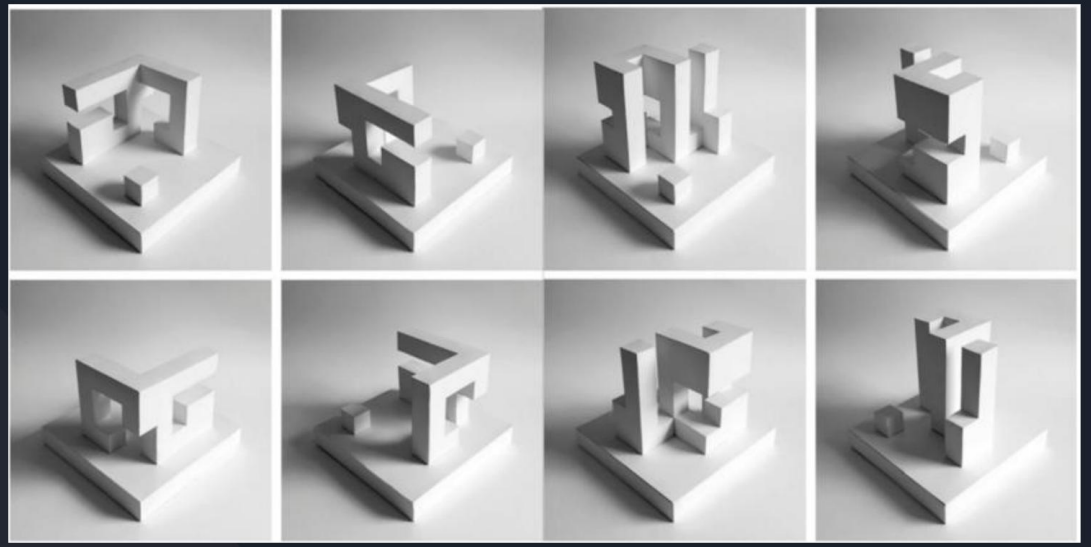
Giới thiệu Mô hình của Odoo



Model là gì?

Mô hình xác định cấu trúc lôgic của cơ sở dữ liệu và về cơ bản xác định cách dữ liệu có thể được lưu trữ, tổ chức và thao tác. Nói cách khác, một mô hình là một bảng thông tin có thể làm cầu nối với các bảng khác.

Các mô hình có thể được định cấu hình bằng cách đặt các thuộc tính trong định nghĩa của chúng. Thuộc tính quan trọng nhất là _name, thuộc tính này được yêu cầu và xác định tên cho mô hình trong hệ thống Odoo .

Đây là định nghĩa tối thiểu về một mô hình:

từ các mô hình nhập khẩu của odoo

lớp TestModel (mô hình.Model):

_name = "test.model"

Các loại mô hình

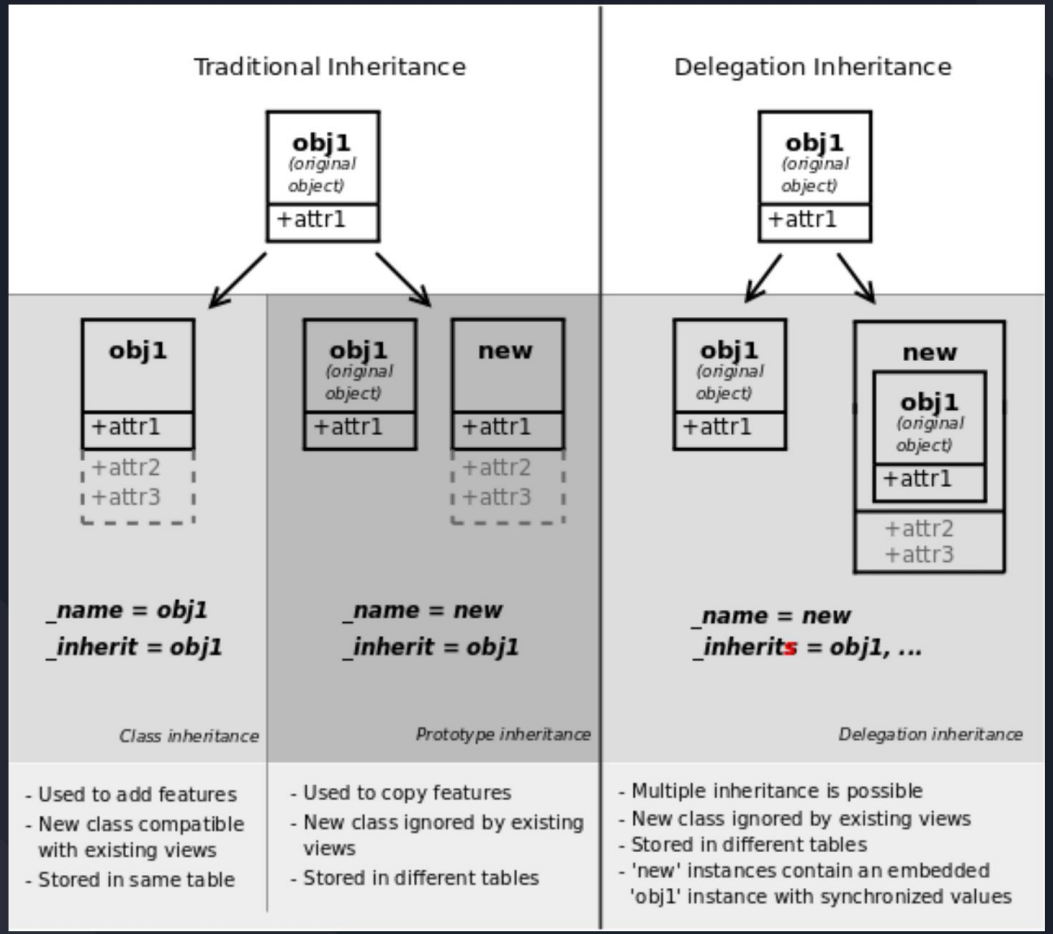
Các mô hình Odoo được tạo ra bằng cách kế thừa một trong những điều sau:

