- `compare=False`: không dùng trong `__eq__` hay `__lt__`

9. Chuyển đổi `dataclass` thành dict

```
from dataclasses import asdict


p = Product("Mouse", 12.5)
print(asdict(p))  # {'name': 'Mouse', 'price': 12.5, 'in_stock': True}
```

10. Hạn chế của `dataclass`

- Không thay thế hoàn toàn ORM như Django Models (chỉ dùng cho logic nghiệp vụ/data layer)
- Không hỗ trợ property getter/setter tự động
- Không hỗ trợ kế thừa nhiều mức quá phức tạp (vẫn dùng được nhưng cần cẩn thận)

Nếu bạn đang làm dự án lớn với kiến trúc như DDD (Domain Driven Design), `dataclass` thường dùng để:

- Định nghĩa **DTO (Data Transfer Object)**
- Làm **request/response schema**
- Tạo các lớp đại diện cho **giá trị** (value objects)

Phiên bản #1
Được tạo 13 tháng 6 2025 02:00:21 bởi Đỗ Ngọc Tú
Được cập nhật 13 tháng 6 2025 02:10:04 bởi Đỗ Ngọc Tú