- Mô hình cho các mô hình cơ sở dữ liệu thường xuyên tồn tại

- TransientModel đối với dữ liệu tạm thời, được lưu trữ trong cơ sở dữ liệu nhưng tự động hút bụi thường xuyên

- AbstractModel cho các siêu lớp trừu tượng có nghĩa là được chia sẻ bởi nhiều mô hình kế thừa

Kế thừa mô hình



ModelClass

- Hệ thống tự động khởi tạo mọi mô hình một lần trên mỗi cơ sở dữ liệu. Các trường hợp đó đại diện cho các mô hình có sẵn trên mỗi cơ sở dữ liệu và phụ thuộc vào mô-đun nào được cài đặt trên cơ sở dữ liệu đó.

- Lớp thực tế của mỗi cá thể được xây dựng từ các lớp Python tạo và kế thừa từ mô hình tương ứng.

- Mỗi cá thể mô hình là một "tập bản ghi", tức là, một tập hợp các bản ghi của mô hình. Các tập bản ghi được trả về bằng các phương thức như duyệt, tìm kiếm hoặc truy cập trường. Các bản ghi không có biểu diễn rõ ràng: một bản ghi được biểu diễn dưới dạng một tập bản ghi của một bản ghi.

Kế thừa mô hình

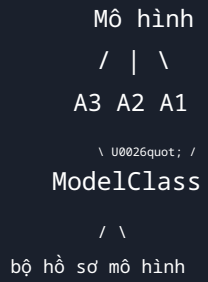
- Trong trường hợp đơn giản nhất, lớp đăng ký của mô hình kế thừa từ cls và các lớp khác xác định mô hình trong một hệ thống phân cấp phẳng.

- Sổ đăng ký chứa mô hình cá thể (ở bên trái). Đẳng cấp của nó, ModelClass, mang siêu dữ liệu suy luận được chia sẻ giữa tất cả các phiên bản của mô hình chỉ dành cho sổ đăng ký này. Ví dụ:

```
class A1 (Model):
     _name = 'a'

lớp A2 (Model):
     _inherit = 'a'

lớp A3 (Kiểu máy):
     _inherit = 'a'
```



Kế thừa mô hình

- Khi một mô hình được mở rộng bởi _inherit, các lớp cơ sở của nó là được sửa đổi để bao gồm lớp hiện tại và các lớp mô hình kế thừa khác.

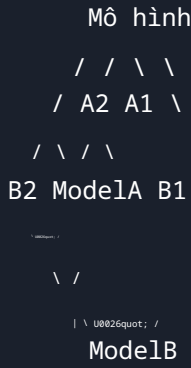
- Chúng tôi thực sự kế thừa từ các ModelClass khác, do đó các phần mở rộng cho một mô hình kế thừa sẽ hiển thị ngay lập tức trong lớp mô hình hiện tại , như trong ví dụ sau:

```
lớp A1 (Mẫu):
      _name = 'a'

lớp B1 (Người mẫu):
      _name = 'b'

hạng B2 (Người mẫu):
      _name = 'b'
      _inherit = ['a', 'b']

lớp A2 (Mẫu):
      _inherit = 'a'
```



ORM chung

1. search ()

Có một miền tìm kiếm, trả về một tập hợp các bản ghi phù hợp. Có thể trả về một tập hợp con các bản ghi phù hợp (tham số offset và giới hạn) và được sắp xếp (tham số thứ tự):

```
>>> # tìm kiếm mô hình hiện tại
>>> self.search ([('is_company', '=', True), ('customer', '=', True)])
res.partner (7, 18, 12, 14, 17, 19, 8, 31, 26, 16, 13, 20, 30, 22, 29, 15, 23, 28, 74)
>>> self.search ([('is_company', '=', True)], limit = 1) .name
'Agrolait'
```

ORM chung

2. tạo ()

Lấy một số giá trị trường và trả về một tập bản ghi có chứa bản ghi đã tạo:

```
>>> self.create ({'name': "New Name"})
res.partner (78)
```

ORM chung

3. ghi ()

Lấy một số giá trị trường, ghi chúng vào tất cả các bản ghi trong tập bản ghi của nó . Không trả lại bất cứ điều gì:

```
self.write ({'name': "Tên mới hơn"})
```

ORM chung

4. duyệt ()

Lấy id cơ sở dữ liệu hoặc danh sách id và trả về tập bản ghi, hữu ích khi id bản ghi được lấy từ bên ngoài Odoo (ví dụ: khứ hồi qua hệ thống bên ngoài) hoặc khi gọi các phương thức trong API cũ:

```
>>> self.browse ([7, 18, 12])
res.partner (7, 18, 12)
```

ORM chung

5. ref ()

Phương thức môi trường trả về bản ghi khớp với một id bên ngoài được cung cấp:

```
>>> env.ref ('base.group_public')
nhóm lại (2)
```

ORM chung

6. name_get ()

Trả lại biểu diễn văn bản của các đối tượng được yêu cầu cho các mối quan hệ x-nhiều

```
>>> self.name_get ()
[(66, “Tên tôi”)]
```